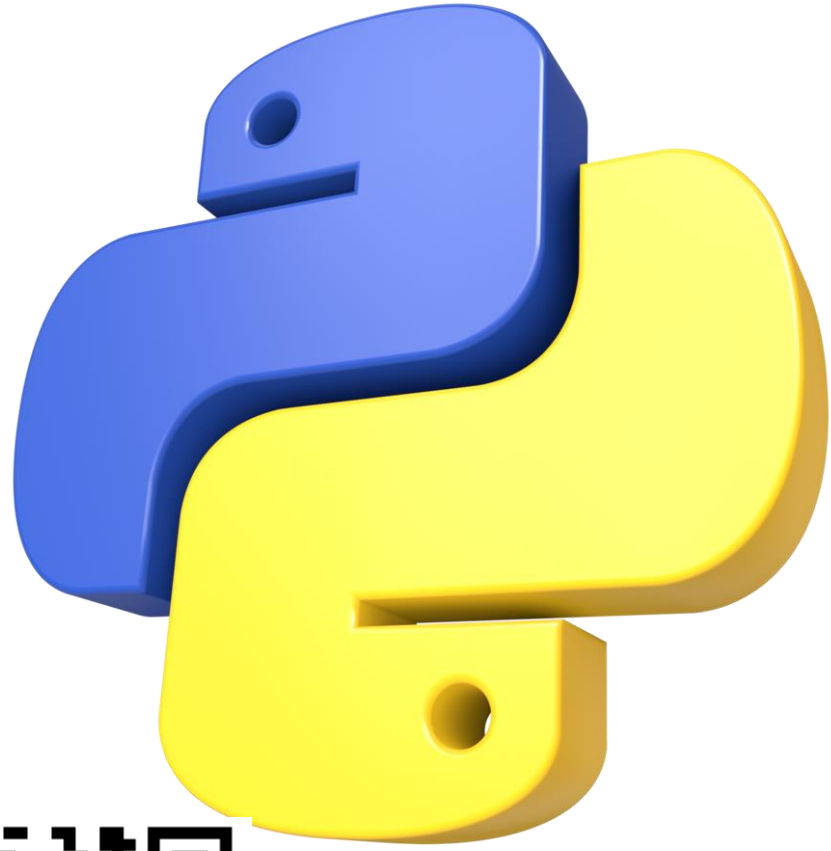


PROGRAMACIÓN EN PYYHON

Manual de referencia

Descripción breve

Este manual te ayudará como consulta para repasar conceptos para tus futuros proyectos.
Con el código QR podrás acceder a estos tutoriales en YouTube.

Pere Manel Verdugo Zamora
pereverdugo@gmail.com

Contenido

1.- Introducción a Python

Python “Siempre es mejor que complejo”.

¿Qué es Python?

- Es un lenguaje de programación interpretado cuya filosofía central es una sintaxis que favorezca un código limpio y legible.
- Es un lenguaje de programación **multiparadigma**, es decir, soporta diferentes estilos de programación como la programación orientada a objeto (POO), la programación imperativa y la funcional.
- Además, es importante resaltar que Python posee **tipado dinámico** (una misma variable puede tomar valores de distinto tipo en distintos momentos) y es **multiplataforma** (existe intérpretes de Python para Windows, Linux y Mac OS).
- Python es uno de los lenguajes de programación más populares hoy en día y se ha convertido en el favorito y además recomendado para dar los primeros pasos en el mundo de la programación.

HISTORIA

- Fue creado a finales de los años 80 con el científico holandés de la computación Guido van Rossum, en el Centro para las Matemáticas y la informática de Holanda.
- Python fue pensado para originalmente interactuar en el sistema operativo distribuido de por la investigación llamado Amoeba.
- Fue en 1991 cuando van Rossum publicó el código del lenguaje Python en su versión 0.9.9 con características avanzadas como clases con herencias, manejo de excepciones, funciones y tipos modulares.
- El nombre proviene del grupo británico de humoristas llamado Monty Python y el logotipo del lenguaje es representado por una serpiente pitón (Python, en inglés)
- Actualmente, Python es administrado por la Python Software Foundation y posee una licencia de código abierto, compatible con la licencia pública de GNU a partir de la versión 2.1.1.

CARACTERÍTICAS DE PYTHON

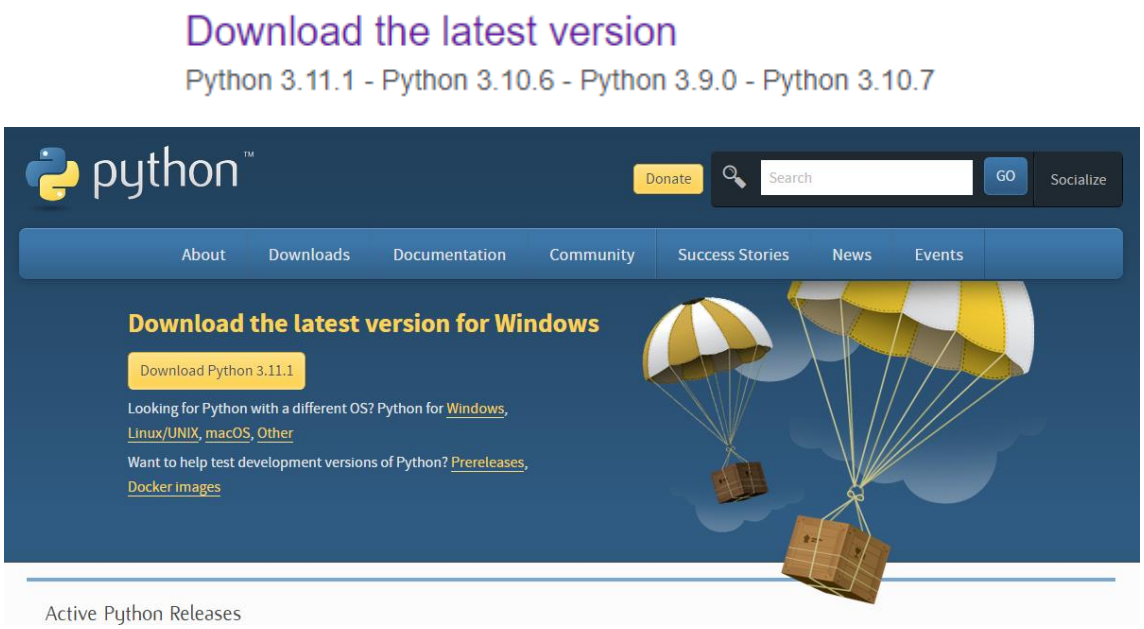
- Simple
- Propósito General (Desde aplicaciones de Escritorio a páginas web)
- Open Source
- Paradigna POO
- Lenguaje de alto nivel
- Incrustable
- Posee extensas librerías
- Sintaxis clara y limpia

¿QUÉ SE NECESITA PARA PRORAMAR EN PYTHON?

- El intérprete de Python correspondiente a tu sistema operativo y arquitectura del sistema.
- Un editor de texto (Subline Text, Block de Notas) o in IDE como Visual Studio Code.
- Muchas ganas de aprender !!!=D.

2.- Instalación del Intérprete de Python | Visual Studio Code

Vamos a escribir en un navegador Python.org.

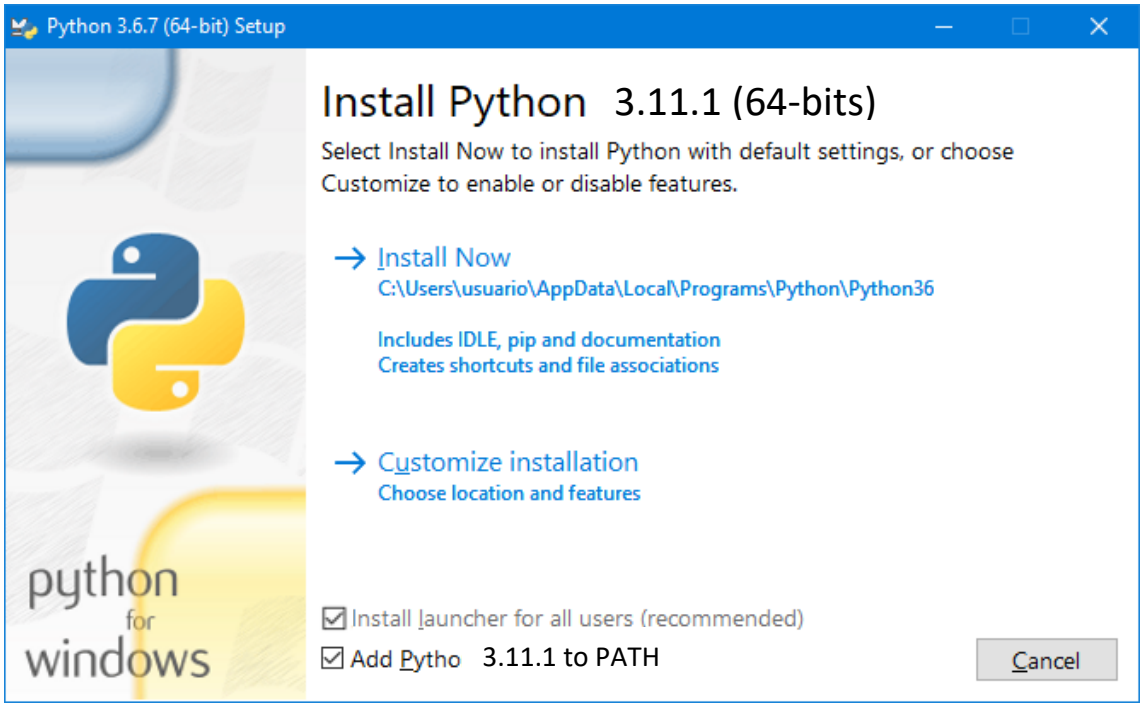


Actualmente tenemos disponible la versión 3.11.1.

Vamos a descargarla.

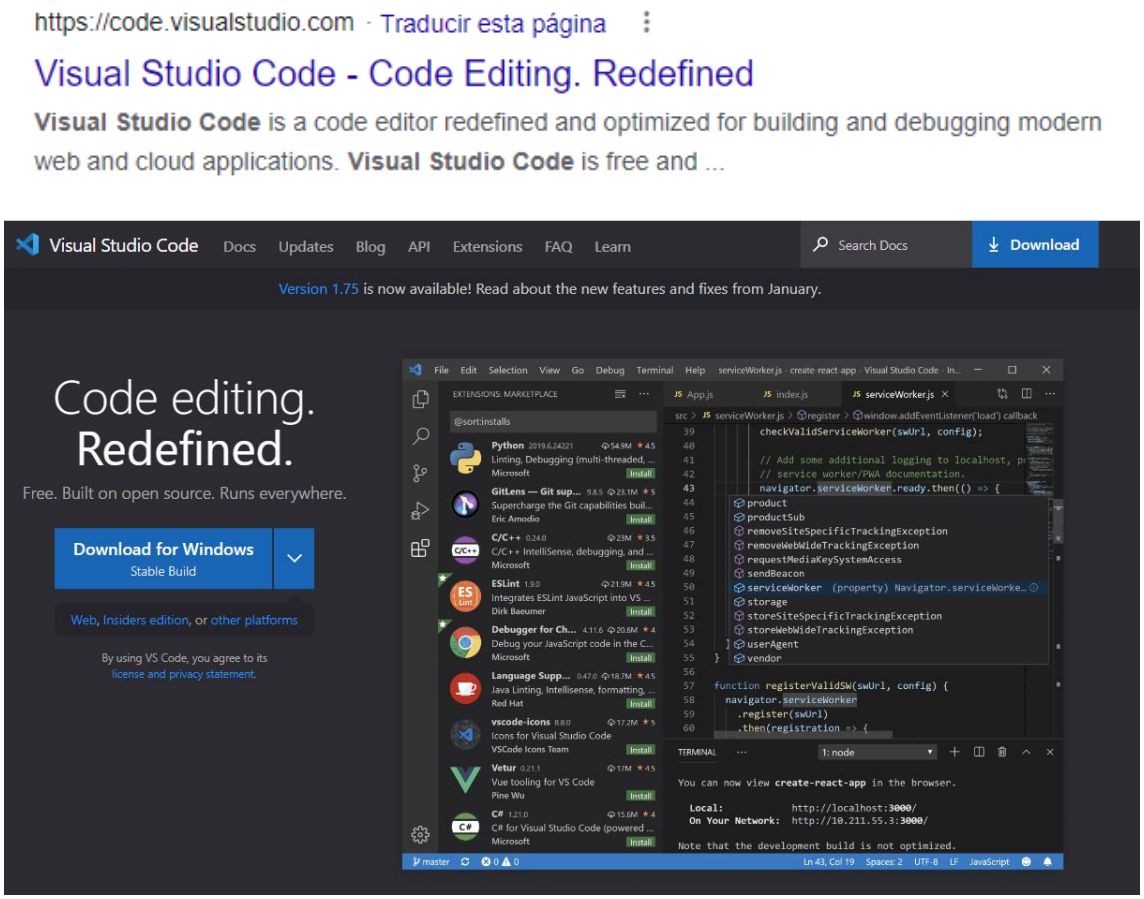
Que a continuación instalaremos.

Lo único que tenemos que tener en cuenta en activar la casilla Add Python, ya que de este modo tendremos acceso a Python desde cualquier carpeta de nuestro ordenador.

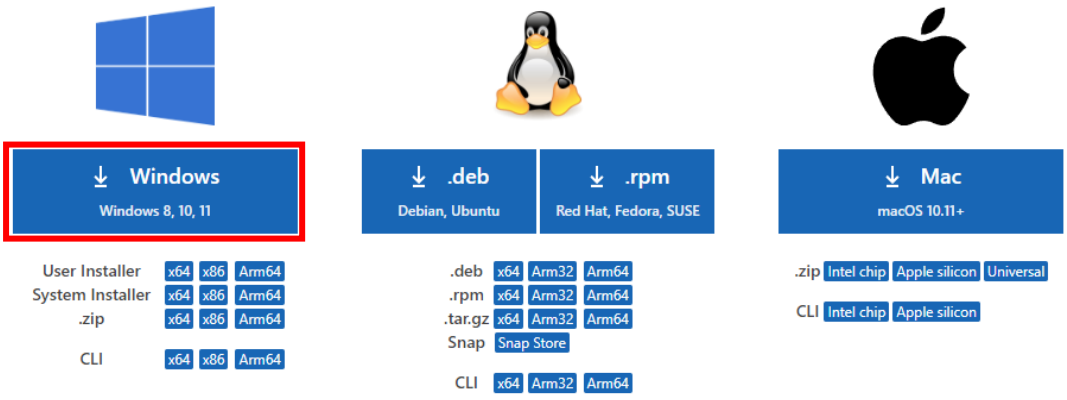


Lo siguiente es descargar Visual Studio Code.

Desde nuestro navegador vamos a buscar Visual Stuido Code.



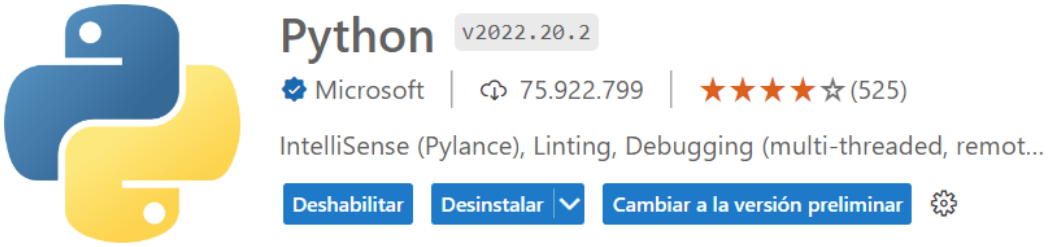
En la parte superior derecha seleccionaremos Download.



Yo como tengo Windows, seleccionaré para Windows.

A continuación procederemos a su instalación.

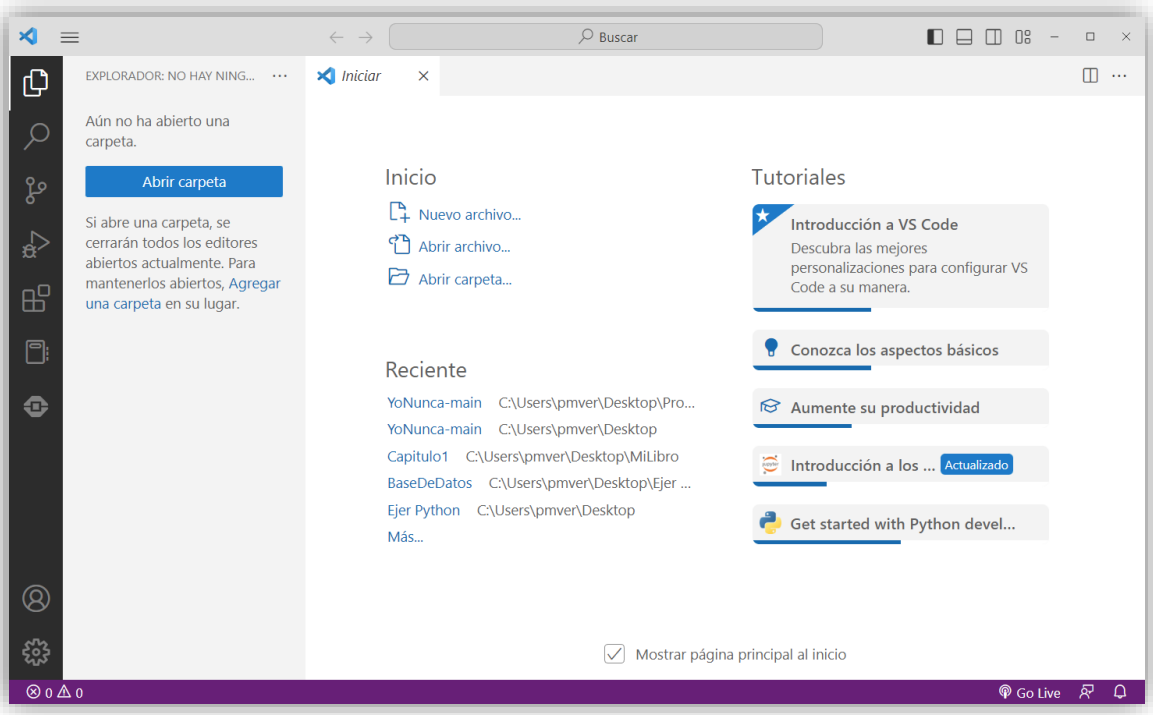
Tendrás que instalar la siguiente extensión:



Desde la siguiente opción situada en la parte izquierda.

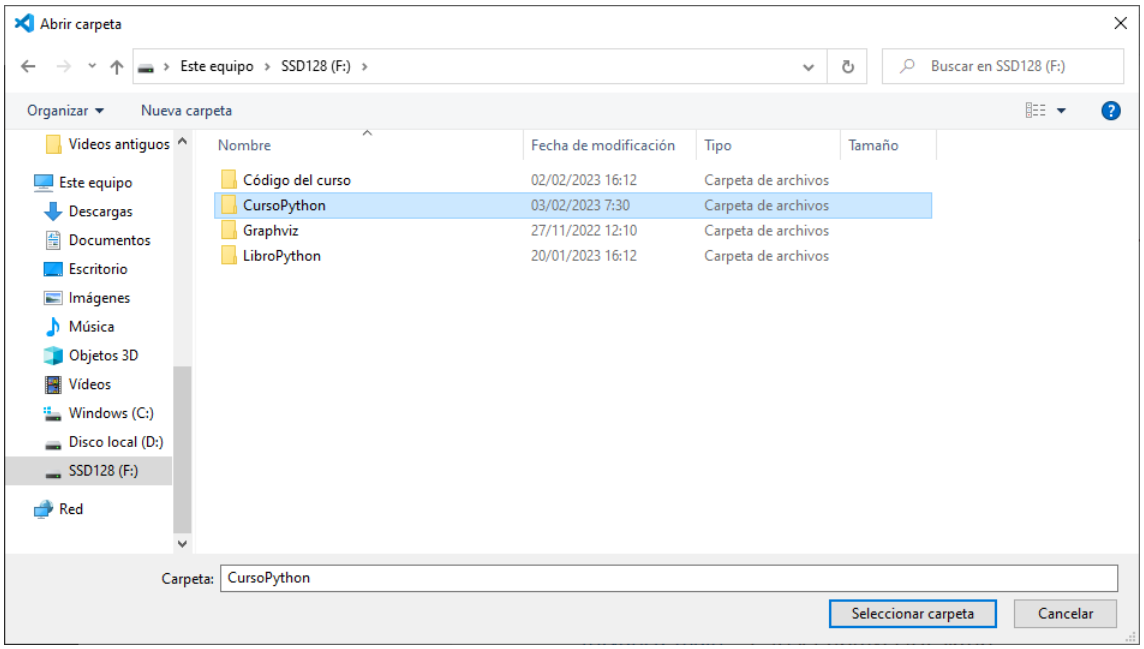
3.- Creación de Proyecto de Python & Hola Mundo

Vamos a ejecutar Visual Studio:

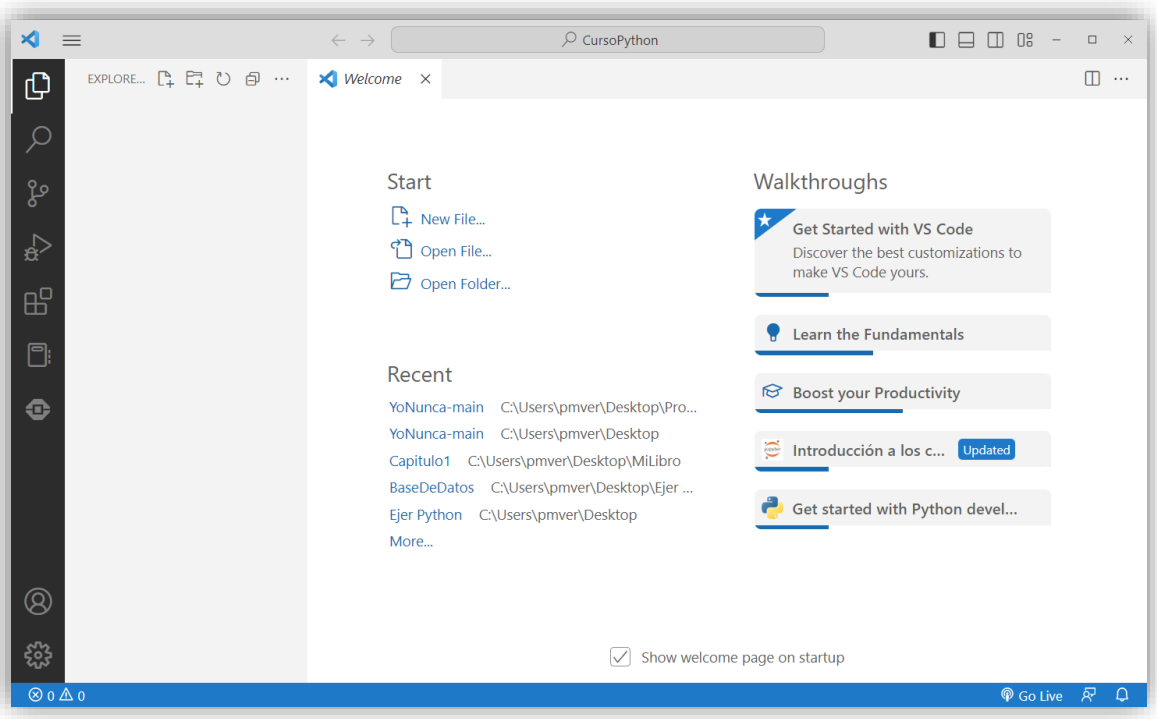


Previamente habremos creado una carpeta para nuestros proyectos, en nuestro caso la hemos llamado CursoPython.

Vamos a seleccionar el botón Abrir carpeta.

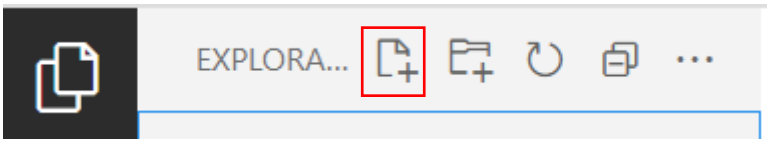


Seguido de seleccionar carpeta.

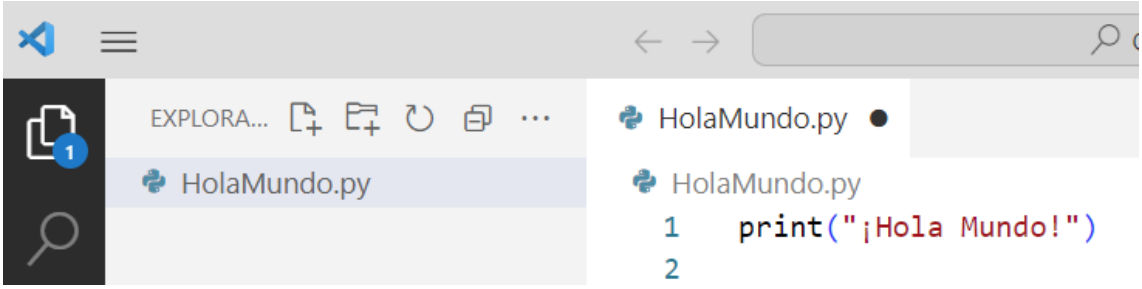


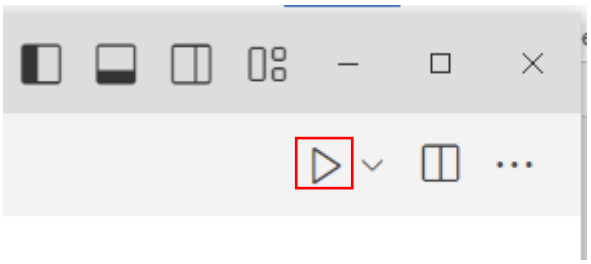
Cómo cambiar el idioma de Visual Studio Code al Español

1. Abre **Visual Studio Code**.
2. Pulsa Ctrl + Shift + P.
3. Escribe en la barra de búsqueda: «Configure Display Language«
4. Haz clic sobre «Install Aditional Languages«
5. Selecciona el pack de **idioma** que te interese de la lista de la izquierda. ..
6. Haz clic en «Install«



En la parte superior izquierda podremos crear nuevos archivos y carpetas, vamos a seleccionar nuevo archivo al que le vamos a llamar HolaMundo.py.





En la parte superior derecha encontraremos el botón de ejecutar.

Nos aparecerá una consola con su resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL                Python

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe
 f:/CursoPython/HolaMundo.py
¡Hola Mundo!
PS F:\CursoPython>
```

Este será el resultado:

Para ejecutarlo desde la ventana Cmd de Windows desde Visual Studio seleccionaremos las teclas Ctrol + Shift + C.

```
C:\WINDOWS\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.2486]
(c) Microsoft Corporation. Todos los derechos reservados.

F:\CursoPython>
```

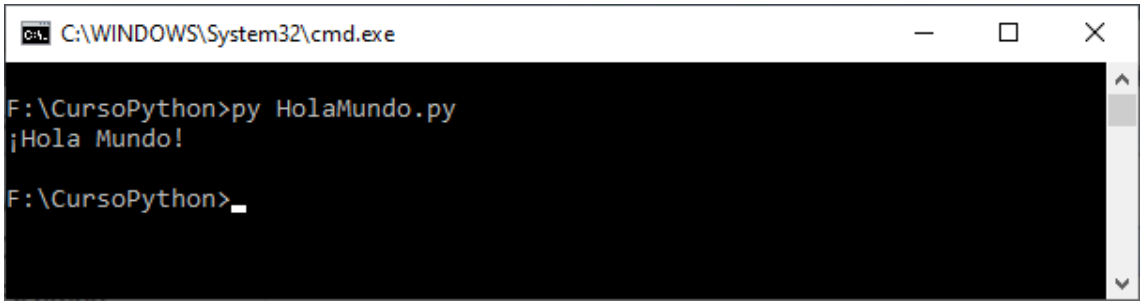
Desde esta ventana vamos a escribir dir seguido de intro.

```
C:\WINDOWS\System32\cmd.exe
F:\CursoPython>dir
 El volumen de la unidad F es SSD128
 El número de serie del volumen es: D446-1294

 Directorio de F:\CursoPython

03/02/2023  07:40    <DIR>          .
03/02/2023  07:40    <DIR>          ..
03/02/2023  07:41                26 HolaMundo.py
               1 archivos             26 bytes
               2 dirs  127.874.424.832 bytes libres

F:\CursoPython>
```

Nos encontramos en la carpeta donde hemos guardado nuestro primer proyecto.

Vamos a escribir cls más intro para borrar la pantalla.

Para ejecutar nuestro primer proyecto desde esta ventana vamos a escribir python HolaMundo.py o bien py HolaMundo.py.

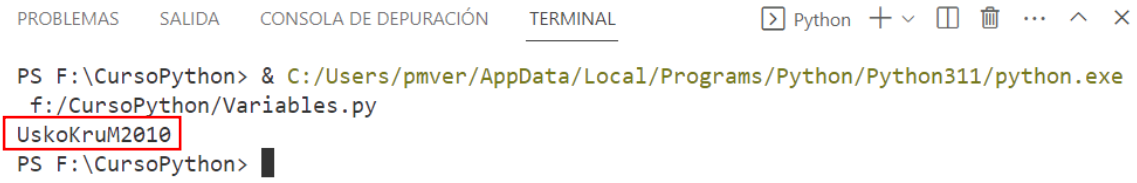


Enhorabuena si ya has llegado hasta este punto, significa que por ahora lo estás realizando muy bien.

4.- Variables en Python

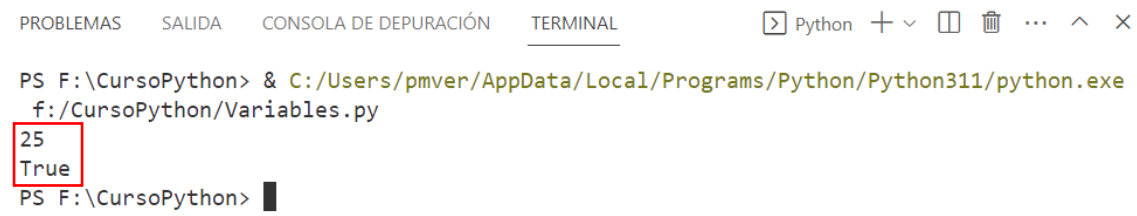
```
1    nombre = "UskoKruM2010"
2    print(nombre)
```

Vamos a ejecutar:




```
1    edad = 25
2    print(edad)
3
4    edad=True
5    print(edad)
```

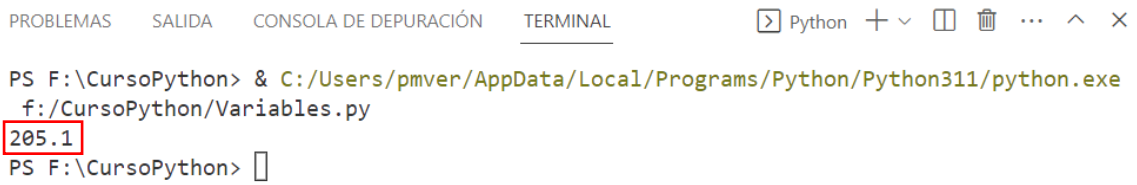
Vamos a ejecutar:



Como podrás observar podemos declarar variables y después cambiar el tipo de variable, la variable edad primero de define numérica de tipo int y a continuación de tipo booleano.

```
1    sueldo = 205.10
2    print(sueldo)
```

Este será el resultado:

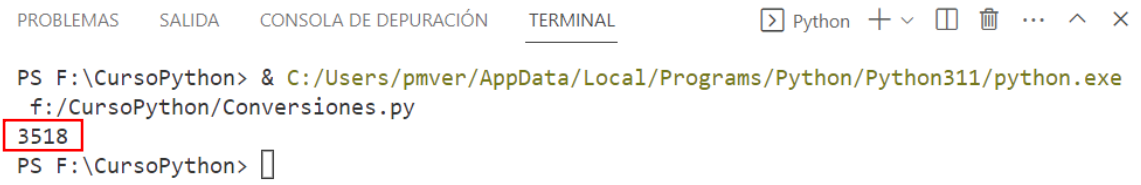


Hemos definido una variable numérica de tipo float.

5.- Conversiones (Casting) en Python

```
1    numero1 = "35"
2    numero2 = "18"
3    print(numero1 + numero2)
```

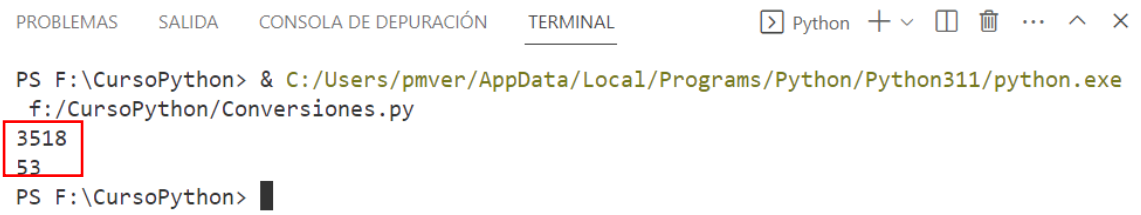
Vamos a ejecutar:



Las variables numero1 y numero2 se le están asignando dos número, pero estos al estar entre comillas los almacena de tipo texto (String) si queremos sumos dos variables de tiempo texto, el lugar de sumar lo que va a realizar en concatenar las dos variables unir 25 y 18.

```
1    numero1 = "35"
2    numero2 = "18"
3    print(numero1 + numero2)
4
5    num1 = int(numero1)
6    num2 = int(numero2)
7    print(num1 + num2)
```

Este será el resultado:

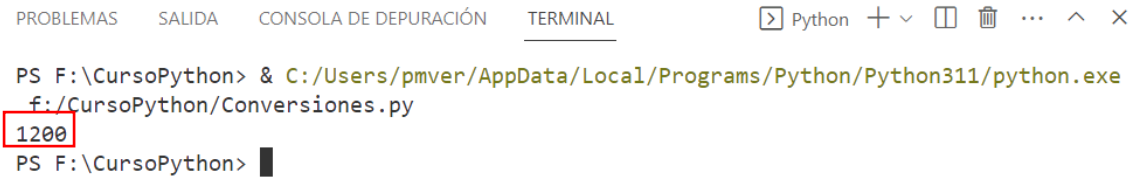


En la línea 5 y 6 hacemos lo que denomina Casting es asignar a una nueva variable el valor de otra variable pero reconvirtiendo el tipo de esta.

Al poner **num1 = int(numero1)** le estamos diciendo pasa el valor de la variable numero1 a la variable num1 pero cambiando el tipo a int (número entero) ya que la variable numero1 es de tipo texto (strintg). Sin el casting la variable num1 también sería de tipo texto.

```
1    sueldo = 1200.43
2    SueldoEntero = int(sueldo)
3    print(SueldoEntero)
```

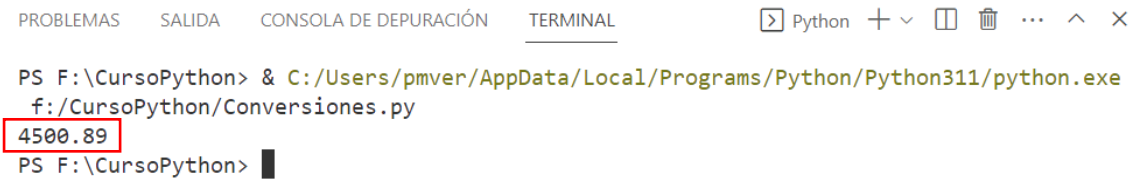
Vamos a ejecutar:



En la línea 1 a la variable sueldo se le asigna el valor 1200.43 al contener decimales esta variable es numérica de tipo Float (Flotante o decimal). Si a la variable SueldoEntero le asignamos el valor int(sueldo) le estamos diciendo que queremos la parte entera de sueldo, la nueva variable omitirá la parte decimal.

```
1    valor = "4500.89"
2    valorDecimal = float(valor)
3    print(valorDecimal)
```

Este será el resultado:



En la línea 1 le estamos asignando a la variable valor “4500.89” este al encontrarse entre comillas lo está almacenando en la variable valor como texto, es decir no podremos realizar operaciones matemáticas con dicha variable.

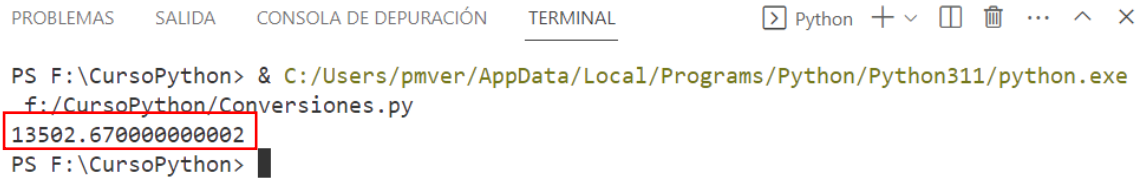
En la línea 2 al asignar a la variable valorDecimal el valor de la variable valor pero con una reconversión de tipo (Float) con esta segunda variable si podremos realizar operaciones matemáticas.

Vamos a modificar la línea 3.

```
3    print(valorDecimal*3)
```

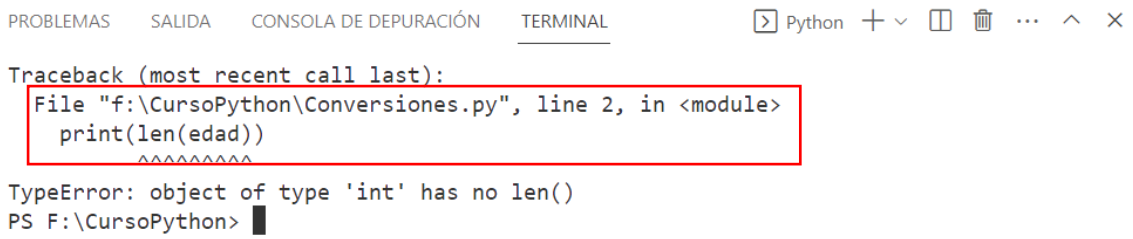
Le estamos diciendo que imprima el valor que contiene valorDecimal multiplicado por 3.

Este será el resultado:




```
1    edad = 100
2    print(len(edad))
```

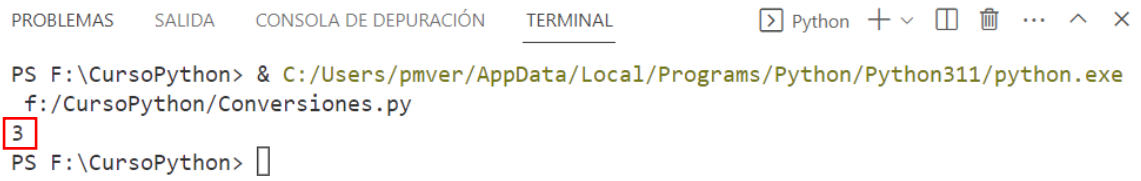
Vamos a ejecutar:



El método len() te dice el número de caracteres que tiene una variable de tipo texto, en este caso al ser una variable de tipo numérico, esto lo que hace es retornarnos un error.

```
1    edad = 100
2    print(len(str(edad)))
```

Al ejecutar:



Al reconvertir la variable a Sting o cadena por mediación str() ya nos dice el número de caracteres que tiene la variable edad ya que hemos realizado un Casting reconvirtiéndolo a variable de tipo texto.

6.- Operaciones Matemáticas y Comentarios en Python

```
1   entero = 23
2   decimal = 31.78
3   complejo = 12 + 5j
4   booleano = True
5
6   print(entero)
7   print(decimal)
8   print(complejo)
9   print(booleano)
```

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL        Python

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe
 f:/CursoPython/Numeros_Operaciones.py
23
31.78
(12+5j)
True
PS F:\CursoPython>
```

Python soporta los números complejos ya que permite cálculos complejos aplicaciones matemáticas y científicas de bastante complejidad.

Vamos a ver los comentarios:

```
1   entero = 23
2   decimal = 31.78
3   complejo = 12 + 5j
4   #booleano = True
5
6   """
7   print(entero)
8   print(decimal)
9   print(complejo)
10  print(booleano)
11  """
```

En la línea 4 si agregamos una almohadilla al principio de una línea es pasa a ser un comentario, cuando se ejecute el programa esta línea será ignorada.

Si queremos comentar más de una línea, pondremos antes de la primera línea a comentar 3 comillas dobles “”” y después de la última línea a comentar tres comillas dobles.

Vamos a ver las operaciones matemáticas:

```
1    num1 = 20
2    num2 = 4
3
4    print("Suma:", (num1 + num2))
5    print("Resta:", (num1 - num2))
6    print("Multiplicación:", (num1 * num2))
7    print("División:", (num1 / num2))
```

Para concatenar texto y valores, el texto irá entre comillas seguido de una coma y a continuación la variable numérica o la operación entre variables.

Si ejecutamos:

En la línea 8 vamos a realizar la división exacta:

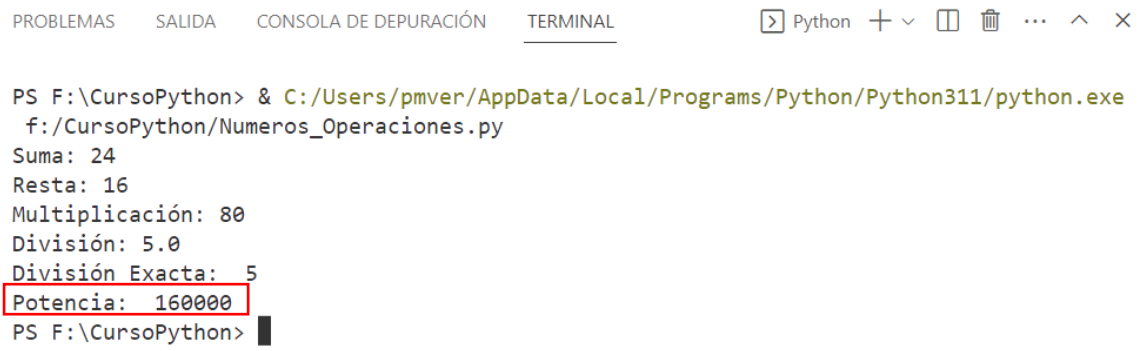
```
8    print("División Exacta: ", (num1 // num2))
```

Este será el resultado:

En la línea 9 vamos a utilizar la potencia.

```
9    print("Potencia: ", (num1 ** num2))
```

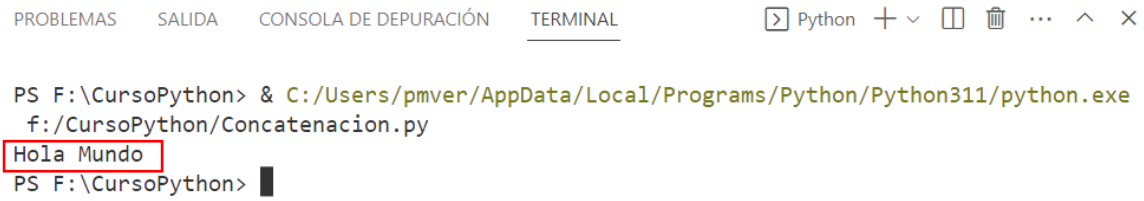
Este será el resultado:


PROBLEMAS
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL
Python
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe
f:/CursoPython/Numeros_Operaciones.py
Suma: 24
Resta: 16
Multiplicación: 80
División: 5.0
División Exacta: 5
Potencia: 160000
PS F:\CursoPython>

7.- Concatenación en Python

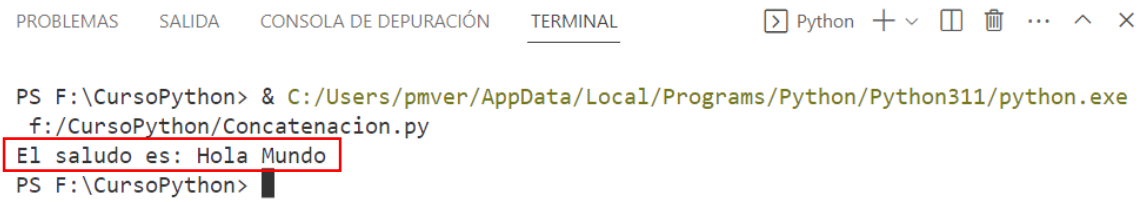
```
1    texto1 = "Hola"
2    texto2 = "Mundo"
3    textoFinal = texto1 + " " + texto2
4    print(textoFinal)
```

Este será el resultado:



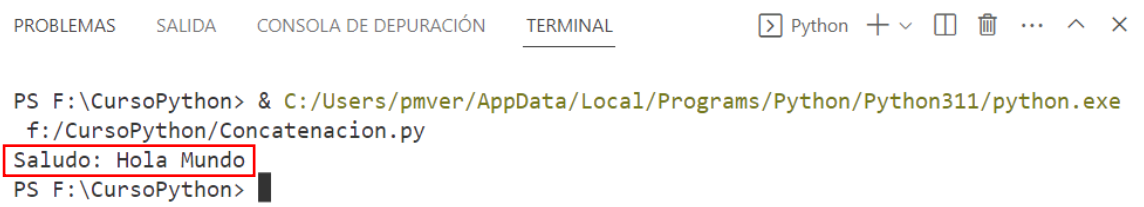

```
1    texto1 = "Hola"
2    texto2 = "Mundo"
3    print("El saludo es: %s %s" %(texto1, texto2))
```

Este será el resultado:




```
1    texto1 = "Hola"
2    texto2 = "Mundo"
3    saludoFinal = "Saludo: {0} {1}".format(texto1, texto2)
4    print(saludoFinal)
```

Este será el resultado:



También funcionará de la siguiente forma.

```
3    saludoFinal = "Saludo: {} {}".format(texto1, texto2)
```

Podemos omitir los números si respetamos el orden.

Ahora vamos a ver el último ejemplo:

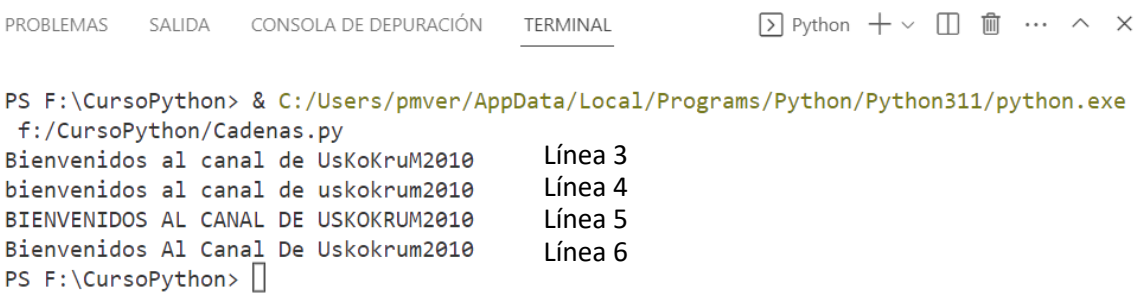
```
1   texto1 = "Hola"
2   texto2 = "Mundo"
3   saludoFinal = "Saludo: {x} {y}".format(x=texto1, y=texto2)
4   print(saludoFinal)
```

El resultado será el mismo que en el ejemplo anterior.

8.- Funciones de Cadena (String) en Python

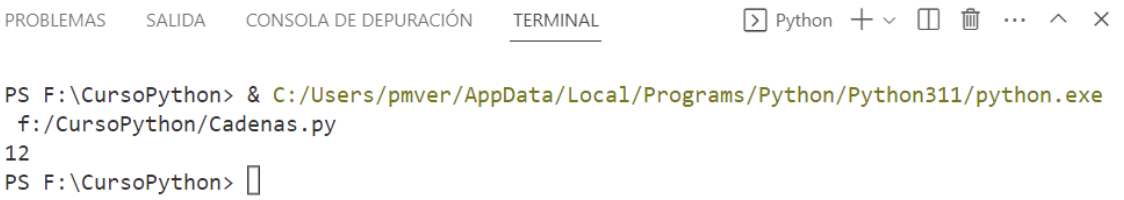
```
1   texto = "Bienvenidos al canal de UsKoKruM2010"
2
3   print(texto)
4   print(texto.lower()) # lower() Pasa texto minúsculas
5   print(texto.upper()) # upper() Pasa texto mayúsculas
6   print(texto.title()) # Pone la primera letra en mayúsculas
```

Este será el resultado:




```
1   texto = "Bienvenidos al canal de UsKoKruM2010"
2   print(texto.find("al"))
```

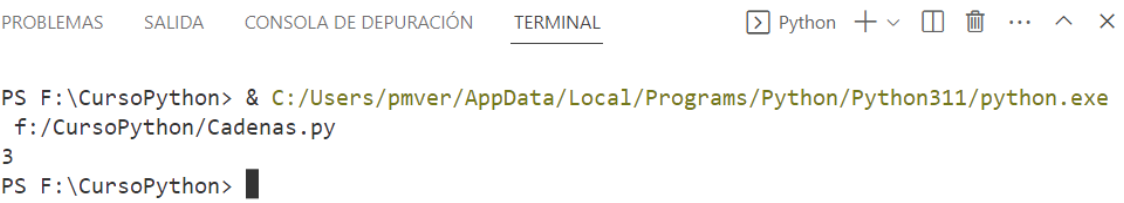
Este será el resultado:



La cadena “al” se encuentra en la posición 12.

```
1   texto = "Bienvenidos al canal de UsKoKruM2010"
2   print(texto.count("e"))
```

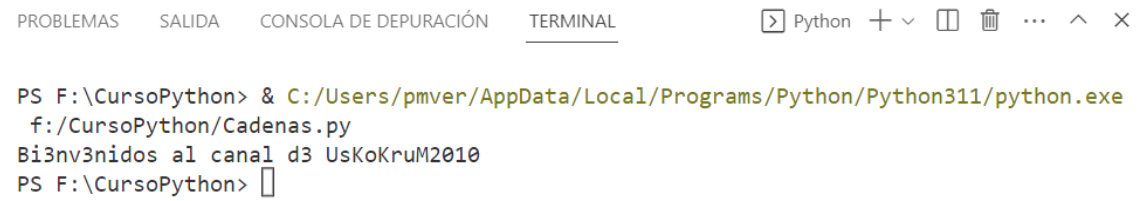
Este será el resultado:



El método texto.count(“e”) nos dice cuantas veces aparecen la letra e en la variable texto.

```
1    texto = "Bienvenidos al canal de UsKoKruM2010"
2    textoFinal = texto.replace("e","3")
3    print(textoFinal)
```

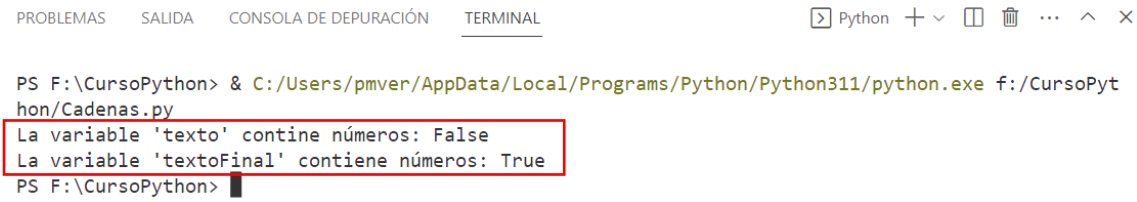
Este será el resultado:



Reemplaza todos los caracteres ‘e’ por el carácter ‘3’ de toda la cadena.

```
1    texto1 = "Hola"
2    texto2 = "1234"
3
4    print("La variable 'texto' contine números: " + str(texto1.isnumeric()))
5    print("La variable 'textoFinal' contiene números: " + str(texto2.isnumeric()))
```

Este será el resultado:



El método .isnumeric() retorna un valor booleano True o False y una variable contiene datos numéricos o no, es decir una variable de tipo texto que contenga solo número nos retornará un True.

```
1    texto = "Bienvenidos al canal de uskoKrum2010"
2    CadenaLista = texto.split()
3    print(CadenaLista)
```

Este será el resultado:

Retorna una lista de toda la cadena que separa por espacios.

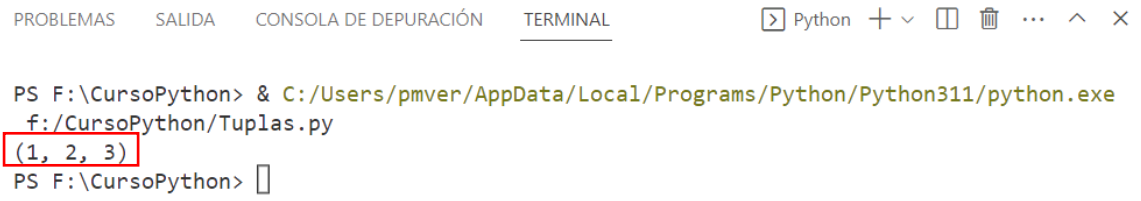
texto.split() es igual a texto.split(“ “) ➔ este método su valor por defecto es el espacio en blanco.

9.- Tuplas (Tuplas) en Python

Tupla: Es una estructura de datos propia de Python que permite almacenar distintos valores, son inmutables, no cambian sus valores una vez inicializadas.

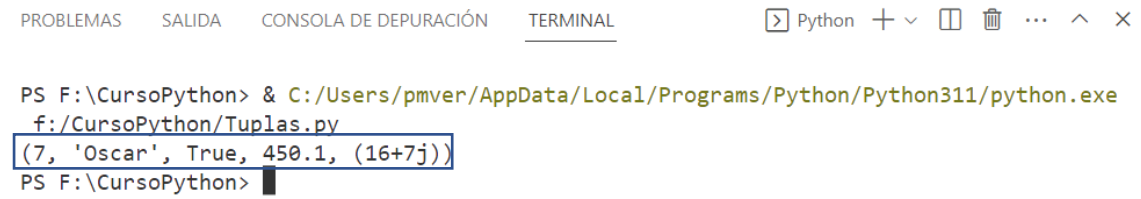
```
1    tupla = (1,2,3)
2    print(tupla)
```

Este será el resultado:




```
1    tupla = (7, "Oscar", True, 450.1, 16+7j)
2    print(tupla)
```

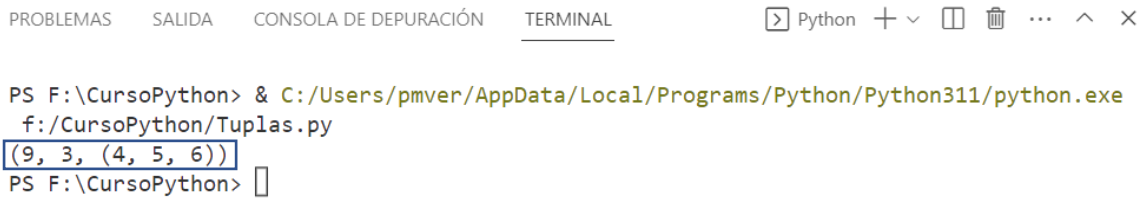
Este será el resultado:



Puede contener datos de distintos tipos.
Numéricos, Textos, booleanos, Decimales y números complejos.

```
1    tupla = (9,3, (4, 5, 6))
2    print(tupla)
```

Este será el resultado:



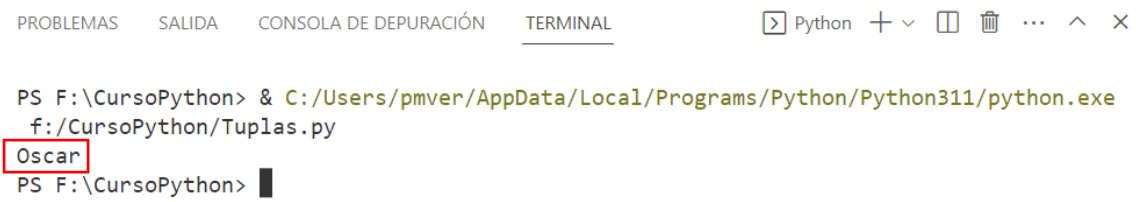

```
1    tupla = (7, "Oscar", True, 450.1, 16 + 7j)
2    print(tupla[1])
```

La tupla se guarda de la siguiente forma:

Tupla				
0	1	2	3	4
7	“Oscar”	True	450.1	16 + 7j

Como queremos visualizar la posición 1, ten en cuenta que se empieza por la posición 0.

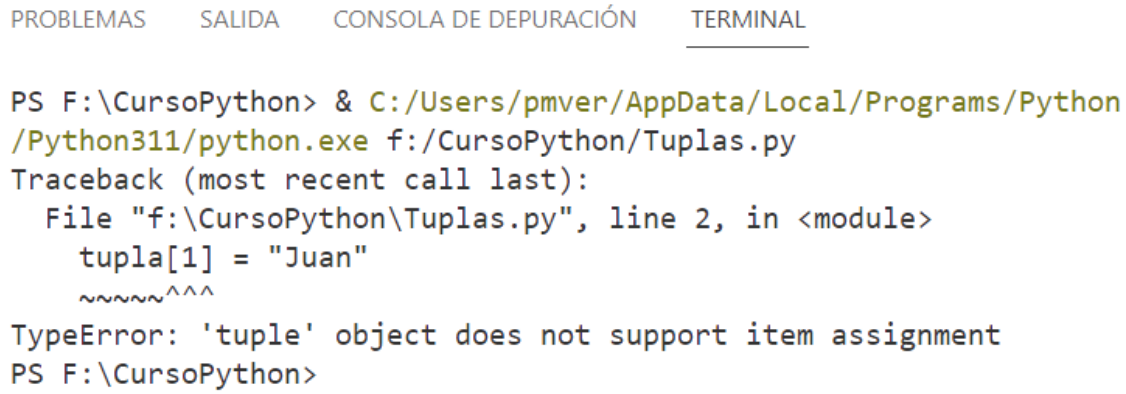
Este será el resultado:



Anteriormente hemos dicho que una tupla es inmutable es decir no se puede cambiar el valor, vamos a intentarlo a ver qué pasa.

```
1    tupla = (7, "Oscar", True, 450.1, 16 + 7j)
2    tupla[1] = "Juan"
```

Este será el resultado cuando ejecutemos:

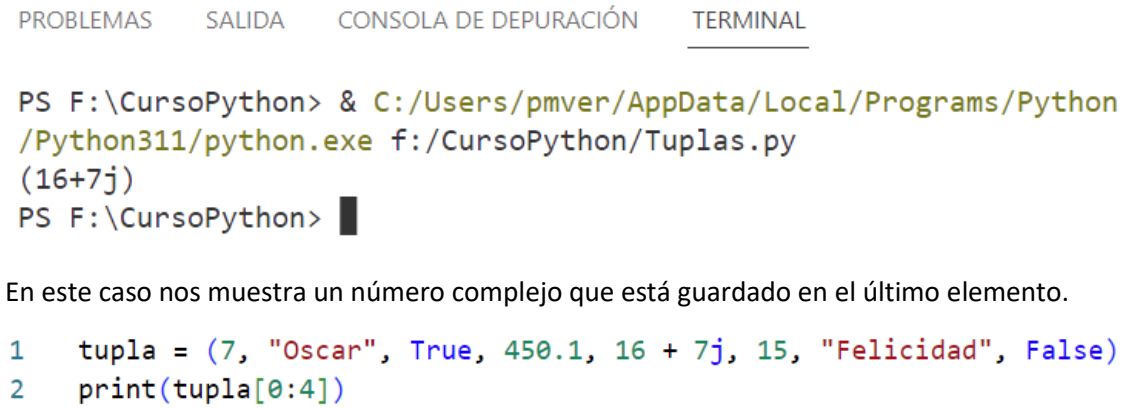


Nos muestra el siguiente error: el objeto no admite la asignación de elementos.

```
1    tupla = (7, "Oscar", True, 450.1, 16 + 7j)
2    print(tupla[-1])
```

Si queremos acceder al último elemento de la tupla será -1 al penúltimo -2, antepenúltimo -3 y así sucesivamente.

Este será el resultado:

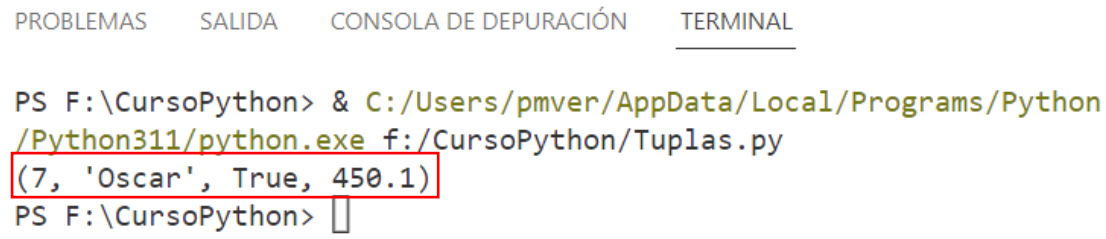


En este caso nos muestra un número complejo que está guardado en el último elemento.

```
1    tupla = (7, "Oscar", True, 450.1, 16 + 7j, 15, "Felicidad", False)
2    print(tupla[0:4])
```

De la tupla queremos mostrar los elementos desde la posición 0 hasta la 4, mostrará las posiciones 0, 1, 2, y 3 la 4 ya no la incluye.

Este será el resultado:

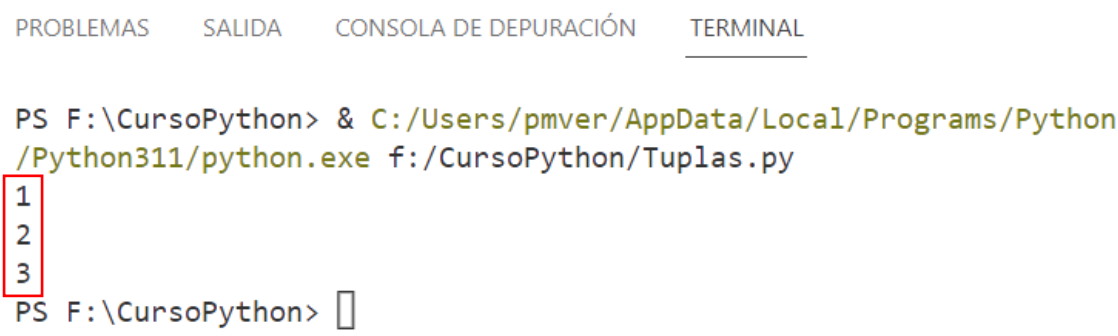


print(tuplas[-2] → Muestra el segundo elemento empezando por el último.

```
1    tupla = (1, 2, 3)
2    a, b, c = tupla
3
4    print(a)
5    print(b)
6    print(c)
```

A cada variable se le asigna un elemento de tupla.

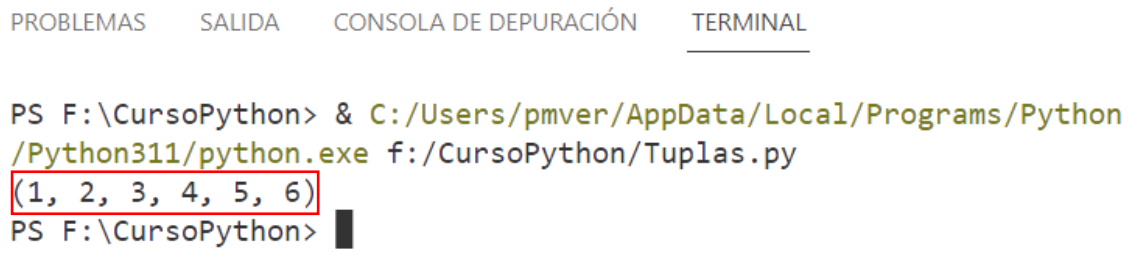
Vamos a ejecutar:



Vamos a unir varias tuplas:

```
1    tupla1 = (1, 2, 3)
2    tupla2 = (4, 5, 6)
3    tuplaFinal = tupla1 + tupla2
4
5    print(tuplaFinal)
```

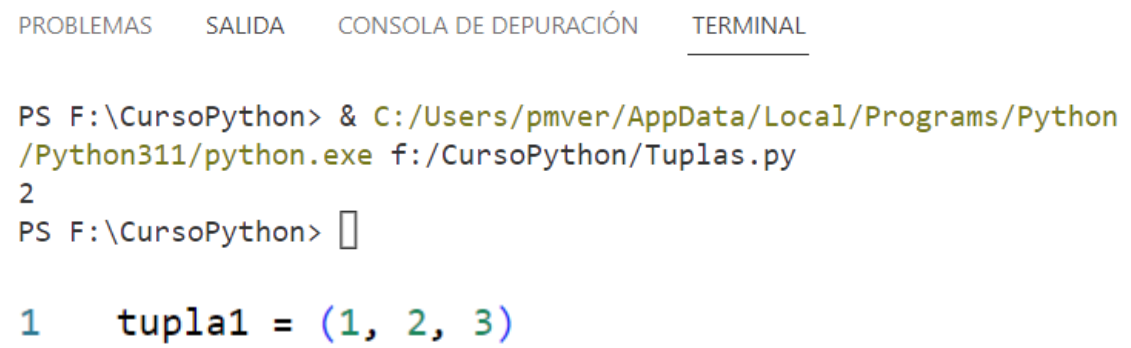
Este será el resultado:



```
1    tupla1 = (1, 2, 3, 4)
2    tupla2 = (4, 5, 6)
3    tuplaFinal = tupla1 + tupla2
4
5    print(tuplaFinal.count(4))
```

Queremos saber cuanto elementos con el número 4 tiene la tuplaFinal.

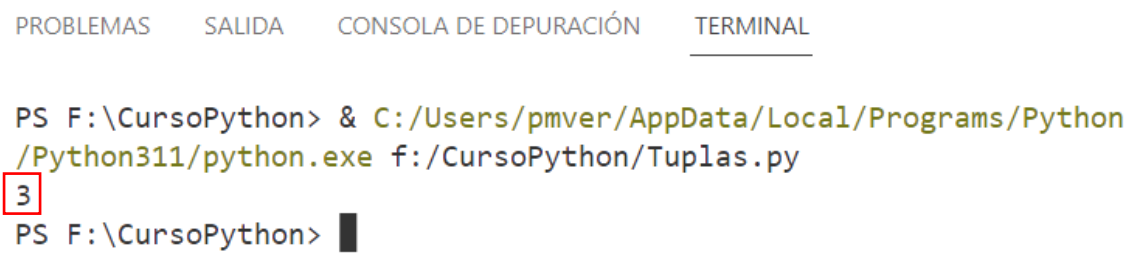
Este será el resultado:




```
1    tupla1 = (1, 2, 3)
2    tupla2 = (4, 5, 6)
3    tuplaFinal = tupla1 + tupla2
4
5    print(tuplaFinal.index(4))
```

Con .index() nos muestra en que posición se encuentra un elemento.

Este será el resultado:



10.- Listas [List] en Python

Listas: Son estructuras de datos que nos permiten almacenar distintos valores (equivalentes a los arrays en otros lenguajes de programación).

Son estructuras dinámicas, pueden MUTAR.

```
1    lista1 = ["Oscar", 25, 98.3]
2    print(lista1)
```

Si ejecutamos este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
['Oscar', 25, 98.3]
PS F:\CursoPython>
```

```
1    lista1 = ["Oscar", 25, 98.3]
2    print(lista1)
3    print(lista1[:]) # Imprimir toda la lista
4    print(lista1[2]) # Imprimir el tercer elemento.
5    print(lista1[-1])# Imprimir el último elemento.
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
['Oscar', 25, 98.3]
['Oscar', 25, 98.3]
98.3
98.3
PS F:\CursoPython>
```

```
1    lista1 = ["Oscar", 25, 98.3, True, "Flavio", 56.3]
2    print(lista1[0:3])  # Mostramos desde la posición 0 hasta 3 elementos
3    print(lista1[:2])   # Mostrar los dos primeros elementos.
4    print(lista1[3:])   # Mostrar desde la posición 3 hasta el final.
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
['Oscar', 25, 98.3]
```

```
['Oscar', 25]
[True, 'Flavio', 56.3]
PS F:\CursoPython>
```

```
1   lista1 = ["Oscar", 25, 98.3, True, "Flavio", 56.3]
2
3   lista1.append("UskoKruM2010") # Añadimos un elemento al final.
4   print(lista1)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
['Oscar', 25, 98.3, True, 'Flavio', 56.3, 'UskoKruM2010']
PS F:\CursoPython>
```

El método append() agrega un elemento al final de la lista.

```
1   lista1 = ["Oscar", 25, 98.3, True, "Flavio", 56.3]
2   print(lista1)
3   lista1.insert(4,"Perú") # El nuevo elemento se inserta en el
4   print(lista1)           # indice 4
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
['Oscar', 25, 98.3, True, 'Flavio', 56.3]
['Oscar', 25, 98.3, True, 'Perú', 'Flavio', 56.3]
PS F:\CursoPython>
```

```
1   lista1 = ["Oscar", 25, 98.3, True, "Flavio", 56.3]
2   print(lista1)
3   lista1.extend(["Alejandro", 110, False])
4   print(lista1)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
['Oscar', 25, 98.3, True, 'Flavio', 56.3]
['Oscar', 25, 98.3, True, 'Flavio', 56.3, 'Alejandro', 110, False
```

```
]
PS F:\CursoPython>
```

Hemos extendido la lista con 3 elementos más.

```
1    lista1 = ["Oscar", 25, 98.3, True, "Flavio", 56.3]
2    print(lista1.index("Flavio"))
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
4
PS F:\CursoPython>
```

Nos muestra en que posición del índice se encuentra “Flavio”.

```
1    lista1 = ["Oscar", 25, 98.3, True, "Flavio", 56.3]
2    lista1.remove(56.3)
3    print(lista1)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
['Oscar', 25, 98.3, True, 'Flavio']
PS F:\CursoPython>
```

Elimina el elemento con el valor 56.3, cuando mostramos de nueva la lista este elemento ya no está.

Si intentamos eliminar un elemento que no existe nos retornará un error.

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
Traceback (most recent call last):
  File "f:\CursoPython\Listas.py", line 2, in <module>
    lista1.remove(32)
    ^^^^^^^^^^^^^^^^^
ValueError: list.remove(x): x not in list
PS F:\CursoPython>
```

```python
lista1 = ["Oscar", 25, 98.3, True, "Flavio", 56.3]
ElementoEliminado = lista1.pop() # Elimina el último elemento
print(f"Se ha eliminado el ultimo elemento {ElementoEliminado}")
print(lista1)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
Se ha eliminado el ultimo elemento 56.3
['Oscar', 25, 98.3, True, 'Flavio']
PS F:\CursoPython>
```

.pop() → Elimina el último elemento de la lista y lo asigna a la variable ElementoEliminado.

```python
lista1 = ["Oscar", 25, 98.3, ]
lista2 = [True, "Flavio", 56.3]
lista3 = lista1 + lista2
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
['Oscar', 25, 98.3, True, 'Flavio', 56.3]
PS F:\CursoPython>
```

Podemos sumar listas.

```python
lista1 = ["Oscar", 25, 98.3, True,"Flavio", 56.3]

print(lista1 * 3)
```

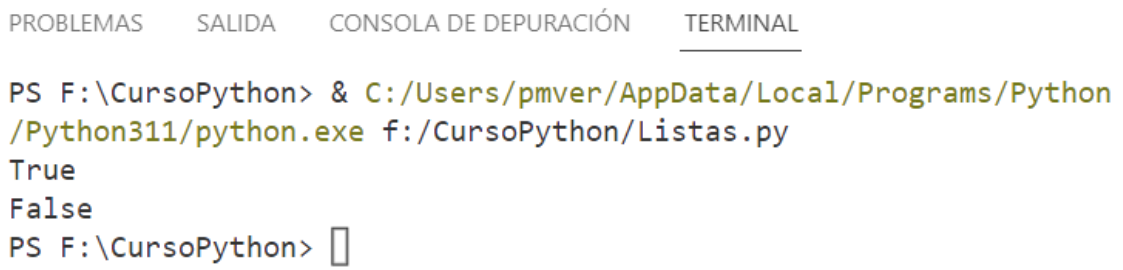
Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/Listas.py
['Oscar', 25, 98.3, True, 'Flavio', 56.3, 'Oscar', 25, 98.3, True
, 'Flavio', 56.3, 'Oscar', 25, 98.3, True, 'Flavio', 56.3]
PS F:\CursoPython>
```

Podemos multiplicar una lista y este mostrará los elementos repitiéndose el número de veces por el número que hemos multiplicado.

```python
lista = ["Oscar", 25, 98.3, True,"Flavio", 56.3]

print("Oscar" in lista)
print(False in lista)
```

Este será el resultado:



in → Retorna con un valor booleano True o False si un elemento se encuentra en la lista o no.

11.- Diccionarios {Diccionario} en Python

Diccionarios: Son estructuras de datos que nos permiten almacenar distintos valores.

Es que los datos se almacenan asociados a una clave única, tenemos una relación clave : valor.

Los elementos almacenados están en desorden, el orden es diferente a la forma de almacenamiento.

```
1   miDiccionario = {"España":"Madrid","Perú":"Lima", "Alemania":"Berlín"}
2   print(miDiccionario["Perú"])
```

Si ejecutamos este será el resultado:

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/P
ython311/python.exe f:/CursoPython/Diccionarios.py
Lima
PS F:\CursoPython>
```

Cuando le indicamos la clave asociada que se relaciona con el valor que deseamos obtener.

```
1   miDiccionario = {"España":"Madrid","Perú":"Lima", "Alemania":"Berlín"}
2   print(miDiccionario)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/P
ython311/python.exe f:/CursoPython/Diccionarios.py
{'España': 'Madrid', 'Perú': 'Lima', 'Alemania': 'Berlín'}
PS F:\CursoPython>
```

```
1   miDiccionario = {"España":"Madrid","Perú":"Lima", "Alemania":"Berlín"}
2   miDiccionario["Venezuela"] = "Caracas"
3   print(miDiccionario)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python
.exe f:/CursoPython/Diccionarios.py
{'España': 'Madrid', 'Perú': 'Lima', 'Alemania': 'Berlín', 'Venezuela': 'Caracas'}

PS F:\CursoPython>
```

Agregamos un nuevo elemento.

```
1   miDiccionario = {"España":"Madrid","Perú":"Lima", "Alemania":"Berlín"}
2   miDiccionario["España"] = "Barcelona"
3   print(miDiccionario)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python
.exe f:/CursoPython/Diccionarios.py
{'España': 'Barcelona', 'Perú': 'Lima', 'Alemania': 'Berlín'}
PS F:\CursoPython>
```

Como la clave 'España" ya existe lo que hace es reemplazar su valor.

```
1   miDiccionario = {"España":"Madrid","Perú":"Lima", "Alemania":"Berlín"}
2   del miDiccionario["España"]
3   print(miDiccionario)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python
.exe f:/CursoPython/Diccionarios.py
{'Perú': 'Lima', 'Alemania': 'Berlín'}
PS F:\CursoPython>
```

Eliminamos la clave y su valor asociado.

```
1   dicPersonal = {"Nombre": "Pere Manel", "Edad": 60, "Altura": 1.92, "Casado": True}
2   print(dicPersonal["Casado"])
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/Diccionarios.py
True
PS F:\CursoPython>
```

Un diccionario puede tener valores de distinto tipo.

```
1   paises =("España", "Francia", "Inglaterra")
2   dicPaises = {paises[0]: "Madrid", paises[1]: "Paris", paises[2]: "Londres"}
3   print(dicPaises)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/Diccionarios.py
{'España': 'Madrid', 'Francia': 'Paris', 'Inglaterra': 'Londres'}
PS F:\CursoPython>
```

Partiendo de una lista, podemos crear un dicccionario partiendo de la posición del elemento del pais y como valor asignarle la capital.

El resultado final será un diccionario como en los ejemplos anteriores.

```
1   datosPersonales = {"Apellido": "Garcia", "hijos": {1:"Carlos", 2:"Ana", 3:"Pedro"}}
2   print(datosPersonales)
3   print(datosPersonales["hijos"])
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/Diccionarios.py
{'Apellido': 'Garcia', 'hijos': {1: 'Carlos', 2: 'Ana', 3: 'Pedro'}}
{1: 'Carlos', 2: 'Ana', 3: 'Pedro'}
PS F:\CursoPython>
```

La línea 2 muestra como dentro de un diccionario tenemos otro diccionario.

La línea 3 imprimimos los valores del diccionario que se encuentra dentro del diccionario.

```
1   datosPersonales = {"Apellido": "Garcia", "hijos": {1:"Carlos", 2:"Ana", 3:"Pedro"}}
2
3   print(datosPersonales.get("Apellido", "La clave no existe"))
4   print(datosPersonales.get("Nombre", "La clave no existe"))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/Diccionarios.py
Garcia
La clave no existe
PS F:\CursoPython>
```

Con el método get ponemos dos argumentos la clave a buscar y el texto que tiene que mostras cuando dicha clave no existe.

```
1   datosPersonales = {"Apellido": "Garcia", "hijos": {1:"Carlos", 2:"Ana", 3:"Pedro"}}
2   print(datosPersonales.keys())
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/Diccionarios.py
dict_keys(['Apellido', 'hijos'])
PS F:\CursoPython>
```

Con el método .keys() nos muestra solo las claves.

```
1   datosPersonales = {"Apellido": "Garcia", "hijos": {1:"Carlos", 2:"Ana", 3:"Pedro"}}
2   print(datosPersonales.values())
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
```

```
/python.exe f:/CursoPython/Diccionarios.py
dict_values(['Garcia', {1: 'Carlos', 2: 'Ana', 3: 'Pedro'}])
PS F:\CursoPython>
```

Con el método .values() muestra todos los valores asocioados de un diccionario.

```
1   datosPersonales = {"Apellido": "Garcia", "Edad": 60, "Casado": True}
2   valoresLista = list(datosPersonales.keys())
3   valoresTupla = tuple(datosPersonales.values())
4   print(valoresLista)
5   print(valoresTupla)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/Diccionarios.py
['Apellido', 'Edad', 'Casado']
('Garcia', 60, True)
PS F:\CursoPython>
```

Con la función list() podemos convertir un diccionario a lista llamada valoresLista.

Con la función tuple() podemos convertir un diccionario a tupla llamada valoresTupla.

12.- Lectura de Datos por Teclado en Python

```
1    nombre = "Oscar"
2    edad = 25
3
4    print("Hola, " + nombre)
5    print("Tu edad es: ", edad)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/IngresoDatos.py
Hola, Oscar
Tu edad es:  25
PS F:\CursoPython>
```

Inicializamos variables que a continuación mostramos por consola.

Pero si nosotros queremos que estos valores sean dinámicos, tendremos que introducir estos valores por teclado.

```
1    nombre = input("Ingrese su nombre: ")
2    edad = input("Ingrese su edad: ")
3
4    print("Hola, " + nombre)
5    print("Tu edad es: ", edad)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/IngresoDatos.py
Ingrese su nombre: Pere Manel
Ingrese su edad: 60
Hola, Pere Manel
Tu edad es:  60
PS F:\CursoPython>
```

La sentencia input() muestra el mensaje que contiene en sus paréntesis y el programa se detiene en espera que el usuario introduzca sus datos.

Una vez introducido el nombre y la edad muestra su resultado.

```
1    nombre = input("Ingrese su nombre: ")
2    edad = input("Ingrese su edad: ")
3
```

```
4    print("Hola, " + nombre)
5    edadFutura = edad + 20
6    print("Tu edad es: ", edad)
7    print("Tu edad (dentro de 20 años) será: ", edadFutura)
```

Queremos mostrar la edad que tendremos dentro de 20 años, lo que estamos intentando es en la línea 5 definir una variable llamada edadFutura que contenga la edad actual del usuario más 20, y en la línea 7 mostrar el resultado por consola.

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/IngresoDatos.py
Ingrese su nombre: Pere Manel
Ingrese su edad: 60
Hola, Pere Manel
Traceback (most recent call last):
  File "f:\CursoPython\IngresoDatos.py", line 5, in <module>
    edadFutura = edad + 20
                 ~~~~~^~~~
TypeError: can only concatenate str (not "int") to str
PS F:\CursoPython>
```

Nos está dando un error en la línea 5, nos dice que no podemos sumar edad + 20 para asignárselo a la variable edadFutura.

El motivo es que la sentencia input() siempre nos va a retornar un valor string (texto) por este motivo no puede realizar la operación.

En capítulos anteriores hablamos sobre el Casting, consistía en reconvertir una variable de un tipo a otro, para este caso vamos a utilizar int() para convertir el resultado en un valor entero y asignárselo a edadFutura.

```
1    nombre = input("Ingrese su nombre: ")
2    edad = int(input("Ingrese su edad: "))
3
4    print("Hola, " + nombre)
5    edadFutura = edad + 20
6    print("Tu edad es: ", edad)
7    print("Tu edad (dentro de 20 años) será: ", edadFutura)
```

Ahora en la línea 2 le estamos diciendo que el valor obtenido por el input() nos lo retorne como un valor entero.

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL
```

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/IngresoDatos.py
Ingrese su nombre: Pere Manel
Ingrese su edad: 60
Hola, Pere Manel
Tu edad es:  60
Tu edad (dentro de 20 años) será:  80
PS F:\CursoPython>
```

```
1   nombre = input("Ingrese su nombre: ")
2   edad = int(input("Ingrese su edad: "))
3   sueldo = float(input("Ingrese su sueldo anual: "))
4
5   print("Hola, " + nombre)
6   edadFutura = edad + 20
7   print("Tu edad es: ", edad)
8   print("Tu edad (dentro de 20 años) será: ", edadFutura)
9   print("Su suedo en 12 pagas será ", sueldo/12 , " mensual")
```

En la línea 3 hacemos una reconversión a float (número decimal).

Este será el resultado:

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/IngresoDatos.py
Ingrese su nombre: Pere Manel
Ingrese su edad: 60
Ingrese su sueldo anual: 35000
Hola, Pere Manel
Tu edad es:  60
Tu edad (dentro de 20 años) será:  80
Su suedo en 12 pagas será  2916.6666666666665  mensual
PS F:\CursoPython>
```

13.- Estructura Condicional IF, ELSE, ELIF en Python

```
1   edad = int(input("Ingrese su edad: "))
2
3   if edad >= 18:
4       print("Eres mayor de edad.")
```

En la línea 3 le estamos diciendo Sí edad es mayor o igual a 18 seguido de dos puntos (:), si eso se cumple que ejecute la línea número 4.

Como podrás observar la línea 4 el codigo está tabulado hacia la derecha, esto se denomina identación significa que todo lo que se encuentre indentado pertenece a la condición, cuando en la siguiente línea dejamos el identado, significa que ya no pertenece a la condición.

Si ejecutamos este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/If_Else.py
Ingrese su edad: 60
Eres mayor de edad.
PS F:\CursoPython>
```

En este ejemplo como 60 en mayor o igual a 18 nos dirá “Eres mayor de edad”.

Si ejecutamos de nuevo y le decimos que tenemos 12 años que pasará:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/If_Else.py
Ingrese su edad: 12
PS F:\CursoPython>
```

La línea 4 al no cumplir la condición no se ejecutará, por este motivo no lo verás en consola.

```
1   edad = int(input("Ingrese su edad: "))
2
3   if edad >= 18:
4       print("Eres mayor de edad.")
5   else:
6       print("No eres mayor de edad.")
```

Con la línea 5 else Si no ejecuta la línea 6.

Ahora tanto si introducimos una edad mayor o igual a 18 o menor vamos a obtener una respuesta.

Vamos a ejecutar contestando como antes 12 años.

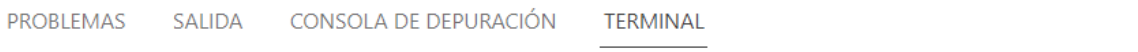

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/If_Else.py
Ingrese su edad: 12
No eres mayor de edad.
PS F:\CursoPython>
```

```
1   edad = int(input("Ingrese su edad: "))
2
3   if edad > 18:
4       print("Eres mayor de edad.")
5   elif edad == 18:
6       print("¡Tienes 18 años!")
7   else:
8       print("No eres mayor de edad.")
```

En la línea 3 si se cumple la condición de que edad es mayor a 18 se ejecutará la línea 4 e ignorará el resto del código.

En la línea 5 si la condición anterior no se cumple comprobará si edad es igual a 18, con lo que se ejecutará la línea 6, de no cumplirse irá a ejecutar la línea 8.

Este será el resultado:




```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/If_Else.py
Ingrese su edad: 18
¡Tienes 18 años!
PS F:\CursoPython>
```

Podemos introducir tanos elif como sean necesarios, en cambio el if y el else solo puede contener uno.

14.- Funciones en Python

Función: Es un conjunto de instrucciones que realizan un proceso o tarea.

Su principal ventaja es que nos ayuda a evitar código repetido.

```
1    print("García")
2    print("Oscar")
3    print("UskoKruM2010")
4
5    print("García")
6    print("Oscar")
7    print("UskoKruM2010")
8
9    print("García")
10   print("Oscar")
11   print("UskoKruM2010")
12
13   print("García")
14   print("Oscar")
15   print("UskoKruM2010")
16
17   print("García")
18   print("Oscar")
19   print("UskoKruM2010")
```

Supongamos que queremos repetir una tarea 5 veces, en principio funcionaría pero hacemos muchas repeticiones de código y no es eficiente.

Pero si lo ejecutamos este funciona correctamente.

Para este ejemplo podemos realizar una función que agrupe este código que estamos repitiendo tantas veces.

```
1    def saludar():
2        print("García")
3        print("Oscar")
4        print("UskoKruM2010")
5
```

```
6    saludar()
7    saludar()
8    saludar()
9    saludar()
10   saludar()
```

En este ejemplo hemos creado la función saludar con la palabra reservada def.

Desde la línea 6 has la línea 10 llamamos a la función 5 veces.

Si la ejecutamos este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/Funciones.py
García
Oscar
UskoKruM2010
García
Oscar
UskoKruM2010
García
Oscar
UskoKruM2010
García
Oscar
UskoKruM2010
García
Oscar
UskoKruM2010
PS F:\CursoPython>
```

Podemos observar que se ejecuta 5 veces.

Esta función también nos permite retornar valores:

```
1    def saludar():
2        print("García")
3        print("Oscar")
4        print("UskoKruM2010")
5        return "Hola"
6
```

```
 7    print(saludar())
 8    print(saludar())
 9    print(saludar())
10    print(saludar())
11    print(saludar())
```

Además a esta función le hemos añadido la palabra return, esto hace que además de ejecutarse nos retorne la palabra “Hola” para ello tenemos que llamar a la función pero le tenemos que decir ¿Cómo queremos que nos retorne el valor? En este ejemplo será por mediación de un print().

Si ejecutamos este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/Funciones.py
García
Oscar
UskoKruM2010
Hola
García
Oscar
UskoKruM2010
Hola
García
Oscar
UskoKruM2010
Hola
García
Oscar
UskoKruM2010
Hola
García
Oscar
UskoKruM2010
Hola
PS F:\CursoPython>
```

Además de ejecutar la función nos devuelve la palabra “Hola”.

Ahora vamos a crear una función a la que le vamos a pasar un parámetro.

```
1    def evaluarSueldoMinimo(sueldo):
2        if sueldo >= 1000:
3            print("Cumples con el sueldo mínimo")
4        else:
5            print("No cumples con el sueldo mínimo")
```

```
6
7
8   evaluarSueldoMinimo(1200)
9
10  evaluarSueldoMinimo(800)
```

Hemos creado una función llamada evaluarSueldoMinimo(Con el parámetro sueldo), además con un condicional comprobamos si cumple o no con el sueldo mínimo.

Cuando tengamos que llamar a esta función no pedirá un valor, ya que este lo espera, la hemos llamado dos veces que con valores distintos, si ejecutamos este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/Funciones.py
Cumples con el sueldo mínimo
No cumples con el sueldo mínimo
PS F:\CursoPython>
```

En la primera función cumple con el sueldo mínimo pero la segunda no.

```
8   evaluarSueldoMinimo()
```

Si la llamamos sin ningún argumento nos genera un mensaje de error.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Funciones.py
Traceback (most recent call last):
  File "f:\CursoPython\Funciones.py", line 8, in <module>
    evaluarSueldoMinimo()
TypeError: evaluarSueldoMinimo() missing 1 required positional argument:
'sueldo'
PS F:\CursoPython>
```

Nos dirá que la función requiere de un argumento llamado ‘sueldo’.

A las funciones se les pueden agregar varios parámetros.

15.- Operadores Lógicos (AND – OR – NOT) en Python

AND: Equivalente a "Y si además…"

OR: "O sino …"

Not: Negación

```
1   distancia = 1200
2   numeroHermanos = 3
3   salarioPadres = 1500
4
5   tieneBeneficio = False
6
7   if(distancia > 1000 and numeroHermanos > 2) or salarioPadres < 2000:
8       tieneBeneficio = True
9
10  print(tieneBeneficio)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/OperadoresLogicos.py
True
PS F:\CursoPython>
```

En la línea 7 estamos comparando si la distancia es mayor de mil y además numeroHermanos mayor de 2, al tener el operador and para que el resultado sea True se ha de cumplir las dos condiciones.

Pero además con el operador or estamos comparando que salarioPadres sea menor de 2000.

En este ejemplo el resultado será True sí una de las condiciones separada por or se cumple o bien las dos condiciones separadas por and se cumplen las dos.

Cambia los valores de las variables para ver qué resultado se obtiene.

1.- Distancia = 800, numeroHermanos = 3, SalarioPadres = 1500 → True

2.- Distancia = 800, numeroHermanos = 2, SalarioPadres = 1500 → True

3.- Distancia = 1200, numeroHermanos = 2, SalarlioPadres = 2000 → False

Si en la línea 10 agregamos el operador not

```
10  print(not(tieneBeneficio))
```

Y vuelves a realizar las pruebas anteriores podrás observar que el resultado será todo lo contrario.

Vamos a ver un par de ejemplos de más.

```python
# Ha de cumplir las dos condiciones para que sea verdad
if (5 > 3) and (8 < 10):
    print("Verdad")
else:
    print("Es mentira...")

# Con solo cumplir una de las condiciones para que sea verdad
if (5 > 3) or (8 < 6):  #
    print("Verdad")
else:
    print("Es mentira...")
```

16.- Operador Ternario en Python

```
"""
  String sexo;
  sexo = (10 > 20) ? "Masculino" : "Femenino";
  """

  sexos = ("Masculino", "Femenino")
  sexo = 10 > 5
  print(sexos[not sexo])
  sexo = 10 > 20
  print(sexos[not sexo])
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/OperadorTernario.py
Masculino
Femenino
PS F:\CursoPython>
```

Operador ternario en Python no existe pero podemos simular con el ejercicio anterior. Es un algoritmo anternativo.

```
# Creamos una tupla
sexos = ("Hombre", "Mujer")
posicion = True
sexo = sexos[1] # Mujer
print(sexo)
sexo = sexos[0] # Hombre
print(sexo)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/OperadorTernario.py
Mujer
Hombre
PS F:\CursoPython>
```

Vamos a cambiar el código:

```
# Creamos una tupla
sexos = ("Hombre", "Mujer")
posicion = True
sexo = sexos[posicion] # Mujer
print(sexo)
sexo = sexos[not posicion] # Hombre
print(sexo)
```

Este será el resultado:

Observamos que el resultado es el mismo.

En otros lenguajes tenemos esta solución:

17.- IF con Tuplas & Listas (IF – IN)

```
1   cursos = ("Matematicas", "Biologia", "Lenguaje", "Ciencias")
2   print("--- Cursos ---")
3   print("Matematicas - Biologia - Lenguaje - Ciencias")
4
5   curso = input("Ingresa el curso deseado: ")
6
7   if curso in cursos:
8       print("Curso {0} seleccionado.".format(curso))
9   else:
10      print("No existe este curso...")
```

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/If_In.py
--- Cursos ---
Matematicas - Biologia - Lenguaje - Ciencias
Ingresa el curso deseado: Lenguaje
Curso Lenguaje seleccionado.
PS F:\CursoPython>
```

Si introducimos un curso que no existe:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/If_In.py
--- Cursos ---
Matematicas - Biologia - Lenguaje - Ciencias
Ingresa el curso deseado: Informatica
No existe este curso...
PS F:\CursoPython>
```

18.- Función Range en Python

Range(): Crear una lista inmutable de números enteros en sucesión aritmética.

```
1    numeros = range(5)
2    print(numeros)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Range.py
range(0, 5)
PS F:\CursoPython>
```

Vamos a imprimir los valores por separado:

```
4    print(numeros[0])
5    print(numeros[1])
6    print(numeros[2])
7    print(numeros[3])
8    print(numeros[4])
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Range.py
range(0, 5)
0
1
2
3
4
PS F:\CursoPython>
```

Si intentamos imprimir un valor de un elemento que no existe:

```
9    print(numeros[5])
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

11/python.exe f:/CursoPython/Range.py
range(0, 5)
0
1
```

```
2
3
4
Traceback (most recent call last):
  File "f:\CursoPython\Range.py", line 9, in <module>
    print(numeros[5])
          ~~~~~~~^^^
IndexError: range object index out of range
PS F:\CursoPython>
```

```
1    numeros = range(4,10)
2    print(numeros[3])
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Range.py
7
PS F:\CursoPython>
```

	números					
Índice	0	1	2	3	4	5
Valor	4	5	6	7	8	9

```
1    numeros = range(10,100,8)
2    print(numeros[9])
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Range.py
82
PS F:\CursoPython>
```

números											
Ind.	0	1	2	3	4	5	6	7	8	9	10	11
Val	10	18	26	34	42	50	58	66	74	82	90	98

Podemos pasar estos valores a una lista:

```
1    numeros = range(10,100,8)
2
3    valores = list(numeros)
4    print(valores)
```

Este será el resultado:

19.- Bucle For en Python

Bucles: Son estrucutras de control de flujo que repiten 1 o varias líneas de codigo.

```
1    for num in range(0,10):
2        print("Valor actual: {0}".format(num))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/For.py
Valor actual: 0
Valor actual: 1
Valor actual: 2
Valor actual: 3
Valor actual: 4
Valor actual: 5
Valor actual: 6
Valor actual: 7
Valor actual: 8
Valor actual: 9
PS F:\CursoPython>
```

El bucle en cada iteración la variable num asume un valor que empieza por 0 y termina en el último valor menos 1, se repite 10 veces.

```
1    for num in range(0,20, 2):
2        print("Valor actual: {0}".format(num))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/For.py
Valor actual: 0
Valor actual: 2
Valor actual: 4
Valor actual: 6
Valor actual: 8
Valor actual: 10
Valor actual: 12
Valor actual: 14
Valor actual: 16
Valor actual: 18
PS F:\CursoPython>
```

En este ejemplo el valor num empieza por 0, tiene que llegar al 20 pero con un incremento de 2, como después del 18 viene el 20 este ya no se muestra.

Vamos a ver otro ejemplo:

```
for i in range(1, 13):
    print("{0} x {1} es {2}".format(i, i, (i*i)))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/For.py
1 x 1 es 1
2 x 2 es 4
3 x 3 es 9
4 x 4 es 16
5 x 5 es 25
6 x 6 es 36
7 x 7 es 49
8 x 8 es 64
9 x 9 es 81
10 x 10 es 100
11 x 11 es 121
12 x 12 es 144
PS F:\CursoPython>
```

Veremos como unas tablas de multiplicar.

```
tupla = ("Karen", "Oscar", "Héctor", "Leonardo")
for nom in tupla:
    print("Cantidad de letras de {0} es: {1}".format(nom, len(nom)))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/For.py
Cantidad de letras de Karen es: 5
Cantidad de letras de Oscar es: 5
Cantidad de letras de Héctor es: 6
Cantidad de letras de Leonardo es: 8
PS F:\CursoPython>
```

20.- Factorial de un número con Python

Factorial: Es el producto de todos los números positivos enteros comprendidos entre 1 y un determinado número.

Por ejemplo:

El Factorial de 5 que también se representa 5! Será 1 * 2 * 3 * 4 * 5 = 120

El Factorial de 4 que también se representa 4! Será 1 * 2 * 3 * 4 = 24

```
1   numero = int(input("Ingrese un número: "))
2   factorial = 1
3   for num in range(1, (numero+1)):
4       factorial = factorial * num
5
6   print("El factorial {0}! es igual a {1}".format(numero, factorial))
```

Probamos con el número 4 y 5.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Factorial.py
Ingrese un número: 4
El factorial 4! es igual a 24
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Factorial.py
Ingrese un número: 5
El factorial 5! es igual a 120
PS F:\CursoPython>
```

En la línea 1 asignamos un valor numérico a la variable numero de los que el usuario introduzca por teclado, con int lo reconvertimos a entero, para poder operar.

En la línea 2 asignamos a la variable factorial el valor 1, esta variable se tiene que multiplicar por todos los números que recorren el factorial, si esta tuviera el valor de 0, todo lo que multiplicara sería igual a 0.

En la línea 3 tenemos que hacer un recorrido desde 1 hasta el número que ha introducido el usuario, como el ciclo for omite el último número hemos puesto (numero+1).

En la línea 4 la iteración pasa por que num asume los valores 1, 2, 3 … hasta el valor introducido por el usuario, que lo multiplica con la variable factorial a su vez lo acumula.

En la línea 6 una vez fuera del bucle imprimimos por consola el resultado del número y su factorial.

21.- Bucle While en Python

While: Estructura repetitiva que nos permite realizar múltiples iteraciones, basándonos en el resultado de una expresión lógica que puede tener como resultado un valor True o False.

```
1    indice = 1
2
3    while indice < 10:
4        print("Valor acutal {0}".format(indice))
```

Si ejecutamos el programa tal como se muestra, vamos a entrar en un bucle infinito porque la variable índice siempre será menos a 10

Tenemos que agregar una instrucción que haga que esto cambie.

```
1    indice = 1
2
3    while indice < 10:
4        print("Valor acutal {0}".format(indice))
5        indice = indice + 1
```

En cada iteración la variable índice se incremente en 1, este pasará a tener los valores de 2, 3, 4, … y así sucesivamente has que índice tenga el valor de 10, cuando lleguemos a este punto la condición del while dejara de cumplirse, es cuando se terminará el bucle.

```
1   indice = 1
2
3   while indice < 10:
4       print("Valor acutal {0}".format(indice))
5       indice = indice + 1
6
7   print("Hemos terminado el bucle while ya que indice vale {0}".format(indice))
```

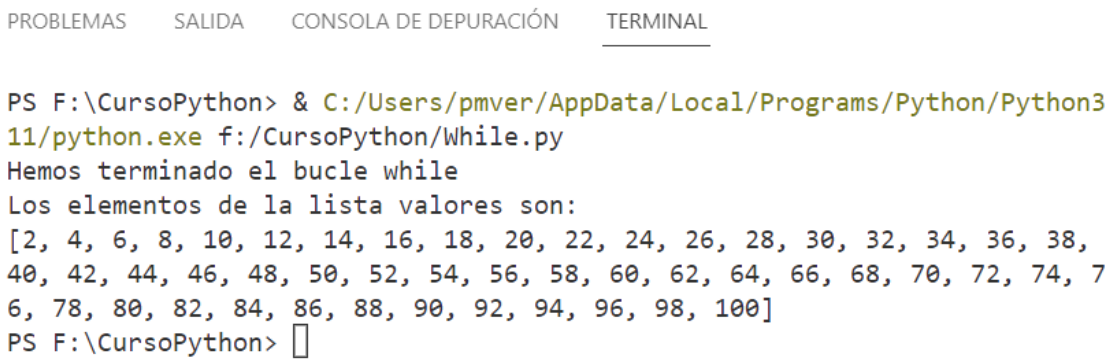
Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/While.py
Valor acutal 1
Valor acutal 2
Valor acutal 3
Valor acutal 4
Valor acutal 5
Valor acutal 6
Valor acutal 7
Valor acutal 8
Valor acutal 9
Hemos terminado el bucle while ya que indice vale 10
PS F:\CursoPython>
```

Vamos a ver otro ejemplo:

```
1   valores = []
2   inicio = 2
3   while inicio <= 100:
4       valores.append(inicio)
5       inicio += 2  # Es igual que inicio = inicio + 2
6
7   print("Hemos terminado el bucle while")
8   print("Los elementos de la lista valores son:")
9   print(valores)
```

Este será el resultado:



Mientras la condición inicio <= 100, los valores se irán añadiendo a la lista valores.

Como empieza por 0 al incrementar por 2 en cada bucle, solo se agregarán los números pares hasta que llegue al valor de 100.

22.- Sentencias Break, Continue, Pass en Python

Break: Permite salir de un bucle sin que este haya terminado, cuando se cumpla una condición.

```
1   for numero in range(1,6):
2       print("El numero es {0}".format(numero))
3
4   print("Bucle terminado")
```

Con este ejemplo el resultado será:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Break_Continue_pass.py
El numero es 1
El numero es 2
El numero es 3
El numero es 4
El numero es 5
Bucle terminado
PS F:\CursoPython>
```

```
1   for numero in range(1,6):
2       if numero == 3:
3           break
4       print("El numero es {0}".format(numero))
5
6   print("Bucle terminado")
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Break_Continue_pass.py
El numero es 1
El numero es 2
Bucle terminado
PS F:\CursoPython>
```

Cuando la variable numero sea igual a 3 que haga en break (rotura) se interrumpe el bucle para salir del bucle.

Continue: Omite una parte del bucle cuando se cumple una condición y continúa con el resto.

```python
for numero in range(1,6):
    if numero == 3:
        continue
    print("El numero es {0}".format(numero))

print("Bucle terminado")
```

Este será el resultado:

Cuando la variable numero es igual a 3 se salta esta iteración para seguir con la siguiente.

Podrás observa que cuando numero tiene el valor 3 no se imprime.

Pass: Permite continuar con una sentencia o función que ya no tiene o aun no tiene un bloque de código útil.

```python
for numero in range(1,6):
    if numero <= 3:
        # Aquí no pasa nada y el bucle sigue trabajando.
        pass
    print("El numero es {0}".format(numero))

print("Bucle terminado")
```

Este será el resultado:

```
PS F:\CursoPython>
```

```python
for numero in range(1,6):
    if numero <= 3:
        # Aquí no pasa nada y el bucle sigue trabajando.
        pass
    else:
        print("El siguiente valor es mayor a 3:")

    print("El numero es {0}".format(numero))

print("Bucle terminado")
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Break_Continue_pass.py
El numero es 1
El numero es 2
El numero es 3
El siguiente valor es mayor a 3:
El numero es 4
El siguiente valor es mayor a 3:
El numero es 5
Bucle terminado
PS F:\CursoPython>
```

En este ejemplo mientras numero sea menos o igual a 3 que no haga nada, de lo contrario que imprima un mensaje “El siguiente valor es mayor a 3:”.

```python
def funcionSinImplemenar():
    pass
```

Cuando definimos una función, pero aún no tenemos claro como la vamos a programar, al introducir la sentencia pass evitamos que nuestro programa genere un error e interrumpiendo el programa.

23.- Generadores en Python I

Generadores: Es un concepto que nos permite extraer valores de una función y almacenarlo de (de uno en uno) en objetos iterables (que se pueden recorrer), si la necesidad de almacenar TODOS A LA VEZ en la memoria RAM de nuestro ordenador.

```
1   def generaMultiplos7(limite):
2       numero = 1
3       listaNumeros=[]
4
5       while numero<=limite:
6           listaNumeros.append(numero*7)
7           numero = numero + 1
8       return listaNumeros # REtornamos toda la lista creada
9
10  print(generaMultiplos7(10))
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Generadores.py
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
PS F:\CursoPython>
```

Retorna 10 números, múltiplos de 7.

Vamos a ver otro ejemplo:

```
1   def generaMultiplos7(limite):
2       numero = 1
3       listaNumeros=[]
4
5       while numero<=limite:
6           listaNumeros.append(numero*7)
7           yield numero * 7 # yield: Genera un objeto iterable.
8           numero = numero + 1
9       return listaNumeros
10
11  obtieneMultiplos7 = generaMultiplos7(10)
12
13  print(obtieneMultiplos7)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL
```

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Generadores.py
<generator object generaMultiplos7 at 0x0000020FA498D0E0>
PS F:\CursoPython>
```

Obtiene un genedador de generarMultiplos7 pero no puede mostrar los elementos como en el ejemplo anterior.

```python
def generaMultiplos7(limite):
    numero = 1
    listaNumeros=[]

    while numero<=limite:
        listaNumeros.append(numero*7)
        yield numero * 7 # yield: Genera un objeto iterable.
        numero = numero + 1

obtieneMultiplos7 = generaMultiplos7(10)

for num in obtieneMultiplos7:
    print(num)
```

Este será el resultado:

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Generadores.py
7
14
21
28
35
42
49
56
63
70
PS F:\CursoPython>
```

Un generador nos permite recorrer e imprimir valor 1 por 1.

La función next() nos permite ir recorriendo valor por valor.

```python
def generaMultiplos7(limite):
    numero = 1
    listaNumeros=[]

    while numero<=limite:
        listaNumeros.append(numero*7)
```

```
7           yield numero * 7 # yield: Genera un objeto iterable.
8           numero = numero + 1
9
10  obtieneMultiplos7 = generaMultiplos7(10)
11
12  print(next(obtieneMultiplos7))
13  print("300 líneas después...")
14  print(next(obtieneMultiplos7))
15  print("Mil líneas después...")
16  print(next(obtieneMultiplos7))
```

Este sería el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/Generadores.py
7
300 líneas después...
14
Mil líneas después...
21
PS F:\CursoPython>
```

No tengo que tener todos los datos almacenados pero si acceder a cada uno de los elementos cuando sea necesario.

Los generadores son más eficientes que las funciones tradicionales.

Muy útiles con listas de valores infinitos, porque no sabemos hasta que elementos de esta lista vamos a usar.

Entre llamada y llamada, el objeto iterable entra en un estado de pausa (suspensión).

24.- Generadores en Python II

Cuando indicamos un * adelante del parámetro de una función, estamos indicando que se va a recibir un número indeterminado de parámetros o elementos. Además, esos parámetros se recibirán en forma de tupla.

```
1   # Como parámetro se va a recibir una cantidad variable
2   def devuelveLenguajes(*lenguajes):
3       for leng in lenguajes: # La función leng nos permite
4           yield leng             # generar el generador.
5
6   lenguajesObtenidos=devuelveLenguajes("Python", "Java", "PHP", "Ruby", "JavaScript")
7
8   print(next(lenguajesObtenidos))
9   print(next(lenguajesObtenidos))
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/Generadores2.py
Python
Java
PS F:\CursoPython>
```

Como puedo hacer para que me retorne letra a letra en lugar de lenguaje a lenguaje.

```
1   # Como parámetro se va a recibir una cantidad variable
2   def devuelveLenguajes(*lenguajes):
3       for leng in lenguajes: # La función leng nos permite
4           for letra in leng:
5               yield letra
6
7
8   lenguajesObtenidos=devuelveLenguajes("Python", "Java", "PHP", "Ruby", "JavaScript")
9
10  print(next(lenguajesObtenidos))
11  print(next(lenguajesObtenidos))
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/Generadores2.py
P
y
PS F:\CursoPython>
```

Nos va retornando letra por letra los lenguajes de programación.

Cuando utilizamos un for dentro de otro for, se conoce con un for anidado.

Ahora vamos a ver como se puede realizar de otra forma:

```python
# Como parámetro se va a recibir una cantidad variable
def devuelveLenguajes(*lenguajes):
    for leng in lenguajes:
        yield from leng

lenguajesObtenidos=devuelveLenguajes("Python", "Java", "PHP", "Ruby", "JavaScript")

print(next(lenguajesObtenidos))
print(next(lenguajesObtenidos))
```

Este será el resultado:

La instrucción from nos permite realizar un elemento iterable de otro elemento iterable.

25.- Excepciones en Python. Bloque TRY EXCEPT FINALLY

Excepción: Error en tiempo de ejecución (duranta la ejecución de un programa).

```
1   numero1 = 20
2   numero2 = 0
3
4   print("La división entre {0} entre {1} es: {2}".format(numero1, numero2, (numero1/numero2)))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/Excepciones.py
Traceback (most recent call last):
  File "f:\CursoPython\Excepciones.py", line 4, in <module>
    print("La división entre {0} entre {1} es: {2}".format(numero1, numero2,
 (numero1/numero2)))

   ~~~~~~~^~~~~~~~
ZeroDivisionError: division by zero
PS F:\CursoPython>
```

No se puede dividir un valor entre 0, genera un error ZeroDivisionError: división by zero.

No hay ningún error de sintaxis, el error se genera cuando intenta dividir un valor entre 0, esto es una excepción.

```
1   numero1 = 20
2   numero2 = 0
3
4   print("La división entre {0} entre {1} es: {2}".format(numero1, numero2, (numero1/numero2)))
5
6   print("Aquí termina mi programa")
```

Vamos a ejecutar de nuevo.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/Excepciones.py
Traceback (most recent call last):
  File "f:\CursoPython\Excepciones.py", line 4, in <module>
    print("La división entre {0} entre {1} es: {2}".format(numero1, numero2,
 (numero1/numero2)))

   ~~~~~~~^~~~~~~~
ZeroDivisionError: division by zero
PS F:\CursoPython>
```

Al haber un error el programa se interrumpe y no finaliza con todas sus instrucciones, en este ejemplo no se ejecuta la línea 6.

```
1   numero1 = 20
2   numero2 = 0
3   try:
4       print("La división entre {0} entre {1} es: {2}".format(numero1, numero2, (numero1/numero2)))
```

```
5  except:
6      print("Error en el programa.")
7
8  print("Aquí termina mi programa")
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe
 f:/CursoPython/Excepciones.py
Error en el programa.
Aquí termina mi programa
PS F:\CursoPython>
```

Nos muestra un mensaje “Error en el programa” este mensaje de la línea 6 se ejecuta cuando en el try: se ha producido un error.

Pero el programa no se detiene y continua con su ejecución, en la línea 8 muestra el mensaje “Aquí termina mi programa”.

```
1  numero1 = 20
2  numero2 = 0
3  try:
4      print("La división entre {0} entre {1} es: {2}".format(numero1, numero2, (numero1/numero2)))
5  except ZeroDivisionError:
6      print("No se puede dividir entre 0.")
7
8  print("Aquí termina mi programa")
```

Al poner except ZeroDivisionError, estamos controlando este tipo de error, de este modo podemos personalizar los distintos tipos de errores que pueden suceder.

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe
 f:/CursoPython/Excepciones.py
No se puede dividir entre 0.
Aquí termina mi programa
PS F:\CursoPython>
```

```
1  numero1 = 20
2  numero2 = 0
3  try:
4      print("La división entre {0} entre {1} es: {2}".format(numero1, numero2, (numero1/numero2)))
5  except ZeroDivisionError:
6      print("No se puede dividir entre 0.")
7  finally:
8      print("Yo siempre aparzco")
9
10 print("Aquí termina mi programa")
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe
 f:/CursoPython/Excepciones.py
No se puede dividir entre 0.
```

```
Yo siempre aparzco
Aquí termina mi programa
PS F:\CursoPython>
```

Tanto si se produce un error como si no siempre eje el contenido de finally:

Vamos a cambiar la variable numero2 por el valor de 2.

Vamos a ejecutar de nuevo.

Como no se produce ningún error, se ejecuta la línea 4, no se ejecuta la línea 6 y la línea 8 si se ejecuta, por finally siempre se ejecutará.

26.- Sentencia RAISE (Lanzamiento de Excepciones)

Raise: Sirve para lanzar (de forma intencional) excepciones en Python.

```python
def evaluarNota(nota):
    if nota < 0:
        print("Nota no válida.")
    elif nota >= 16:
        print("Excelente")
    elif nota >= 11:
        print("Aprobado")
    else:
        print("Suspendido")

evaluarNota(13)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe
 f:/CursoPython/Raise.py
Aprobado
PS F:\CursoPython>
```

```python
def evaluarNota(nota):
    if nota < 0:
        raise ZeroDivisionError("No se permiten valores negativos...")
    elif nota >= 16:
        print("Excelente")
    elif nota >= 11:
        print("Aprobado")
    else:
        print("Suspendido")

evaluarNota(-1)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe
 f:/CursoPython/Raise.py
Traceback (most recent call last):
  File "f:\CursoPython\Raise.py", line 11, in <module>
    evaluarNota(-1)
    ^^^^^^^^^^^^^^^
```

```
  File "f:\CursoPython\Raise.py", line 3, in evaluarNota
    raise ZeroDivisionError("No se permiten valores negativos...")
ZeroDivisionError: No se permiten valores negativos...
PS F:\CursoPython>
```

```python
def evaluarNota(nota):
    if nota < 0:
        raise ZeroDivisionError("No se permiten valores negativos...")
    elif nota >= 16:
        print("Excelente")
    elif nota >= 11:
        print("Aprobado")
    else:
        print("Suspendido")

evaluarNota(-1)

print("Este es el fin de mi programa.")
```

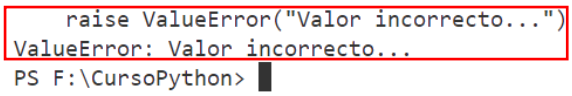
En este caso la línea 13 no se ejecuta.

Otro ejemplo:

```python
def evaluarNota(nota):
    if nota < 0:
        raise ValueError("Valor incorrecto...")
    elif nota >= 16:
        print("Excelente")
    elif nota >= 11:
        print("Aprobado")
    else:
        print("Suspendido")

evaluarNota(-7)

print("Este es el fin de mi programa.")
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe
 f:/CursoPython/Raise.py
Traceback (most recent call last):
  File "f:\CursoPython\Raise.py", line 11, in <module>
    evaluarNota(-7)
    ^^^^^^^^^^^^^^^
  File "f:\CursoPython\Raise.py", line 3, in evaluarNota
```

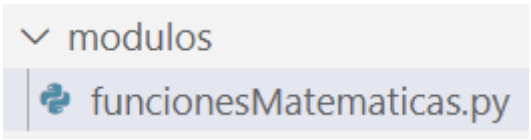


Resumiendo con raise podemos mandar excepciones de forma voluntaria.

27.- Módulos de Python. Importación de archivos externos

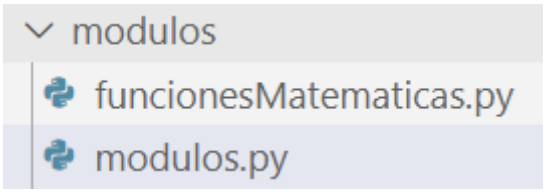
Para este capítulo vamos a crear una nueva carpeta llamada módulos.

En la carpeta vamos a crear un archivo llamado funcionesMatematicas.py.




```
1   def sumar(n1, n2):
2       return n1 + n2
3
4   def multiplicar(n1, n2):
5       return n1 * n2
```

Ahora en la misma carpeta vamos a crear un archivo llamado modulos.py



Un módulo: Es un archivo con extensión .py de Python o -pyc (Python compilado), que posee su propio espacio de nombres y que puede contener variables, funciones, clases o incluso módulos.

¿Para qué sirven los módulos? Sirve para organizar mejor el código y poder reutilizarlo mejor.

Modularización y reutilización.

```
1   import funcionesMatematicas
2
3   print(funcionesMatematicas.sumar(5, 6))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/pytho
n.exe f:/CursoPython/modulos/modulos.py
11
PS F:\CursoPython>
```

Ahora vamos a utilizar la función multiplicar.

```
1   import funcionesMatematicas
2
3   print(funcionesMatematicas.sumar(5, 6))
```

```
4    print(funcionesMatematicas.multiplicar(5, 6))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/pytho
n.exe f:/CursoPython/modulos/modulos.py
11
30
PS F:\CursoPython>
```

```
1    from funcionesMatematicas import sumar, multiplicar
2
3    print(sumar(5, 6))
4    print(multiplicar(5, 6))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/modulos/modulos.py
11
30
PS F:\CursoPython>
```

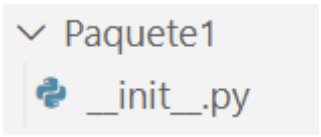
```
1    from funcionesMatematicas import *
2
3    print(sumar(5, 6))
4    print(multiplicar(5, 6))
```

Con el asterisco importamos todas las funciones, este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/modulos/modulos.py
11
30
PS F:\CursoPython>
```

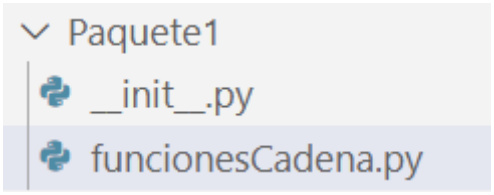
28.- Paquetes de Python. Archivo __init__.py

Para este capítulo vamos a crear una carpeta llamada Paquete1 y dentro un archivo llamado __init__.py.



La función de un archivo __init_.py es convertir mi cable al interprete de Python que convierta un directorio o carpeta simple en un paquete que contiene que contiene de echo otros paquetes o módulos y esto lo hace para poder importarlos de una mejor manera.

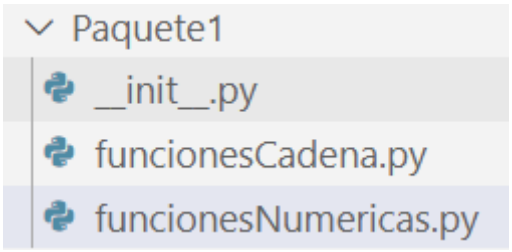
Vamos a crear otro fichero llamado funcionesCadena.py



Vamos a escribir:

```
1  def contarLetras(texto):
2      return len(texto)
```

Dentro de Paquete1 vamos a crear otro archivo llamado funcionesNumericas.py

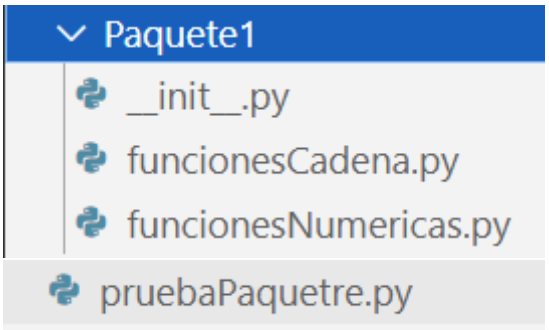


Vamos a escribir:

```
1  def multiplicar(n1,n2):
2      return n1*n2
3
4  def potenciar(base,exponente):
5      return base ** exponente
```

Si abrimos el archivo __init__.py podréis observar que está vacío, su principal funcionalidad es indicarle al interprete de Python que la carpeta Paquete1 se va a tratar como un paquete de Python.

A continuación fuera de la carpeta Paqauete1 vamos a crear un nuevo archivo llamado pruebaPaquete.py.



Paquetes: Directorios (carpetas) donde se almacenan módulos relacionados entre sí.

¿Para qué sirven?

Para organizar mejor el código y poder reutilizarlo mejor (modularización y reutilización).

¿Cómo se crea un paquete?

Crear una carpeta o directorio con un archivo dentro llamado __init__.py.

Lo que hace __init__.py es "convertir" un directorio en un modulo (paquete) que contiene otros módulos, y estos lo hace para poder importarlos.

```
1  from Paquete1.funcionesCadena import contarLetras
2  from Paquete1.funcionesNumericas import *
3
4  print(multiplicar(5,6))
5
6  print(contarLetras("UskoKruM2010"))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/pruebaPaquetre.py
30
12
PS F:\CursoPython>
```

29.- Programación Orientada a Objetos (POO) en Python: Clases y Objetos

¿En qué consiste la Programación Orientada a Objetos?

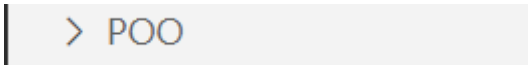
- En trasladar la naturaleza de los objetos de la vida real a código de programación (en algún lenguaje de programación, como Python).

Los objetos de la realidad tienen características (atributos y propiedades) y funcionalidades o comportamientos (funciones o métodos).

Ventajas:

- Modularización (dividir en pequeñas partes) de un programa complejo.
- Código fuente muy reutilizable.
- Código fuente más fácil de incrementar en el futuro y mantener.
- Si existe un fallo en una pequeña parte del código el programa completo no debe fallar necesariamente. Además, es más fácil de corregir esos fallos.
- Encapsulamiento: Ocultación del funcionamiento interno de un objeto.

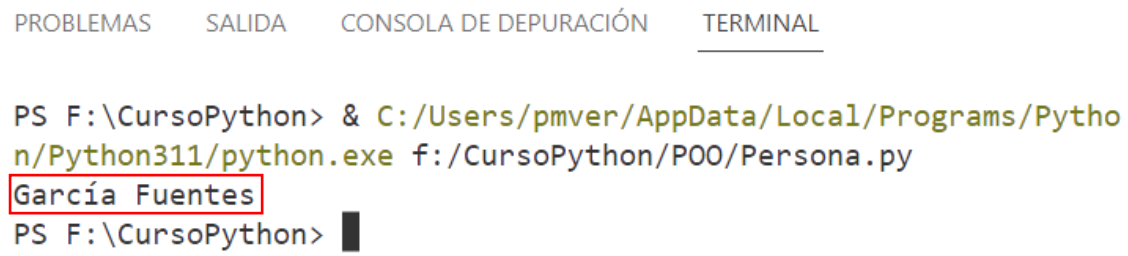
Para este capítulo vamos a crear una nueva carpeta llamada POO.



Dentro de la carpeta un archivo llamado Persona.py


```python
class Persona():
    # Propiedades, características o atributos:
    apellidos = ""
    nombre = ""
    edad = 0
    despierta = False

    # Funcionalidades:
    def despertar(self):
        pass

persona1 = Persona()
persona1.apellidos = "García Fuentes"
print(persona1.apellidos)
```

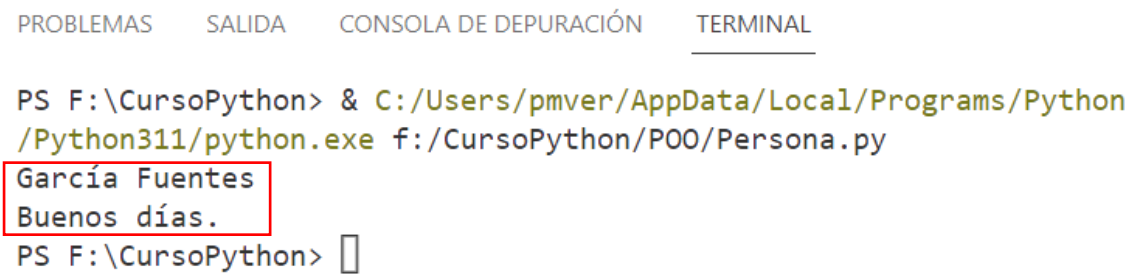
Este será el resultado:



Seguimos con el ejemplo:

```python
class Persona():
    # Propiedades, características o atributos:
    apellidos = ""
    nombre = ""
    edad = 0
    despierta = False

    # Funcionalidades:
    def despertar(self):
        print("Buenos días.")

persona1 = Persona()
persona1.apellidos = "García Fuentes"
print(persona1.apellidos)
persona1.despertar()
```

Este será el resultado:




```python
class Persona():
    # Propiedades, características o atributos:
    apellidos = ""
    nombre = ""
```

```
    edad = 0
    despierta = False

    # Funcionalidades:
    def despertar(self):
        # self: Parámetro que hace referencia a la
        # instancia perteneciente a la clase.
        self.despierta = True
        print("Buenos días.")

persona1 = Persona()
persona1.apellidos = "García Fuentes"
print(persona1.apellidos)
persona1.despertar()
print(persona1.despierta)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/POO/Persona.py
García Fuentes
Buenos días.
True
PS F:\CursoPython>
```

Agregamos las siguientes líneas:

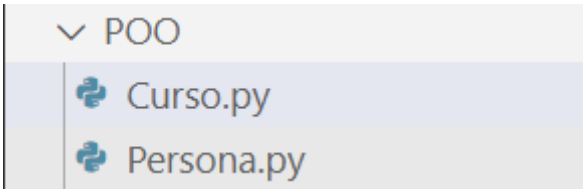
```
persona2 = Persona()
persona2.apellidos = "Paz Torres"
print(persona2.apellidos)
print(persona2.despierta)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/POO/Persona.py
García Fuentes
Buenos días.
True
Paz Torres
False
PS F:\CursoPython>
```

30.- Constructores de Clase en Python. Método _ _init_ _() para inicializar objetos

Dentro de la carpeta POO vamos a crear un archivo llamado Curso.py.




```
1   class Curso():
2       nombre = "Matemáticas"
3       creditos = 5
4       profesion = "Ingeniería Civil"
5
6   curso1 = Curso()
7   print(curso1.nombre)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/POO/Curso.py
Matemáticas
PS F:\CursoPython>
```

Los constructores sirven para dar un estado inicial a nuestro objeto.

```
1   class Curso():
2       def __init__(self):
3           self.nombre="Matemáticas"
4           self.creditos=5
5           self.profesion="Ingeniería Civil"
6
7   curso1 = Curso()
8   print(curso1.nombre)
```

Estado inicial del objeto, por medio del constructor.

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL
```

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/POO/Curso.py
Matemáticas
PS F:\CursoPython>
```

```
1  class Curso():
2      def __init__(self, nom, cre, pro):
3          self.nombre=nom
4          self.creditos=cre
5          self.profesion=pro
6
7  curso1 = Curso("Matemáticas", 5, "Ingeniería Civil")
8  print(curso1.nombre)
```

Este será el resultado final:

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/POO/Curso.py
Matemáticas
PS F:\CursoPython>
```

```
1  class Curso():
2      def __init__(self, nom, cre, pro):
3          self.nombre=nom
4          self.creditos=cre
5          self.profesion=pro
6
7  curso1 = Curso("Matemáticas", 5, "Ingeniería Civil")
8  print(curso1.nombre)
9
10 curso2 = Curso("Lenguaje", 4, "Ingeniería Industrial")
11 print(curso2.nombre)
```

Este será el resultado:

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/POO/Curso.py
Matemáticas
Lenguaje
PS F:\CursoPython>
```

31.- Encapsulamiento de Variables de Clase en Python

```
1  class Curso():
2      def __init__(self, nom, cre, pro):
3          self.nombre=nom
4          self.creditos=cre
5          self.profesion=pro
6          self.imparticion="Presencial"
7
8    curso1 = Curso("Matemáticas", 5, "Ingeniería Civil")
9    print(curso1.nombre)
10   curso1.imparticion="Virtual"
11   print(curso1.imparticion)
```

En el constructor agregamos una propiedad llamada impartición que por defecto será Presencial.

En la línea 10 a la instancia curso1.importacion la cambiamos a Virtual.

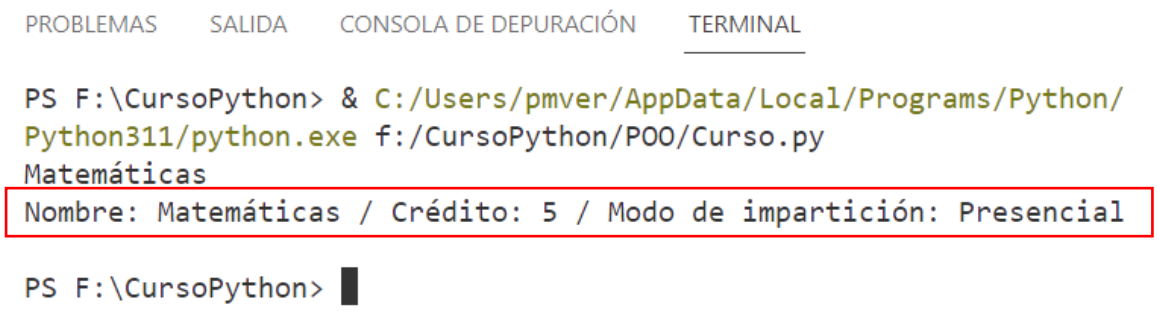
Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/POO/Curso.py
Matemáticas
Virtual
PS F:\CursoPython>
```

En que consiste el encapsulamiento, consiste en esconder u ocultar una propiedad o función dentro de una clase.

```
1  class Curso():
2      def __init__(self, nom, cre, pro):
3          self.nombre=nom
4          self.creditos=cre
5          self.profesion=pro
6          self.__imparticion="Presencial" # Propiedad encapsular
7
8      def mostrarDatos(self):
9          dat="Nombre: {0} / Crédito: {1} / Modo de impartición: {2}"
10         print(dat.format(self.nombre, self.creditos, self.__imparticion))
11
12   curso1 = Curso("Matemáticas", 5, "Ingeniería Civil")
13   print(curso1.nombre)
14   curso1.mostrarDatos()
```

Este será el resultado:



Desde dentro de la clase podemos crear una función y acceder a la propiedad __imparticion.

Además fuera de la clase no tendremos acceso a la propiedad __importacion pero si a la función mostrarDatos().

32.- Encapsulamiento de Método de Clase en Python

En este capítulo vamos a aprender a encapsular un función en Python.

```python
class Curso():
    def __init__(self, nom, cre, pro):
        self.nombre=nom
        self.creditos=cre
        self.profesion=pro
        self.__imparticion="Presencial" # Propiedad encapsular

    def mostrarDatos(self):
        dat="Nombre: {0} / Crédito: {1} / Modo de impartición: {2}"
        print(dat.format(self.nombre, self.creditos, self.__imparticion))
        docenteAsignado = self.__verificarDocente()
        if docenteAsignado: # True
            print("Exite docente asignado.")
        else:
            print("No es necesario asignar un docente...")

    def __verificarDocente(self):
        print("Verificando si existe docente asignado...")
        if self.__imparticion == "Presencial":
            return True
        else:
            return False

curso1 = Curso("Matemáticas", 5, "Ingeniería Civil")
print(curso1.nombre)
curso1.mostrarDatos()
```

La función que se encuentra dentro de la clase llama a la función encapsulada.

Función encapsulada.

Llamamos a la función que no está encapsulada

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/
Python311/python.exe f:/CursoPython/POO/Curso.py
Matemáticas
Nombre: Matemáticas / Crédito: 5 / Modo de impartición: Presencial
Verificando si existe docente asignado...
Exite docente asignado.
PS F:\CursoPython>
```

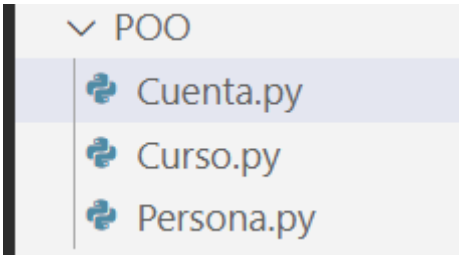
La función __verificarDocente(self): se encuentra encapsulada por los dos guiones bajos que tiene al principio, a esta función solo se tiene acceso desde la clase.

La función mostrarDatos(self): al encontrarse dentro de la clase tiene acceso a la función encapsulada y a la vez como esta no está encapsulada podemos tener acceso desde fuera de la clase.

33.- Método Accesores (GET – SET) en Python

En este capitulo vamos a aprender a usar los métodos Get y Set en Python.

Para este capítulo vamos a crear un nuevo fichero en la carpeta POO llamado Cuenta.py.



Para que sirven los métodos GET y SET, sirven para leer o modificar las propiedades que se encuentra encapsuladas de una clase.

```
1  class Cuenta():
2
3      def __init__(self, pro, sal, mon):
4          self.__propietario = pro
5          self.__saldo = sal
6          self.__moneda = mon
7
8      # Getters (método GET)
9      def get_Saldo(self):
10         return self.__saldo
11
12     def get_Propietario(self):
13         return self.__propietario
14
15     def get_Moneda(self):
16         return self.__moneda
17
18     # Setters (Método SET)
19     def set_Moneda(self, moneda):
20         self.__moneda = moneda
```

```
21
22
23    cuenta1=Cuenta("Oscar García", 15000, "Soles")
24
25    print(cuenta1.get_Saldo())
26    print(cuenta1.get_Moneda())
27    cuenta1.set_Moneda("Dólares")
28    print(cuenta1.get_Moneda())
29    print(cuenta1.get_Propietario())
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/
Python311/python.exe f:/CursoPython/POO/Cuenta.py
15000
Soles
Dólares
Oscar García
PS F:\CursoPython>
```

En la línea 3 en el método constructor definimos las tres variables encapsuladas.

En las líneas 9, 12 y 5 hacemos tres métodos get para poder obtener los datos fuera de la clase.

En la línea 19 hacemos un método set para modificar el tipo de moneda fuera de la clase.

En la línea 23 creamos una instancia llamada cuenta1, con sus correspondientes propiedades.

En la línea 25 gracias al método .get_Saldo() podemos ver el saldo por la consola.

En la línea 26 gracias al método .get_Moneda() podemos ver el tipo de moneda por consola

En la línea 27 gracias al método .set_Moneda() podemos cambiar el tipo de moneda.

En la línea 28 gracias al método .get_Moneda() podemos comprobar si se han realizado los cambios.

En la línea 29 gracias el método .get_Propietario podemos comprobar el nombre del propietario.

34.- Método de Clase __str__ (Conversión a String)

En este capitulo vamos a ver como sobrescribir la función __str__ en Python.

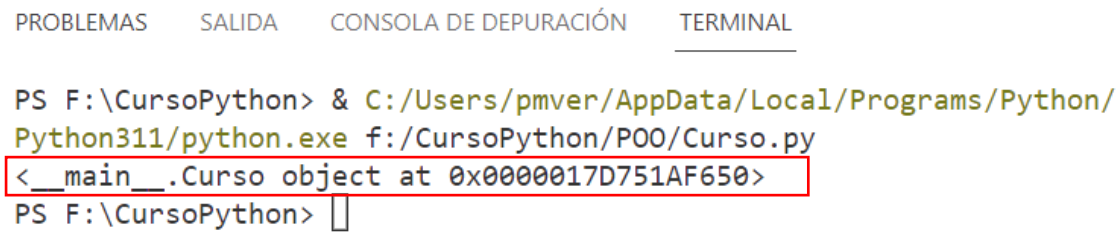
Vamos a modificar el código del fichero Curso1.py

```
1  class Curso():
2      def __init__(self, nom, cre, pro):
3          self.nombre=nom
4          self.creditos=cre
5          self.profesion=pro
6          self.__imparticion="Presencial" # Propiedad encapsular
7
8      def mostrarDatos(self):
9          dat="Nombre: {0} / Crédito: {1} / Modo de impartición: {2}"
10         print(dat.format(self.nombre, self.creditos, self.__imparticion))
11         docenteAsignado = self.__verificarDocente()
12         if docenteAsignado: # True
13             print("Exite docente asignado.")
14         else:
15             print("No es necesario asignar un docente...")
16
17     def __verificarDocente(self):
18         print("Verificando si existe docente asignado...")
19         if self.__imparticion == "Presencial":
20             return True
21         else:
22             return False
23
24     def __str__(self):
25         texto = "Nombre: {0} - Créditos: {1}"
26         return texto.format(self.nombre, self.creditos)
27
28 curso1 = Curso("Matemáticas", 5, "Ingeniería Civil")
29 print(curso1)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/
Python311/python.exe f:/CursoPython/POO/Curso.py
Nombre: Matemáticas - Créditos: 5
PS F:\CursoPython>
```

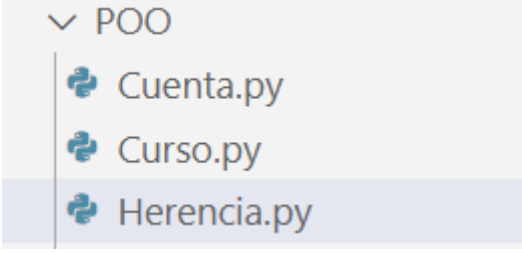
El método __str__(self): nos permitre que al mostrar la instancia muestre la información en el formato escrito en dicho método con solo **print(curso1**), de lo contrario pasaría esto:



Nos da una información del objeto, pero esto no es lo que queremos.

35.- Herencia en Python. Método super() | Programación orientada a Objetos

Dentro de la carpeta POO vamos a crear un nuevo fichero llamado Herncia.py.




```
1  class Persona():
2
3      def __init__(self, apePat, apeMat, nom):
4          self.apellidoPaterno = apePat
5          self.apellidoMaterno = apeMat
6          self.nombre = nom
7
8      def mostrarNombreCompleto(self):
9          txt="{0} {1}, {2}"
10         return txt.format(self.apellidoPaterno, self.apellidoMaterno, self.nombre)
11
12 class Estudiantes(Persona):
13     pass
14
15 estu1 = Estudiantes("Torres", "López", "Juan")
16 print(estu1.mostrarNombreCompleto())
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/
Python311/python.exe f:/CursoPython/POO/Herencia.py
Torres López, Juan
PS F:\CursoPython>
```

La clase Persona() tiene un método llamado mostrarNombreCompleto(Self).

En la línea 12 la clase Estudiantes hereda de Persona.

En la línea 15 creamos una instancia de Estudiantes llamada estu1, con sus respectivos parámetros.

En la línea 16 queremos utilizar el método mostrarNombreCompleto con la instancia estu1, en este ejemplo, como la instancia estu1 es de la clase Estudiantes a su vez hereda de Persona, todos los métodos de Persona pueden ser reutilizados por las instancias de Estudiantes.

En la línea 16 con la instancia estu1 llamamos al método .nostrarNombreCompleto() y como podrás comprobar funciona correctamente.

```
1   class Persona():
2
3       def __init__(self, apePat, apeMat, nom):
4           self.apellidoPaterno = apePat
5           self.apellidoMaterno = apeMat
6           self.nombre = nom
7
8       def mostrarNombreCompleto(self):
9           txt="{0} {1}, {2}"
10          return txt.format(self.apellidoPaterno, self.apellidoMaterno, self.nombre)
11
12  class Estudiantes(Persona):
13
14      def __init__(self, pro):
15          self.profesion = pro
16
17  estu1 = Estudiantes("Torres", "López", "Juan", "Ingeniería Civil")
18  print(estu1.mostrarNombreCompleto())
```

Si ejecutamos este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/POO/Herencia.py
Traceback (most recent call last):
  File "f:\CursoPython\POO\Herencia.py", line 17, in <module>
    estu1 = Estudiantes("Torres", "López", "Juan", "Ingeniería Civil")
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
TypeError: Estudiantes.__init__() takes 2 positional arguments but 5 were
 given
PS F:\CursoPython>
```

Este error es porque en la clase Estudiantes no está preparado para recibir todos los parámetros.

```
12  class Estudiantes(Persona):
13
14      def __init__(self, apePat, apeMat, nom, pro):
15          super().__init__(apePat, apeMat, nom)
16          self.profesion = pro
```

En la clase Estudiantes que hereda de Persona, en el constructor en la línea 14 le decimos cual serán los argumentos de Estudiantes.

En la línea 15 que argumetnos hereda de la clase Persona.

En la línea 16 los nuevos argumentos que tendrá la clase Estudiantes.

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/POO/Herencia.py
```

```
Torres López, Juan
PS F:\CursoPython>
```

Vemos que funciona correctamente.

Vamos a imprimir la característica profesión de la instancia estu1 de la clase Estudiantes.

Agregamos esta línea:

```
21    print(estu1.profesion)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python3
11/python.exe f:/CursoPython/POO/Herencia.py
Torres López, Juan
Ingeniería Civil
PS F:\CursoPython>
```

Funciona correctamente.

36.- Sobreescritura de Método en Python: Uso de Método super()

Seguimos utilizando el ejemplo del capítulo anterior.

```
1  class Persona():
2      def __init__(self, apePat, apeMat, nom):
3          self.apellidoPaterno = apePat
4          self.apellidoMaterno = apeMat
5          self.nombre = nom
6
7      def mostrarNombreCompleto(self):
8          txt="{0} {1}, {2}"
9          return txt.format(self.apellidoPaterno, self.apellidoMaterno, self.nombre)
10
11     def datos(self):
12         print(self.mostrarNombreCompleto())
13
14 class Estudiantes(Persona):
15     def __init__(self, apePat, apeMat, nom, pro):
16         super().__init__(apePat, apeMat, nom)
17         self.profesion = pro
18
19     def datos(self):
20         print("Profesion: {0}".format(self.profesion))
21
22 estu1 = Estudiantes("Torres", "López", "Juan", "Ingeniería Civil")
23
24 estu1.datos()
```

Estamos escribiendo el mismo método tanto para la clase Persona como la clase Estudiantes.

El código de ambos son distintos.

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/pytho
n.exe f:/CursoPython/POO/Herencia.py
Profesion: Ingeniería Civil
PS F:\CursoPython>
```

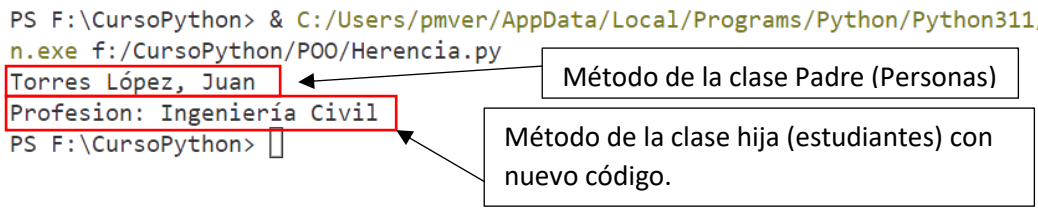
Solo se ejecuta el método datos de la clase Estudiantes, pero no hereda parte del código del método datos de la clase Persona.

```
19     def datos(self):
20         super().datos()
21         print("Profesion: {0}".format(self.profesion))
```

Modificamos el método creado en la clase Estudiantes, diciéndole que cuando ejecutemos este método primero ejecute el método datos de la clase personas y a continuación lo nuevo que corresponde a un dato “Profesión” que es de la clase Estudiantes.

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL
```



Resumiendo cuando tenemos dos métodos con el mismo nombre, el método que se va a ejecutar será el método que corresponde a la clase de su respectiva instancia, para poder utilizar el método de la clase padre, Personas debemos decir super().datos(), y a continuación agregar el nuevo código.

37.- Principio de Sustitución entre Clases

El principio de sustitución significa que cuando aplicamos herencias existe clases padres y clases hijas, las clases hijas en este caso Estudiantes es siempre una Persona sin embargo no se puede decir lo contrario, no se puede decir que una Persona siempre es un Estudiante.

Siguiendo con el fichero Herencia.py al final del mismo vamos a agregar el siguiente código:

```
25    print(isinstance(estu1, Estudiantes))
26    print(isinstance(estu1, Persona))
```

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/pytho
n.exe f:/CursoPython/POO/Herencia.py
True
True
PS F:\CursoPython>
```

La instancia estu1 pertenece a la clase Estudiantes como la clase Persona.

Agregamos las siguientes líneas.

```
28    perso1 = Persona("Martínez", "Fernández", "Antonio")
29    print(isinstance(perso1, Estudiantes))
```

Este será el resultado:

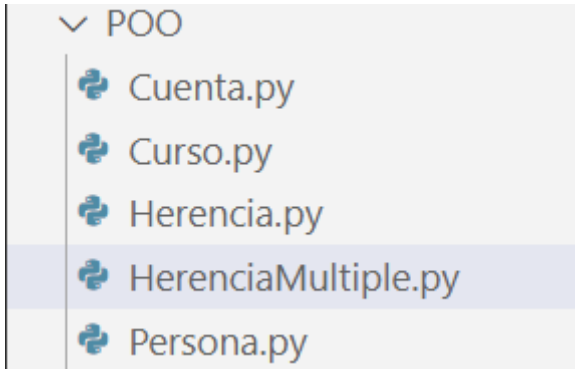
```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/pytho
n.exe f:/CursoPython/POO/Herencia.py
True
True
False
PS F:\CursoPython>
```

La instancia perso1 de la clase Persona no tiene que ser una instancia de la clase Estudiantes.

Por el principio de sustitución diremos que un estudiante siempre es una persona, pero una persona no tiene por qué ser estudiante.

38.- Herencia MÚLTIPLE en Python | Programación Orientada a Objetos

Dentro de la carpeta POO vamos a crear un fichero llamado HerenciaMultiple.py.




```python
class ClaseA():
    def __init__(self, par1, par2):
        self.parametro1 = par1
        self.parametro2 = par2

class ClaseB():
    def __init__(self, par3, par4, par5):
        self.parametro3 = par3
        self.parametro4 = par4
        self.parametro5 = par5

class ClaseX(ClaseA, ClaseB):
    pass

cX1 = ClaseX()
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe f:/CursoPython/POO/HerenciaMultiple.py
Traceback (most recent call last):
  File "f:\CursoPython\POO\HerenciaMultiple.py", line 15, in <module>
    cX1 = ClaseX()
          ^^^^^^^^
TypeError: ClaseA.__init__() missing 2 required positional arguments: 'par1' and 'par2'
PS F:\CursoPython>
```

Se genera un error. Dice la ClaseX requiere de dos argumentos. ‘par1’ y ‘par2’

```
12  class ClaseX(ClaseB, ClaseA):
13      pass
```

Si invertimos el orden de los argumentos.

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/py
thon.exe f:/CursoPython/POO/HerenciaMultiple.py
Traceback (most recent call last):
  File "f:\CursoPython\POO\HerenciaMultiple.py", line 15, in <module>
    cX1 = ClaseX()
          ^^^^^^^^
TypeError: ClaseB.__init__() missing 3 required positional arguments: 'par3',
'par4', and 'par5'
PS F:\CursoPython>
```

Nos va a decir que al ClaseX requiere de tres argumentos. ‘par3’, ‘par4’ y ‘par5’.

La primera clase marca el número de argumentos.

```
12  class ClaseX(ClaseA, ClaseB):
13      pass
14
15  cX1 = ClaseX(15, 21)
```

Como la ClaseA que la ponemos en primer lugar requiere de dos argumentos, el la línea 15 le pasamos dos valores.

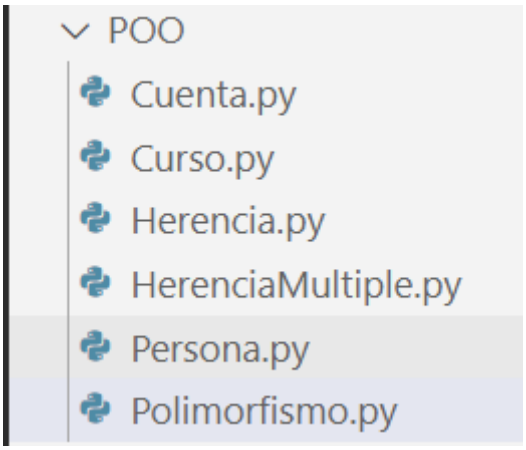
Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/py
thon.exe f:/CursoPython/POO/HerenciaMultiple.py
PS F:\CursoPython>
```

No genera ningún error.

39.- Polimorfismo en Python | Programación Orientada a Objetos

Dentro de la carpeta POO vamos a crear un nuevo fichero llamado Polimorfismo.py.



Polimorfismo viene de la palabra poli => muchas y morfos -> formas.

Como finalidad que un objeto pueda cambiar, pueda mutar de forma dependiendo del contexto en el cual se utilice.

Y al cambiar de forma un objeto también cambia su comportamiento.

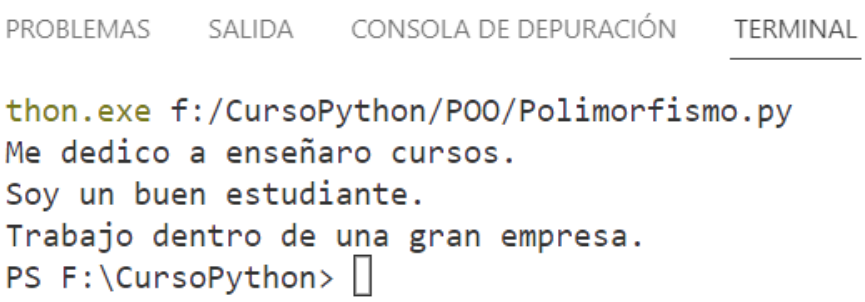
En Python es muy fácil incrementar el polimorfismo ya que Python como tal es un lenguaje de tipado dinámico.

Vamos a crear 3 clases:

```
1   class Estudiante():
2       def describir(self):
3           print("Soy un buen estudiante.")
4
5   class Docente():
6       def describir(self):
7           print("Me dedico a enseñaro cursos.")
8
9   class Trabajador():
10      def describir(self):
11          print("Trabajo dentro de una gran empresa.")
12
13  def describirPersona(persona):
14      persona.describir()
15
16  doc1 = Docente()
17  describirPersona(doc1)
18
```

```
19    est1 = Estudiante()
20    describirPersona(est1)
21
22    tra1 = Trabajador()
23    describirPersona(tra1)
```

Este será el resultado:

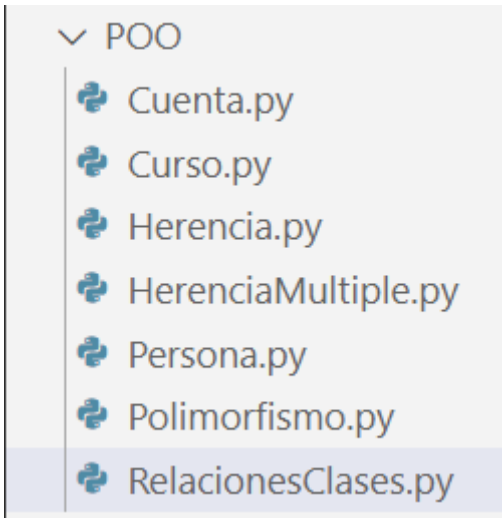


El método describirPersona(objeto) según el tipo de objeto que le pasamos por parámetro, ejecutará el método de la clase correspondiente.

El polimorfismo en Python solo con llamar al método y pasar como argumento el tipo de objeto ya sabe que método tiene que ejecutar ya que ejecutará el método de la clase perteneciente al objeto que hemos pasado como argumento.

40.- Relaciones entre Clase en Python (Dependencia entre Clases)

Dentro de la carpeta POO vamos a crear un nuevo fichero llamado RelacionesClases.py.



Estas relaciones pueden ser de dependencia o independencia, vamos a verlo con un ejemplo:

```python
class Pais():
    def __init__(self, nom, pre):
        self.nombre = nom
        self.presidente = pre

    def __str__(self):
        txt = "País: {0} - Presidente: {1}"
        return txt.format(self.nombre, self.presidente)

pais1 = Pais("Perú", "Martín Vizcarra")
print(pais1)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/py
thon.exe f:/CursoPython/POO/RelacionesClases.py
País: Perú - Presidente: Martín Vizcarra
PS F:\CursoPython>
```

```python
class Pais():
    def __init__(self, nom, pre):
        self.nombre = nom
        self.presidente = pre

    def __str__(self):
        txt = "País: {0} - Presidente: {1}"
        return txt.format(self.nombre, self.presidente)
```

```
9
10  class Ciudad():
11      def __init__(self, nom, hab, pai):
12          self.nombre = nom
13          self.habitantes = hab
14          self.pais = pai
15
16      def __str__(self):
17          txt = "Ciudad: {0} - Nº Habitantes: {1} ({2})"
18          return txt.format(self.nombre, self.habitantes, self.pais)
19
20  pais1 = Pais("Perú", "Martín Vizcarra")
21  print(pais1)
22
23  ciudad1 = Ciudad("Chiclayo", 150000, pais1)
24  print(ciudad1)
```

En la línea 23 cuando instanciamos un objeto llamado ciudad1 de la clase Ciudad, como parámetros le pasamos (el nombre de una ciudad), (El número de habitantes) y como tercer argumento el objeto de tipo País llamado pais1.

De este modo hacemos una relación entre las clases Pais y Ciudad.

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

exe f:/CursoPython/POO/RelacionesClases.py
País: Perú - Presidente: Martín Vizcarra
Ciudad: Chiclayo - Nº Habitantes: 150000 (País: Perú - Presidente: Martín Vizcarra)
PS F:\CursoPython>
```

Vamos a crear un tercera clase para relacionarla:

```
1   class Pais():
2       def __init__(self, nom, pre):
3           self.nombre = nom
4           self.presidente = pre
5
6       def __str__(self):
7           txt = "País: {0} - Presidente: {1}"
8           return txt.format(self.nombre, self.presidente)
9
10  class Ciudad():
11      def __init__(self, nom, hab, pai):
12          self.nombre = nom
13          self.habitantes = hab
14          self.pais = pai
15
16      def __str__(self):
17          txt = "Ciudad: {0} - Nº Habitantes: {1} ({2})"
18          return txt.format(self.nombre, self.habitantes, self.pais)
```

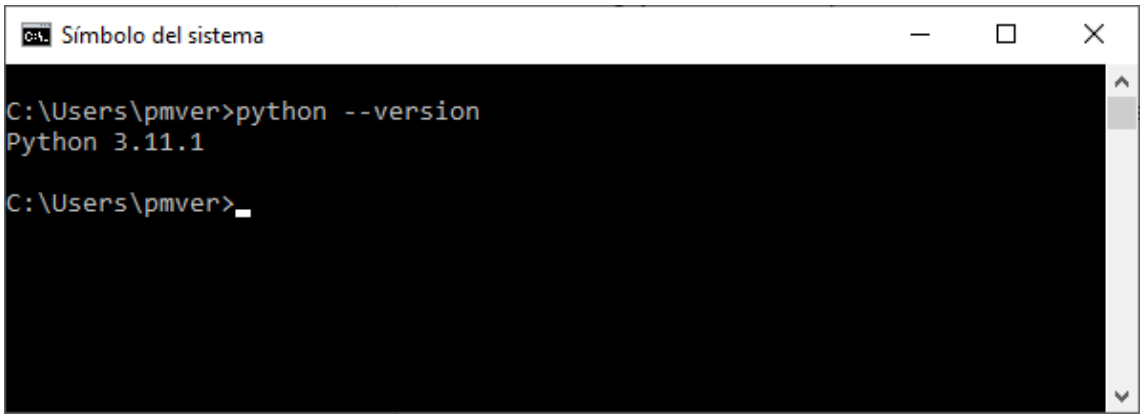
```
19
20  class Urbanizacion():
21      def __init__(self, nom, ciu):
22          self.nombre = nom
23          self.ciudad = ciu
24
25      def __str__(self):
26          txt = "Urbanización: {0} ({1})"
27          return txt.format(self.nombre, self.ciudad)
28
29  pais1 = Pais("Perú", "Martín Vizcarra")
30  print(pais1)
31
32  ciudad1 = Ciudad("Chiclayo", 150000, pais1)
33  print(ciudad1)
34
35  urba1 = Urbanizacion("María de los Ángeles", ciudad1)
36  print(urba1)
```

Este será el resultado cuando ejecutemos:

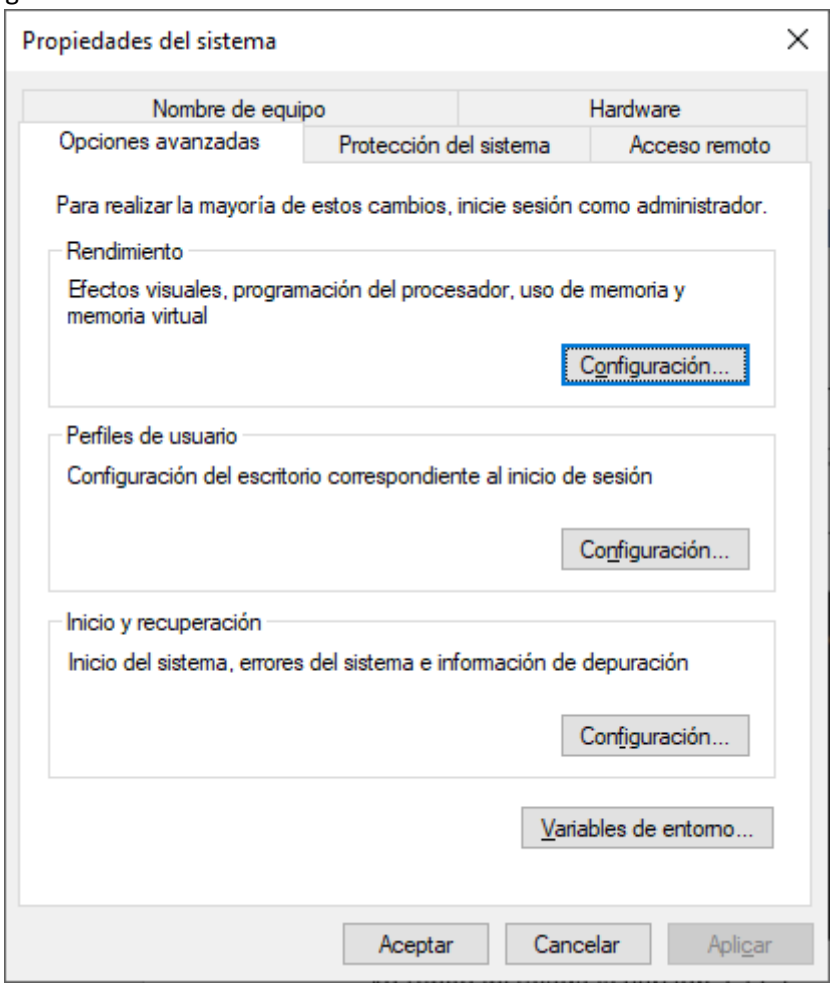
```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

exe f:/CursoPython/POO/RelacionesClases.py
País: Perú - Presidente: Martín Vizcarra
Ciudad: Chiclayo - Nº Habitantes: 150000 (País: Perú - Presidente: Martín Vizcarra)
Urbanización: María de los Ángeles (Ciudad: Chiclayo - Nº Habitantes: 150000 (País:
 Perú - Presidente: Martín Vizcarra))
PS F:\CursoPython>
```

41.- Conexión con MySQL | Instalación de Driver mysql-connector-python.

¿Qué necesitamos para conectarnos a una base de datos?
El SQLite3 ya viene preparado con Python.
Para saber que versión de Python tengo instalada, desde la ventana cmd y escribiremos python --version

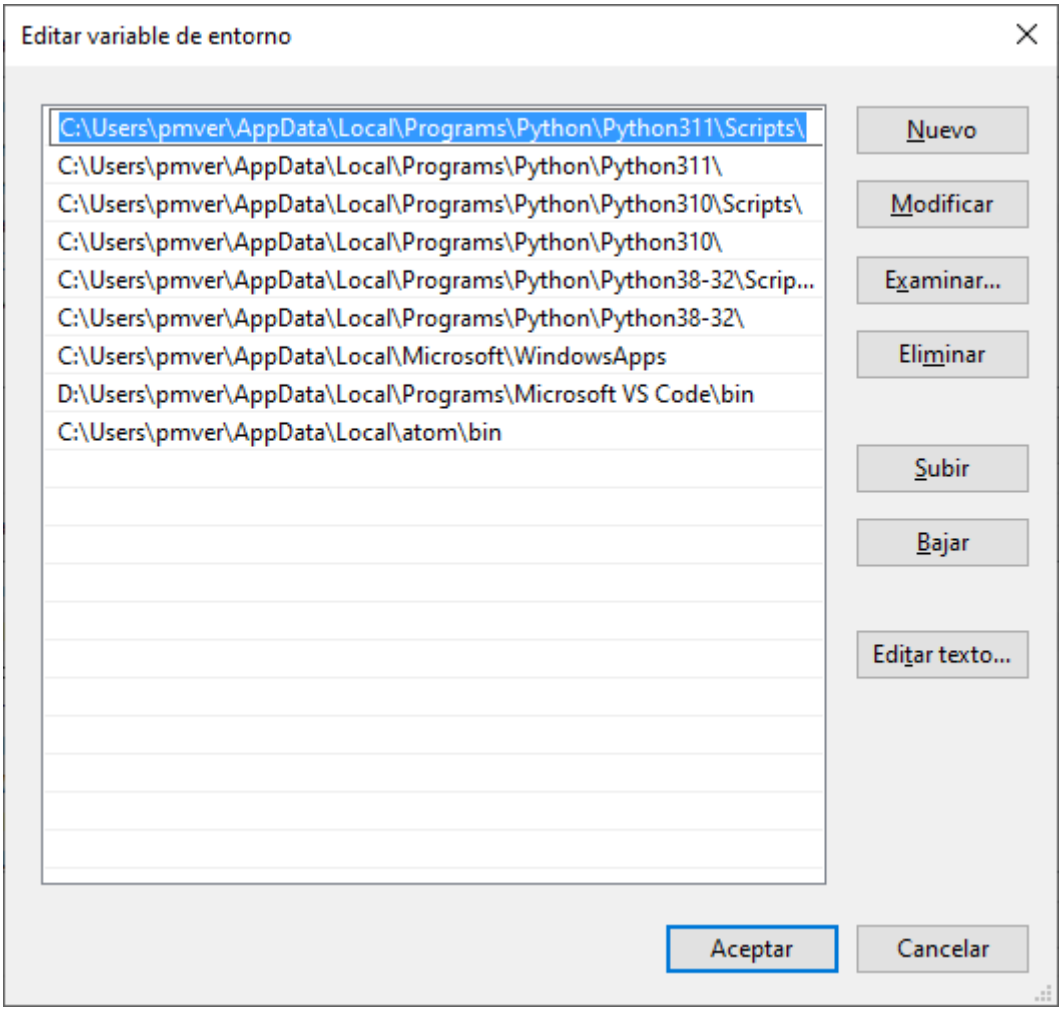


Yo tengo instalada la versión 3.11.1
Vamos a buscar la carpeta donde tenemos el Python.
Desde el explorador de Windows seleccionamos Este equipo con el botón derecho del ratón y seleccionaremos propiedades.
De este configuración avanzada del sistema:



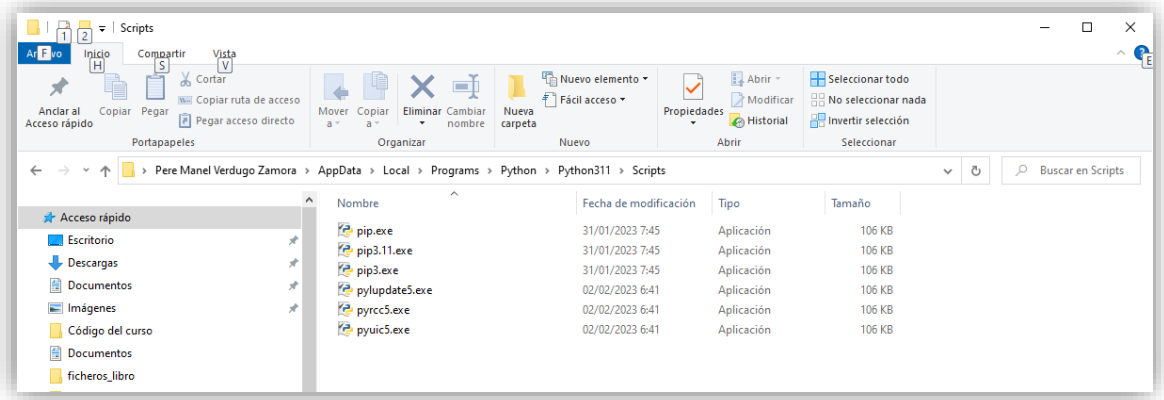
Seleccionaremos Variables de entorno…

Dentro del apartado Path, seleccionaremos Editar.

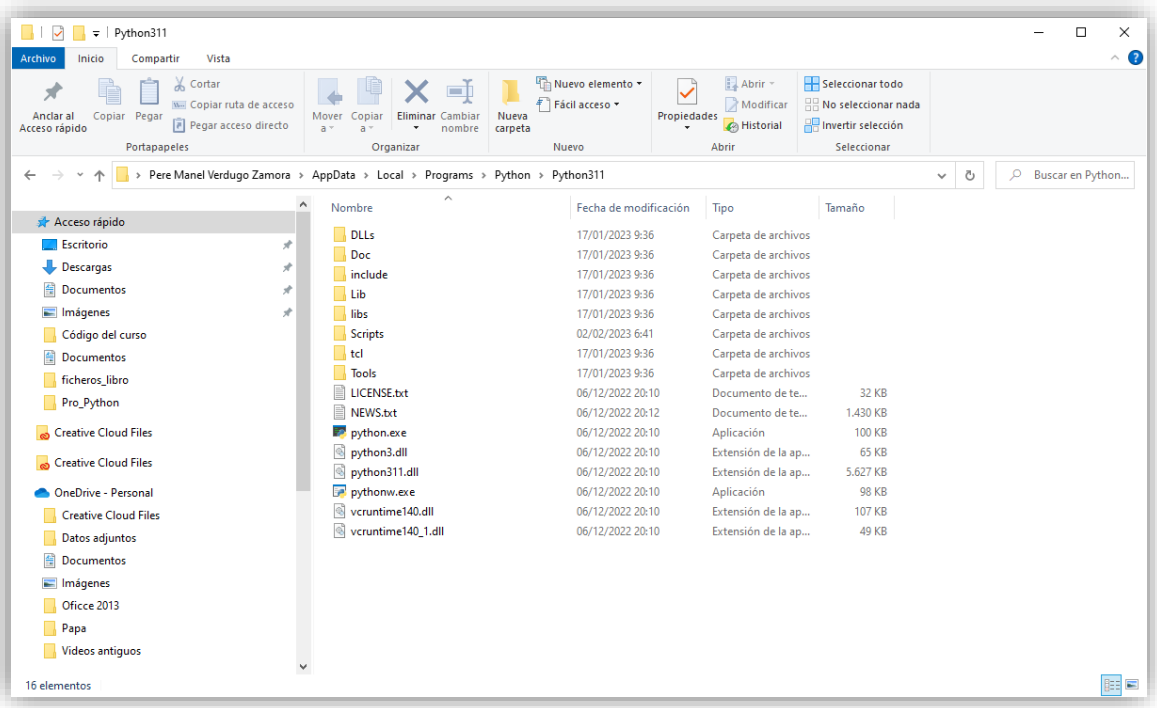


Copiamos esta ruta.

Desde el explorador vamos a acceder a esta carpeta:

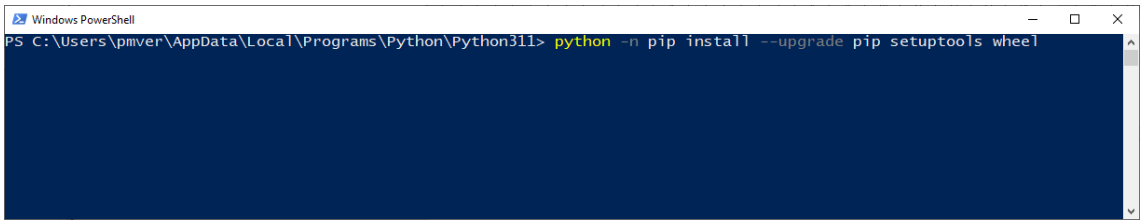


Ahora subimos un nivel.



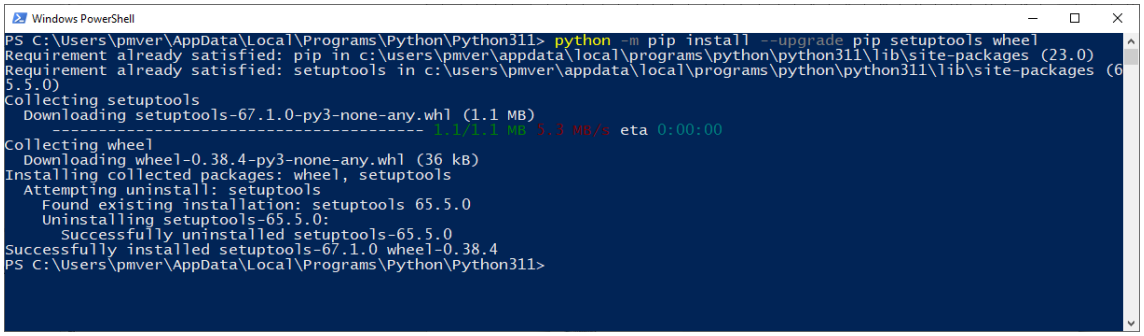
Ya estamos en la carpeta que contiene el Python principal.

Manteniendo la tecla shift presionado y botón derecho del ratón seleccionaremos Abrir la ventana de PowerShell aquí.



Introduciremos la siguiente instrucción:

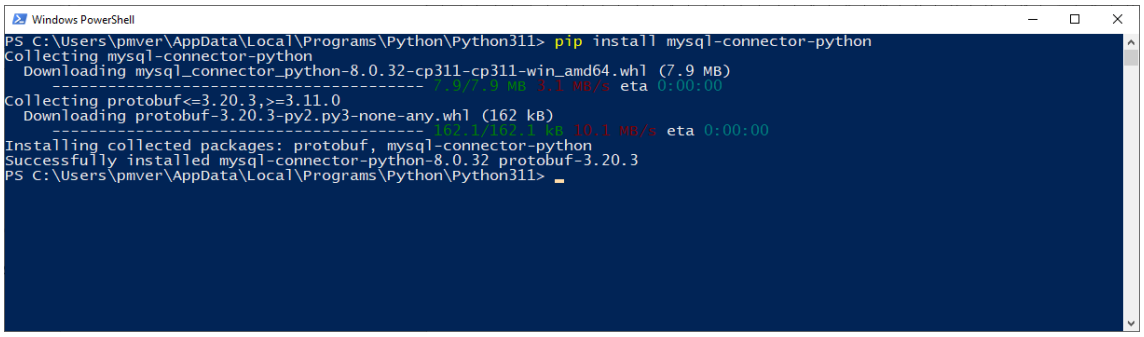
python -m pip install --upgrade pip setuptools wheel



Se tienen que instalar o actualizar.

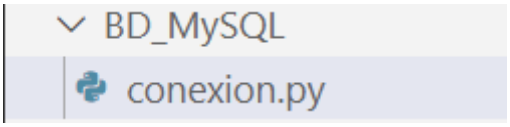
La siguiente instrucción:

pip install mysql-connector-python



Ahora vamos a realizar una nueva carpeta para estos proyectos, llamada BD-MySQL

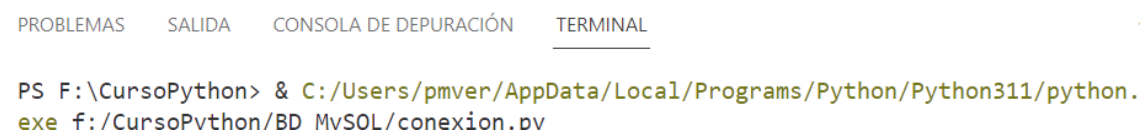
En esta carpeta haremos un fichero llamado conexion.py.



Vamos a escribir el siguiente código:

```
1    import mysql.connector
```

Vamos a ejecutar, si no da ningún tipo de error es que lo hemos instalado correctamente.

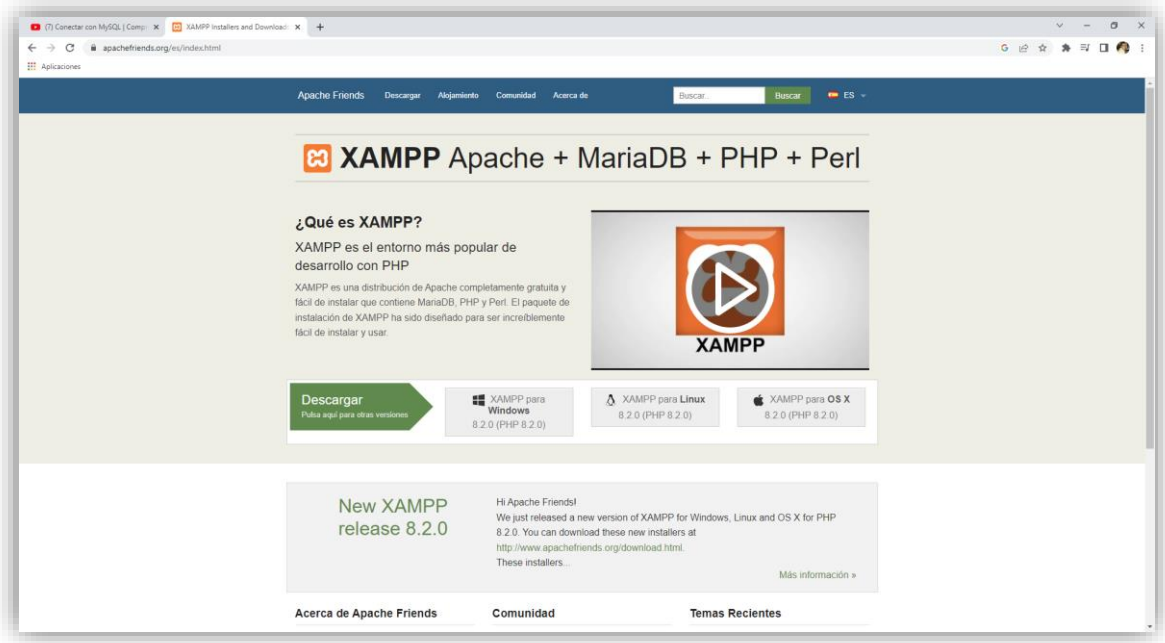


Hemos importado de forma exitosa el conector de mysql.

42.- Conectar con MySQL | Comprobar conexión y leer Datos del Servidor

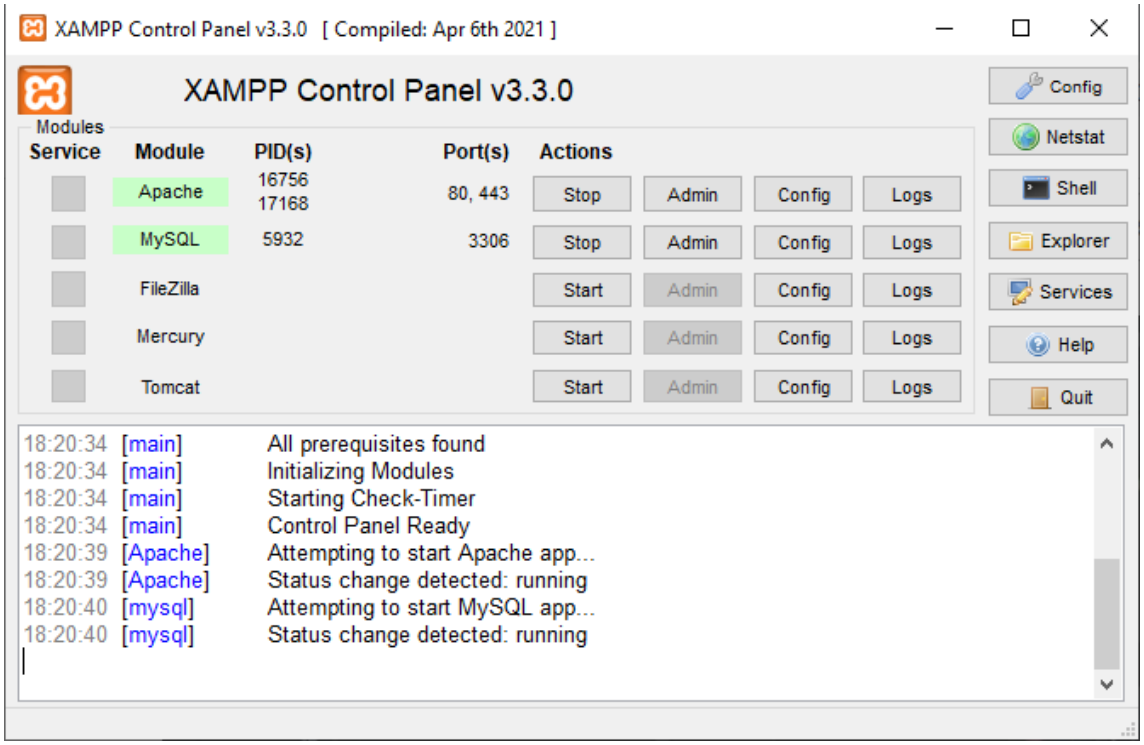
Para poder realizar esta práctica tendrás que instalar XAMPP.

https://www.apachefriends.org/es/index.html



Después de descargarlo procederemos a su instalación.

Cuando lo ejecutemos:

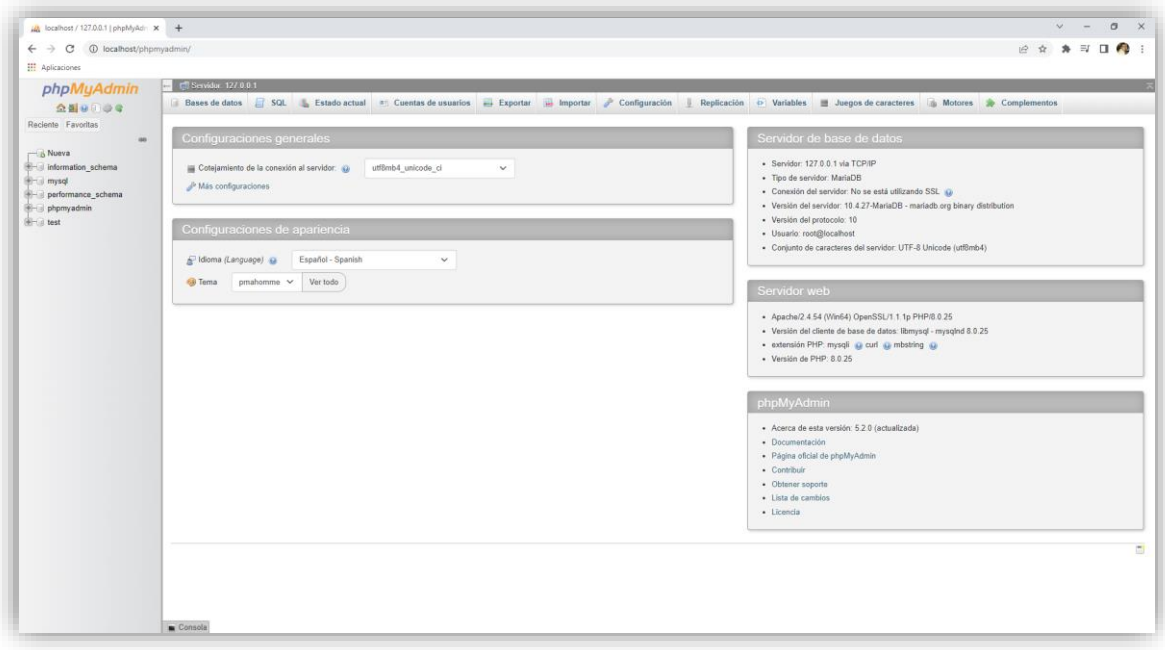


Activaremos Apache y MySql.

Ya podemos minimizar esta ventana pero no cerrarla.

Ejecutaremos el navegador y accederemos a la siguiente dirección.

http://localhost/phpmyadmin/



Este será el código:

```python
import mysql.connector
from mysql.connector import Error

try:
    conexion = mysql.connector.connect(
        host="localhost",
        port=3306,
        user='root',
        password='',
        db='condominio'
    )
    if conexion.is_connected():
        print("Conexión exitosa.")
except Error as ex:
    print("Error durante la conexión:", ex)
```

Si ejecutamos este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.
exe f:/CursoPython/BD_MySQL/conexion.py
Conexión exitosa.
PS F:\CursoPython>
```

```python
import mysql.connector
from mysql.connector import Error

try:
    conexion = mysql.connector.connect(
        host="localhost",
        port=3306,
        user='root',
        password='',
        db='condominio'
    )
    if conexion.is_connected():
        print("Conexión exitosa.")
        infoServer = conexion.get_server_info()
        print("Info del servidor: ", infoServer)
except Error as ex:
    print("Error durante la conexión:", ex)
```

Agrega las siguientes líneas y ejecuta de nuevo:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.
exe f:/CursoPython/BD_MySQL/conexion.py
Conexión exitosa.
Info del servidor:  5.5.5-10.4.27-MariaDB
PS F:\CursoPython>
```

```python
import mysql.connector
from mysql.connector import Error

try:
    conexion = mysql.connector.connect(
        host="localhost",
        port=3306,
```

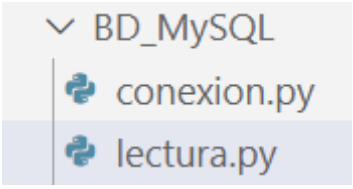
```
        user='root',
        password='',
        db='condominio'
    )
    if conexion.is_connected():
        print("Conexión exitosa.")
        infoServer = conexion.get_server_info()
        print("Info del servidor: ", infoServer)
except Error as ex:
    print("Error durante la conexión:", ex)
finally:
    if conexion.is_connected():
        conexion.close()
        print("La conexión ha finalizado")
```

Agrega las siguientes líneas y ejecuta para ver el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

exe f:/CursoPython/BD_MySQL/conexion.py
Conexión exitosa.
Info del servidor:  5.5.5-10.4.27-MariaDB
La conexión ha finalizado
PS F:\CursoPython>
```

43.- Sentencia SELECT (Leer Registros) de la Base de Datos MySQL

Dentro de la carpeta MySQL vamos a crear otro fichero llamado lectura.py.



Vamos a copiar todo el código del capítulo anterior.

```python
import mysql.connector
from mysql.connector import Error

try:
    conexion = mysql.connector.connect(
        host="localhost",
        port=3306,
        user='root',
        password='',
        db='condominio'
    )
    if conexion.is_connected():
        print("Conexión exitosa.")
        cursor = conexion.cursor()
        cursor.execute("SELECT database();")
        registro=cursor.fetchone()
        print("Conectado a la BD. ", registro)
except Error as ex:
    print("Error durante la conexión:", ex)
finally:
    if conexion.is_connected():
        conexion.close()
        print("La conexión ha finalizado")
```

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python
exe f:/CursoPython/BD_MySQL/lectura.py
Conexión exitosa.
Conectado a la BD.  ('condominio',)
```

```
La conexión ha finalizado
PS F:\CursoPython>
```

Línea 14 es para realizar los pasos necesarios para ejecutar una consulta para esto necesitamos un cursor. Un cursor es un objeto relacionado con las bases de datos y que actúa como un nexo para poder hacer lectura de datos y también para pode hacer inserciones. Y no solamente inserciones sino también actualizaciones y eliminaciones.

Línea 15 cursor.execute() que nos va a permitir ejecutar una sentencia SQL.

“SELECT database();”) nos permite obtener el nombre de la base de datos.

Línea 15 la variable registro va a obtener el registro que hemos realizado, fetch significa ir a buscar, se puede determinar así. Aquí esta la tabla a leer con el nombre de tipousuario.

		Codigo	Nombre	Vigencia
☐	Editar Copiar Borrar	1	Administrador	1
☐	Editar Copiar Borrar	2	Propietario	1
☐	Editar Copiar Borrar	3	Inquilino	1
☐	Editar Copiar Borrar	4	Visitante	1

Queremos consultar la tabla tipousuario.

```python
import mysql.connector
from mysql.connector import Error

try:
    conexion = mysql.connector.connect(
        host="localhost",
        port=3306,
        user='root',
        password='',
        db='condominio'
    )
    if conexion.is_connected():
        print("Conexión exitosa.")
        cursor = conexion.cursor()
        cursor.execute("SELECT database();")
        registro=cursor.fetchone()
        print("Conectado a la BD. ", registro)
        cursor.execute("SELECT * FROM tipousuario")
        resultados=cursor.fetchall()
        for fila in resultados:
            print("Codigo: ", fila[0], "Nombre: ", fila[1], "Vigencia: ", fila[2])
except Error as ex:
    print("Error durante la conexión:", ex)
finally:
    if conexion.is_connected():
        conexion.close()
        print("La conexión ha finalizado")
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.
exe f:/CursoPython/BD_MySQL/lectura.py
Conexión exitosa.
Conectado a la BD.  ('condominio',)
Codigo:  1 Nombre:  Administrador Vigencia:  1
Codigo:  2 Nombre:  Propietario Vigencia:  1
Codigo:  3 Nombre:  Inquilino Vigencia:  1
Codigo:  4 Nombre:  Visitante Vigencia:  1
La conexión ha finalizado
PS F:\CursoPython>
```

Para que nos diga el total de registros consultados.

```
20          for fila in resultados:
21              print("Codigo: ", fila[0], "Nombre: ", fila[1], "Vigencia: ", fila[2])
22          print("Total de registros: ", cursor.rowcount)
```

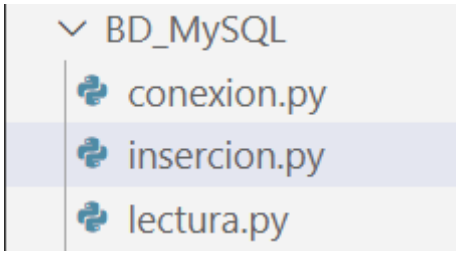
Después de imprimir todos los registros añadimos el código de la línea 22 este nos dirá el número de registros consultados.

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.
exe f:/CursoPython/BD_MySQL/lectura.py
Conexión exitosa.
Conectado a la BD.  ('condominio',)
Codigo:  1 Nombre:  Administrador Vigencia:  1
Codigo:  2 Nombre:  Propietario Vigencia:  1
Codigo:  3 Nombre:  Inquilino Vigencia:  1
Codigo:  4 Nombre:  Visitante Vigencia:  1
Total de registros:  4
La conexión ha finalizado
PS F:\CursoPython>
```

44.- Sentencia INSERT INTO (Inserción de Registros) en Base de Datos MySQL

Para este capítulo vamos a crear un nuevo fichero inserción.py en la carpeta DB_MySQL.



Copiamos el código del fichero lectura.py

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	Codigo	tinyint(1)			No	*Ninguna*		AUTO_INCREMENT
2	Nombre	varchar(20)	utf8mb4_general_ci		No	*Ninguna*		
3	Vigencia	tinyint(1)			No	1		

```
1   import mysql.connector
2   from mysql.connector import Error
3
4   try:
5       conexion = mysql.connector.connect(
6           host="localhost",
7           port=3306,
8           user='root',
9           password='',
10          db='condominio'
11      )
12      if conexion.is_connected():
13          print("Conexión exitosa.")
14          cursor = conexion.cursor()
15          cursor.execute("INSERT INTO tipousuario (nombre) VALUES ('Vigilante')")
16          conexion.commit()  # Confirma la acción que estamos ejecutando.
17          print("Registro insertado con exito.")
18  except Error as ex:
19      print("Error durante la conexión:", ex)
20  finally:
21      if conexion.is_connected():
22          conexion.close()
23          print("La conexión ha finalizado")
```

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.
exe f:/CursoPython/BD_MySQL/insercion.py
Conexión exitosa.
Registro insertado con exito.
La conexión ha finalizado
```

```
PS F:\CursoPython>
```

Vamos a ver la base de datos:

Codigo	Nombre	Vigencia
1	Administrador	1
2	Propietario	1
3	Inquilino	1
4	Visitante	1
6	Vigilante	1

Ya hemos agregado el registro.

Ahora lo que vamos a realizar es que el usuario ingrese por teclado en registro que queremos agregar.

```
1   import mysql.connector
2   from mysql.connector import Error
3
4   try:
5       conexion = mysql.connector.connect(
6           host="localhost",
7           port=3306,
8           user='root',
9           password='',
10          db='condominio'
11      )
12      if conexion.is_connected():
13          print("Conexión exitosa.")
14          cursor = conexion.cursor()
15          nombre = input("Ingrese nombre de usuario:")
16          sentencia ="INSERT INTO tipousuario (nombre) VALUES ('{0}')".format(nombre)
17          cursor.execute(sentencia)
18          conexion.commit()  # Confirma la acción que estamos ejecutando.
19          print("Registro insertado con exito.")
20  except Error as ex:
21      print("Error durante la conexión:", ex)
22  finally:
23      if conexion.is_connected():
24          conexion.close()
25          print("La conexión ha finalizado")
```

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.
exe f:/CursoPython/BD_MySQL/insercion.py
Conexión exitosa.
Ingrese nombre de usuario:Huesped
Registro insertado con exito.
La conexión ha finalizado
```

```
PS F:\CursoPython>
```

Vamos a consultar la base de datos:

Codigo	Nombre	Vigencia
1	Administrador	1
2	Propietario	1
3	Inquilino	1
4	Visitante	1
6	Vigilante	1
7	Huesped	1

Vuelve a ejecutar el programa para ingresar Recepcionista.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.
exe f:/CursoPython/BD_MySQL/insercion.py
Conexión exitosa.
Ingrese nombre de usuario:Recepcionista
Registro insertado con exito.
La conexión ha finalizado
PS F:\CursoPython>
```

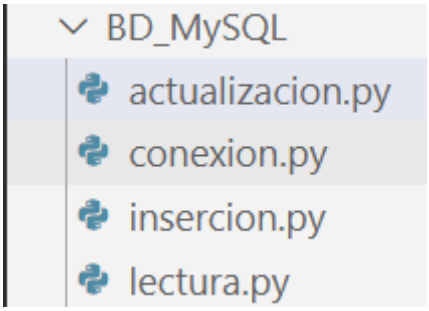
Vamos a consultar la base de datos:

Codigo	Nombre	Vigencia
1	Administrador	1
2	Propietario	1
3	Inquilino	1
4	Visitante	1
6	Vigilante	1
7	Huesped	1
8	Recepcionista	1

45.- Sentencia UPDATE (Actualización de Registros) en Base de Datos MySQL

En este capítulo vamos a aprender a actualizar o modificar una tabla.

Vamos a crear en la carpeta DB_MySQL otro fichero llamado actualización.py.



Copiamos todo el código del fichero inserción.py al nuevo fichero.

Codigo	Nombre	Vigencia
1	Administrador	1
2	Propietario	1
3	Inquilino	1
4	Visitante	1
6	Vigilante	1
7	Huesped	1
8	Recepcionista	1

En el registro 6 que pone Vigilante lo queremos cambiar por el de Conserje.

```
import mysql.connector
from mysql.connector import Error

try:
    conexion = mysql.connector.connect(
        host="localhost",
        port=3306,
        user='root',
        password='',
        db='condominio'
    )
    if conexion.is_connected():
        print("Conexión exitosa.")
        cursor = conexion.cursor()
        cursor.execute("UPDATE tipousuario SET nombre = 'Conserje' WHERE Codigo = 6")
        conexion.commit()  # Confirma la acción que estamos ejecutando.
        print("Registro actualizado con éxito.")
except Error as ex:
    print("Error durante la conexión:", ex)
```

```
20  finally:
21      if conexion.is_connected():
22          conexion.close()
23          print("La conexión ha finalizado")
```

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

exe f:/CursoPython/BD_MySQL/actualizacion.py
Conexión exitosa.
Registro actualizado con éxito.
La conexión ha finalizado
PS F:\CursoPython>
```

Vamos a consultar la tabla:

Codigo	Nombre	Vigencia
1	Administrador	1
2	Propietario	1
3	Inquilino	1
4	Visitante	1
6	Conserje	1
7	Huesped	1
8	Recepcionista	1

El registro 6 ya se ha actualizado.

Ahora vamos a ver como cambiar dos valores a la vez, el Nombre y la Vigencia.

Vamos a modificar el registro 7, como Nombre Monitor y como Vigencia 0.

```
cursor.execute("UPDATE tipousuario SET nombre =
'Monitor', Vigencia = 0 WHERE Codigo = 7")
```

Esto lo cambiaremos en la línea 15 de nuestro código.

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe f:/CursoPython/
BD_MySQL/actualizacion.py
Conexión exitosa.
Registro actualizado con éxito.
La conexión ha finalizado
```

```
PS F:\CursoPython>
```

Vamos a consultar la tabla:

Codigo	Nombre	Vigencia
1	Administrador	1
2	Propietario	1
3	Inquilino	1
4	Visitante	1
6	Conserje	1
7	Monitor	0
8	Recepcionista	1

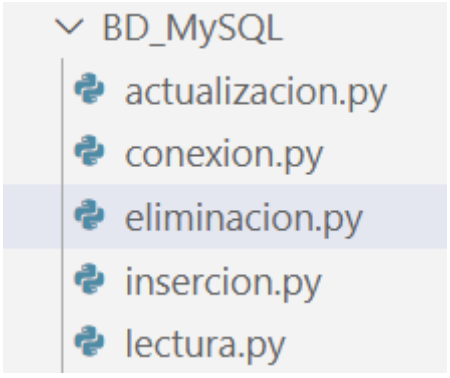
```
15          cursor.execute("""UPDATE tipousuario SET nombre = 'Monitor',
16                            Vigencia = 0 WHERE Codigo = 7""")
```

Si el código para una sola línea es muy largo lo puedes realizar entre 3 comillas dobles y así poderlo realizar en varias líneas.

46.- Sentencia DELETE (Eliminación de Registros) en Base de Datos MySQL

En este capítulo vamos a aprender como eliminar datos de una tabla en una Base de Datos MySQL.

En la Carpeta DB_MySQL vamos a crear un nuevo fichero llamado eliminación.py.



Vamos a copiar el código que teníamos en actualización.py.

Codigo	Nombre	Vigencia
1	Administrador	1
2	Propietario	1
3	Inquilino	1
4	Visitante	1
6	Conserje	1
7	Monitor	0
8	Recepcionista	1

Vamos a eliminar Monitor que tiene el código 7.

```
import mysql.connector
from mysql.connector import Error

try:
    conexion = mysql.connector.connect(
        host="localhost",
        port=3306,
        user='root',
        password='',
        db='condominio'
    )
    if conexion.is_connected():
        print("Conexión exitosa.")
        cursor = conexion.cursor()
        cursor.execute("DELETE FROM tipousuario WHERE Codigo = 7 AND Vigencia = 0")
        conexion.commit()  # Confirma la acción que estamos ejecutando.
        print("Registro Eliminado.")
```

```
18    except Error as ex:
19        print("Error durante la conexión:", ex)
20    finally:
21        if conexion.is_connected():
22            conexion.close()
23            print("La conexión ha finalizado")
```

En la línea 15 le estamos diciendo que elimine el registro conde Codigo es igual a 7 y Vigencia es igual a 0

El operador AND lo puedes cambiar por OR, pero ten mucho cuidado ya que el resultado no puede ser el mismo.

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

Conexión exitosa.
Registro Eliminado.
La conexión ha finalizado
PS F:\CursoPython>
```

Vamos a consultar la tabla:

Codigo	Nombre	Vigencia
1	Administrador	1
2	Propietario	1
3	Inquilino	1
4	Visitante	1
6	Conserje	1
8	Recepcionista	1

El registro 7 ya se ha eliminado.

Vamos a consultarlo ejecutando el fichero lectura.py.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.exe f:/CursoPython/
BD_MySQL/lectura.py
Conexión exitosa.
Conectado a la BD.  ('condominio',)
Codigo:  1 Nombre:  Administrador Vigencia:  1
Codigo:  2 Nombre:  Propietario Vigencia:  1
Codigo:  3 Nombre:  Inquilino Vigencia:  1
Codigo:  4 Nombre:  Visitante Vigencia:  1
Codigo:  6 Nombre:  Conserje Vigencia:  1
Codigo:  8 Nombre:  Recepcionista Vigencia:  1
Total de registros:  6
La conexión ha finalizado
PS F:\CursoPython>
```

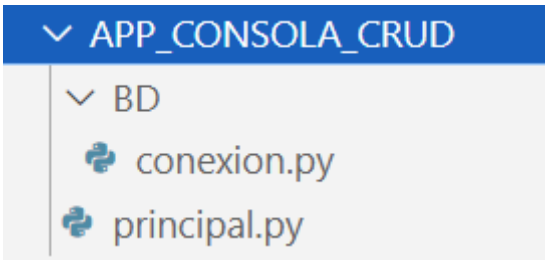
El registro número 7 no está, nos confirmamos de nuevo que el registro número 7 se ha eliminado con éxito.

47.- CRUD con Python & MySQL en Aplicación de Consola: Creación de Conexión

Para este nuevo proyecto vamos a crear una nueva carpeta llamada APP_CONSOLA_CRUD.

Dentro de esta dos carpetas una llamada __pycache__ y otra llamada DB

A continuación vamos a crear dos ficheros funciones.py y principal.py.



Vamos a crear una base de datos llamada universidad y la tabla curso con la siguiente estructura y datos:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	**Codigo** 🔑	varchar(6)	utf8mb4_general_ci		No	*Ninguna*
2	**Nombre**	varchar(50)	utf8mb4_general_ci		Sí	*NULL*
3	**Creditos**	int(2)			Sí	*NULL*

Codigo	Nombre	Creditos
103156	Programación Intermedia	5
365491	Matemática Discreta	4
786401	Teoría de Sistemas	4
872914	Física General	5
910212	Introducción a la Ingeniería	4

Este será el código del fichero conexión.py

```
1   import mysql.connector
2   from mysql.connector import Error
3
4   class DAO(): # Data Access Object
5       def __init__(self):
6           try:
7               self.conexion=mysql.connector.connect(
8                   host='localhost',
```

```
            port=3306,
            user='root',
            password='',
            db='universidad'
        )
    except Error as ex:
        print("Error al intentar la conexión: {0}".format(ex))

def ListarCursos(self):
    if self.conexion.is_connected():
        try:
            cursor=self.conexion.cursor()
            cursor.execute("SELECT * FROM curso ORDER BY nombre ASC")
            resultados=cursor.fetchall()
            return resultados
        except Error as ex:
            print("Error al intentar la conexión: {0}".format(ex))
```

Ahora nos vamos al fichero principal.py

```
def menuPrincipal():
    print("=== MENÚ PRINCIPAL ===")
    print("1.- Listar cursos")
    print("2.- Registrar curso")
    print("3.- Actualizar curso")
    print("4.- Eliminar curso")
    print("5.- Salir")
    print("===")
    opcion = int(input("Seleccione una opción: "))

    if opcion < 1 or opcion > 5:
        print("Opción incorrecta, ingrese nuevamente...")
    elif opcion == 5:
        print("¡Gracias por usar este sistema!")
    else:
        ejecutarOpcion(opcion)

def ejecutarOpcion(opcion):
    print(opcion)

menuPrincipal()
```

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python
/Python311/python.exe f:/CursoPython/APP_CONSOLA_CRUD/principal.p
```

```
y
=== MENÚ PRINCIPAL ===
1.- Listar cursos
2.- Registrar curso
3.- Actualizar curso
4.- Eliminar curso
5.- Salir
===
Seleccione una opción:
```

Vamos a modificar el código para que el menú se repita tantas veces como el usuario desee.

```python
def menuPrincipal():
    continuar = True
    while continuar:
        print("=== MENÚ PRINCIPAL ===")
        print("1.- Listar cursos")
        print("2.- Registrar curso")
        print("3.- Actualizar curso")
        print("4.- Eliminar curso")
        print("5.- Salir")
        print("===")
        opcion = int(input("Seleccione una opción: "))

        if opcion < 1 or opcion > 5:
            print("Opción incorrecta, ingrese nuevamente...")
        elif opcion == 5:
            print("¡Gracias por usar este sistema!")
            continuar = False
        else:
            ejecutarOpcion(opcion)

def ejecutarOpcion(opcion):
    print(opcion)

menuPrincipal()
```

Vamos a ejecutar:

Hasta que no pones la opción 5 para salir del bucle y finalizar.

```python
import os
def menuPrincipal():
    continuar = True
    while continuar:
        print("=== MENÚ PRINCIPAL ===")
```

```
 6          print("1.- Listar cursos")
 7          print("2.- Registrar curso")
 8          print("3.- Actualizar curso")
 9          print("4.- Eliminar curso")
10          print("5.- Salir")
11          print("===")
12          opcion = int(input("Seleccione una opción: "))
13          os.system("cls")
```

Si en la línea 1 importamos la librería os y después de la sentencia input en la línea siguiente escribimos **os.system("cls")** nos irá borrando la pantalla.

Para que puede interpretar la instrucción cls del sistema operativo de Windows.

Resumimos todo el código para que puedas comprobar y realizar pequeñas modificaciones:

```
import os
def menuPrincipal():
    continuar = True
    while continuar:
        opcionCorrecta = False
        while(not opcionCorrecta):
            print("=== MENÚ PRINCIPAL ===")
            print("1.- Listar cursos")
            print("2.- Registrar curso")
            print("3.- Actualizar curso")
            print("4.- Eliminar curso")
            print("5.- Salir")
            print("===")
            opcion = int(input("Seleccione una opción: "))
            os.system("cls")
            if opcion < 1 or opcion > 5:
                print("Opción incorrecta, ingrese nuevamente...")
            elif opcion == 5:
                continuar = False
                print("¡Gracias por usar este sistema!")
                break
            else:
                opcionCorrecta=True
                ejecutarOpcion(opcion)

def ejecutarOpcion(opcion):
    print(opcion)

menuPrincipal()
```

48.- CRUD con Python & MySQL en Aplicación de Consola: Lectura de Datos

Siguiente el ejercicio del capítulo anterior.

He modificado la presentación del menú del fichero principal.py

```
7           print("================= MENÚ PRINCIPAL =================")
8           print("                  1.- Listar cursos")
9           print("                  2.- Registrar curso")
10          print("                  3.- Actualizar curso")
11          print("                  4.- Eliminar curso")
12          print("                  5.- Salir")
13          print("==================================================")
```

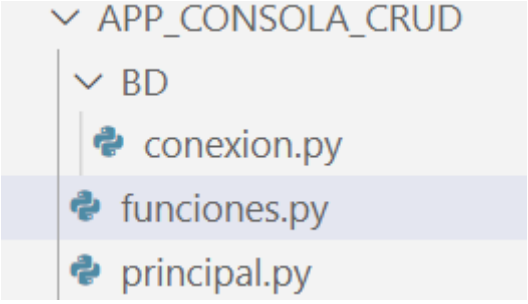
Seguimos con el documento principal.py.

```
from BD.conexion import DAO
import os
def menuPrincipal():
    continuar = True
    while continuar:
        opcionCorrecta = False
        while(not opcionCorrecta):
            print("================= MENÚ PRINCIPAL =================")
            print("                  1.- Listar cursos")
            print("                  2.- Registrar curso")
            print("                  3.- Actualizar curso")
            print("                  4.- Eliminar curso")
            print("                  5.- Salir")
            print("==================================================")
            opcion = int(input("Seleccione una opción: "))
            os.system("cls")
            if opcion < 1 or opcion > 5:
                print("Opción incorrecta, ingrese nuevamente...")
            elif opcion == 5:
                continuar = False
                print("¡Gracias por usar este sistema!")
                break
            else:
                opcionCorrecta=True
                ejecutarOpcion(opcion)

def ejecutarOpcion(opcion):
    dao = DAO()
    if opcion == 1:
        try:
            cursos = dao.ListarCursos()
            if len(cursos)>0:
                pass
                # a hacer algo
```

```
            else:
                print("No se encontraron cursos...")
        except:
            print("Ocurrió un error...")
    elif opcion ==2:
        print("Registro")
    elif opcion == 3:
        print("Actualización")
    elif opcion == 4:
        print("Eliminación")
    else:
        print("Opción no válida...")

menuPrincipal()
```

Así tiene que quedar el código.

Ahora vamos a crear otro fichero llamado funciones.py.



Vamos al fichero funciones.py.

```
1  def ListarCursos(cursos):
2      print("Cursos: ")
3      contador = 1
4      for cur in cursos:
5          datos = "{0}. Código: {1} | Nombre: {2} ({3} créditos)"
6          print(datos.format(contador, cur[0], cur[1], cur[2]))
7          contador = contador + 1
8      print(" ")
```

Ahora volvemos al fichero principal.py.

```
from BD.conexion import DAO
import funciones
import os
def menuPrincipal():
    continuar = True
    while continuar:
        opcionCorrecta = False
        while(not opcionCorrecta):
            print("================= MENÚ PRINCIPAL =================")
```

Para poder acceder a las funciones del fichero funciones.py

```
            print("                1.- Listar cursos")
            print("                2.- Registrar curso")
            print("                3.- Actualizar curso")
            print("                4.- Eliminar curso")
            print("                5.- Salir")
            print("================================================")
            opcion = int(input("Seleccione una opción: "))
            os.system("cls")
            if opcion < 1 or opcion > 5:
                print("Opción incorrecta, ingrese nuevamente...")
            elif opcion == 5:
                continuar = False
                print("¡Gracias por usar este sistema!")
                break
            else:
                opcionCorrecta=True
                ejecutarOpcion(opcion)

def ejecutarOpcion(opcion):
    dao = DAO()
    if opcion == 1:
        try:
            cursos = dao.ListarCursos()
            if len(cursos)>0:
                funciones.ListarCursos(cursos)
            else:
                print("No se encontraron cursos...")
        except:
            print("Ocurrió un error...")
    elif opcion ==2:
        print("Registro")
    elif opcion == 3:
        print("Actualización")
    elif opcion == 4:
        print("Eliminación")
    else:
        print("Opción no válida...")

menuPrincipal()
```

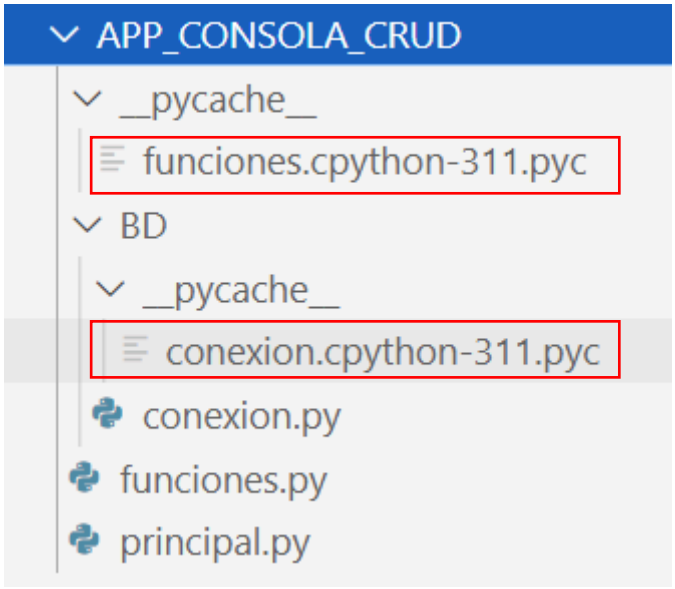
Llamamos a la función ListarCursos pasándole el parámetro cursos que está en el fichero funciones.py

Vamos a ejecutar y contestar con la opción 1.

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
1. Código: 872914 | Nombre: Física General (5 créditos)
2. Código: 910212 | Nombre: Introducción a la Ingeniería (4 créditos)
3. Código: 365491 | Nombre: Matemática Discreta (4 créditos)
4. Código: 103156 | Nombre: Programación Intermedia (5 créditos)
5. Código: 786401 | Nombre: Teoría de Sistemas (4 créditos)
```

```
================= MENÚ PRINCIPAL =================
                  1.- Listar cursos
                  2.- Registrar curso
                  3.- Actualizar curso
                  4.- Eliminar curso
                  5.- Salir
==================================================
Seleccione una opción:
```



Si vemos nuestro proyecto veremos que se han creado dos carpetas __pycahe__ es el caché de Python que contiene bytecode de Python 3 complilado y lista para ser ejecutado.

Dentro de las carpetas tenemos los archivos compilado de Python, simplemente está para el mejor funcionamiento del programa con Python.

No tenemos que tocarlos ni eliminarlos, solo tienen que estar.

Para verificar si son estos los datos lo vamos a verificar desde la base de datos:

Codigo	Nombre	Creditos
103156	Programación Intermedia	5
365491	Matemática Discreta	4
786401	Teoría de Sistemas	4
872914	Física General	5
910212	Introducción a la Ingeniería	4

Ahora vamos a modificar el fichero funciones.py.

```python
def ListarCursos(cursos):
    print("\nCursos: \n")
    contador = 1
    for cur in cursos:
        datos = "{0}. Código: {1} | Nombre: {2} ({3} créditos)"
        print(datos.format(contador, cur[0], cur[1], cur[2]))
        contador = contador + 1
    print(" ")
```

Vamos a ejecutar y contestar por la opción 1 desde el fichero principal.py.

```
Cursos:

1. Código: 872914 | Nombre: Física General (5 créditos)
2. Código: 910212 | Nombre: Introducción a la Ingeniería (4 créditos)
3. Código: 365491 | Nombre: Matemática Discreta (4 créditos)
4. Código: 103156 | Nombre: Programación Intermedia (5 créditos)
5. Código: 786401 | Nombre: Teoría de Sistemas (4 créditos)
```

49.- CRUD con Python & MySQL en Aplicación de Consola: Registro (INSERT INTO)

En el fichero principal.py realizaremos las siguientes modificaciones:

```
39      elif opcion ==2:
40          curso = None
41          try:
42              dao.registrarCurso(curso)
43          except:
44              print("Ocurrió un error...")
```

Ahora nos vamos al fichero conexión.py

```
1  import mysql.connector
2  from mysql.connector import Error
3
4  class DAO(): # Data Access Object
5      def __init__(self):
6          try:
7              self.conexion=mysql.connector.connect(
8                  host='localhost',
9                  port=3306,
10                 user='root',
11                 password='',
12                 db='universidad'
13             )
14         except Error as ex:
15             print("Error al intentar la conexión: {0}".format(ex))
16
17     def ListarCursos(self):
18         if self.conexion.is_connected():
19             try:
20                 cursor=self.conexion.cursor()
21                 cursor.execute("SELECT * FROM curso ORDER BY nombre ASC")
22                 resultados=cursor.fetchall()
23                 return resultados
24             except Error as ex:
25                 print("Error al intentar la conexión: {0}".format(ex))
28     def registrarCurso(self, curso):
29         if self.conexion.is_connected:
30             try:
31                 cursor=self.conexion.cursor()
32                 sql = "INSERT INTO curso (Codigo, Nombre, Creditos) VALUES ('{0}','{1}', {2})"
33                 cursor.execute(sql.format(curso[0], curso[1], curso[2]))
34                 self.conexion.commit()
35                 print("¡Curso registrado!\n")
36             except Error as ex:
37                 print("Error al intentar la conexión: {0}".format(ex))
```

Hay que añadir la parte remarcada en color rojo.

Volvemos al fichero funciones.py

```
1   def ListarCursos(cursos):
2       print("\nCursos: \n")
3       contador = 1
4       for cur in cursos:
5           datos = "{0}. Código: {1} | Nombre: {2} ({3} créditos)"
6           print(datos.format(contador, cur[0], cur[1], cur[2]))
7           contador = contador + 1
8       print(" ")
9
10  def perdirDatosRegistro():
11      codigo=input("Ingreso código: ")
12      nombre=input("Ingrese nombre: ")
13      creditos=int(input("Ingrese créditos:"))
14
15      curso=(codigo,nombre,creditos)
16      return curso
```

Agregar el código que está enmarcado en rojo.

Ahora volvemos al fichero principal.py

```
39      elif opcion ==2:
40          curso = funciones.perdirDatosRegistro()
41          try:
42              dao.registrarCurso(curso)
43          except:
44              print("Ocurrió un error...")
```

Modifica la siguiente línea.

Vamos a ejecutar y contestaremos por la opción 2.

```
Ingreso código: 765321
Ingrese nombre: Redes y Conectividad
Ingrese créditos:6
¡Curso registrado!
```

Vamos a seleccionar ahora la opción 1.

```
1. Código: 872914 | Nombre: Física General (5 créditos)
2. Código: 910212 | Nombre: Introducción a la Ingeniería (4 créditos)
3. Código: 365491 | Nombre: Matemática Discreta (4 créditos)
4. Código: 103156 | Nombre: Programación Intermedia (5 créditos)
5. Código: 765321 | Nombre: Redes y Conectividad (6 créditos)
6. Código: 786401 | Nombre: Teoría de Sistemas (4 créditos)
```

Del fichero funciones.py vamos a modificar la función pedirDatosRegistro().

```
10  def perdirDatosRegistro():
11      codigoCorrecto = False
12      while(not codigoCorrecto):
13          codigo=input("Ingreso código: ")
14          if len(codigo)==6:
15              codigoCorrecto = True
16          else:
17              print("Código incorrecto: Debe tener 6 dígitos.")
18
19      nombre=input("Ingrese nombre: ")
20      creditos=int(input("Ingrese créditos:"))
21
22      curso=(codigo,nombre,creditos)
23      return curso
```

Vamos a ejecutar contestaremos por la opción 2, pondremos un codigo de 9 números, tiene que decir “Código incorrecto: Debe tener 6 dígitos”, ahora volvemos a poner un codigo, esta vez de 4 números nos tendrá que volver a mostrar el mismo mensaje.

Ahora vamos a introducir un código de 6 números, si todo va bien nos preguntará por nombre: a lo que contestaremos por Biología y en créditos por 4.

Ahora vamos a seleccionar la opción 1 para comprobar si se a agregado el registro.

```
Cursos:

1. Código: 678012 | Nombre: Biología (4 créditos)
2. Código: 872914 | Nombre: Física General (5 créditos)
3. Código: 910212 | Nombre: Introducción a la Ingeniería (4 créditos)
4. Código: 365491 | Nombre: Matemática Discreta (4 créditos)
5. Código: 103156 | Nombre: Programación Intermedia (5 créditos)
6. Código: 765321 | Nombre: Redes y Conectividad (6 créditos)
7. Código: 786401 | Nombre: Teoría de Sistemas (4 créditos)
```

Ahora queremos validar los créditos:

```
10  def perdirDatosRegistro():
11      codigoCorrecto = False
12      while(not codigoCorrecto):
13          codigo=input("Ingreso código: ")
14          if len(codigo)==6:
15              codigoCorrecto = True
16          else:
17              print("Código incorrecto: Debe tener 6 dígitos.")
18
19      nombre=input("Ingrese nombre: ")
20
```

```
21      creditosCorrectos = False
22      while (not creditosCorrectos):
23          creditos=input("Ingrese créditos:")
24          if creditos.isnumeric():
25              creditosCorrectos = True
26              creditos = int(creditos)
27          else:
28              print("Créditos incorrectos: Debe ser un número únicamente.")
29
30
31      curso=(codigo,nombre,creditos)
32      return curso
```

Vamos a comprobarlo.

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

Ingreso código: 567813
Ingrese nombre: Investigación
Ingrese créditos:gato
Créditos incorrectos: Debe ser un número únicamente.
Ingrese créditos:4
¡Curso registrado!

================= MENÚ PRINCIPAL =================
                  1.- Listar cursos
                  2.- Registrar curso
                  3.- Actualizar curso
                  4.- Eliminar curso
                  5.- Salir
==================================================
Seleccione una opción:
```

Vamos a seleccionar la opción 1 para ver si se a añadido el registro.

```
Cursos:

1. Código: 678012 | Nombre: Biología (4 créditos)
2. Código: 872914 | Nombre: Física General (5 créditos)
3. Código: 910212 | Nombre: Introducción a la Ingeniería (4 créditos)
4. Código: 567813 | Nombre: Investigación (4 créditos)
5. Código: 365491 | Nombre: Matemática Discreta (4 créditos)
6. Código: 103156 | Nombre: Programación Intermedia (5 créditos)
7. Código: 765321 | Nombre: Redes y Conectividad (6 créditos)
8. Código: 786401 | Nombre: Teoría de Sistemas (4 créditos)
```

El registro se ha añadido correctamente.

También queremos controlar que el número introducido es positivo.

```
10  def perdirDatosRegistro():
11      codigoCorrecto = False
12      while(not codigoCorrecto):
13          codigo=input("Ingreso código: ")
14          if len(codigo)==6:
15              codigoCorrecto = True
16          else:
17              print("Código incorrecto: Debe tener 6 dígitos.")
18
19      nombre=input("Ingrese nombre: ")
20
21      creditosCorrectos = False
22      while (not creditosCorrectos):
23          creditos=input("Ingrese créditos:")
24          if creditos.isnumeric():
25              if(int(creditos)>0):
26                  creditosCorrectos = True
27                  creditos = int(creditos)
28              else:
29                  print("Los créditos deben ser mayor a 0")
30          else:
31              print("Créditos incorrectos: Debe ser un número únicamente.")
```

Controlamos que los créditos sean números y además positivos.

Vamos a probarlo:

```
Ingreso código: 567001
Ingrese nombre: Disedño gráfico
Ingrese créditos:-6
Créditos incorrectos: Debe ser un número únicamente.
Ingrese créditos:0
Los créditos deben ser mayor a 0
Ingrese créditos:4
¡Curso registrado!
```

Vamos a seleccionar la opción 1:

```
Cursos:

1. Código: 678012 | Nombre: Biología (4 créditos)
2. Código: 567001 | Nombre: Disedño gráfico (4 créditos)
3. Código: 872914 | Nombre: Física General (5 créditos)
4. Código: 910212 | Nombre: Introducción a la Ingeniería (4 créditos)
5. Código: 567813 | Nombre: Investigación (4 créditos)
6. Código: 365491 | Nombre: Matemática Discreta (4 créditos)
7. Código: 103156 | Nombre: Programación Intermedia (5 créditos)
8. Código: 765321 | Nombre: Redes y Conectividad (6 créditos)
9. Código: 786401 | Nombre: Teoría de Sistemas (4 créditos)
```

50.- CRUD con Python & MySQL en Aplicaciones de Consola: Eliminación de registros

Vamos al fichero principal.py para realizar la siguiente modificación.

```
47      elif opcion == 4:
48          try:
49              cursos=dao.ListarCursos()
50              if len(cursos)>0:
51                  codigoEliminar=None
52                  if not(codigoEliminar ==""):
53                      dao.eliminarCurso(codigoEliminar)
54                  else:
55                      print("Código de curso no encontrado...\n")
56              else:
57                  print("No se encontraron cursos...")
58          except:
59              print("Ocurrió un error...")
60      else:
61          print("Opción no válida...")
```

Este código es para eliminar un registro.

Ahora nos vamos al fichero conexión.py y vamos a crear el método eliminar.

```
39      def eliminarCurso(self, codigoCursoEliminar):
40          if self.conexion.is_connected():
41              try:
42                  cursor=self.conexion.cursor()
43                  sql = "DELETE FROM curso WHERE codigo = '{0}'"
44                  cursor.execute(sql.format(codigoCursoEliminar))
45                  self.conexion.commit()
46                  print("¡Curso eliminado!\n")
47              except Error as ex:
48                  print("Error al intentar la conexión: {0}".format(ex))
```

Ahora vamos al fichero funciones.py para crear una nueva función.

```
37  def pedirDatosEliminacion(cursos):
38      ListarCursos(cursos)
39      existeCodigo=False
40      codigoEliminar=input("Ingrese el código del curso a eliminar: ")
41      for cur in cursos:
42          if cur[0] == codigoEliminar:
43              existeCodigo=True
44              break
45
46      if not existeCodigo:
47          codigoEliminar=""
48
49      return codigoEliminar
```

Volvemos al fichero principal.py

```
47      elif opcion == 4:
48          try:
49              cursos=dao.ListarCursos()
50              if len(cursos)>0:
51                  codigoEliminar=funciones.pedirDatosEliminacion(cursos)
52                  if not(codigoEliminar ==""):
53                      dao.eliminarCurso(codigoEliminar)
54                  else:
55                      print("Código de curso no encontrado...\n")
56              else:
57                  print("No se encontraron cursos...")
58          except:
59              print("Ocurrió un error...")
60      else:
61          print("Opción no válida...")
```

Modifica la línea seleccionada.

Vamos a ejecutar:

```
Cursos:

1. Código: 678012 | Nombre: Biología (4 créditos)
2. Código: 567001 | Nombre: Disedño gráfico (4 créditos)
3. Código: 872914 | Nombre: Física General (5 créditos)
4. Código: 910212 | Nombre: Introducción a la Ingeniería (4 créditos)
5. Código: 567813 | Nombre: Investigación (4 créditos)
6. Código: 365491 | Nombre: Matemática Discreta (4 créditos)
7. Código: 103156 | Nombre: Programación Intermedia (5 créditos)
8. Código: 765321 | Nombre: Redes y Conectividad (6 créditos)
9. Código: 786401 | Nombre: Teoría de Sistemas (4 créditos)

Ingrese el código del curso a eliminar:
```

Se nos muestran todos los cursos para poder escribir el código del registro que deseamos eliminar.

Hemos puesto el código 111111.

```
Ingrese el código del curso a eliminar: 111111
Código de curso no encontrado...
```

Ahora vamos a eliminar Biología.

```
Ingrese el código del curso a eliminar: 678012
¡Curso eliminado!
```

Ahora vamos a seleccionar el 1 para ver si se ha eliminado este registro.

```
1. Código: 567001 | Nombre: Disedño gráfico (4 créditos)
2. Código: 872914 | Nombre: Física General (5 créditos)
3. Código: 910212 | Nombre: Introducción a la Ingeniería (4 créditos)
4. Código: 567813 | Nombre: Investigación (4 créditos)
5. Código: 365491 | Nombre: Matemática Discreta (4 créditos)
6. Código: 103156 | Nombre: Programación Intermedia (5 créditos)
7. Código: 765321 | Nombre: Redes y Conectividad (6 créditos)
8. Código: 786401 | Nombre: Teoría de Sistemas (4 créditos)
```

El registro de Biología ya no está.

51.- CRUD con Python & MySQL en Aplicación de Consola: Actualización de Registros

Vamos a fichero principal.py y modificamos esta parte del código:

```
45      elif opcion == 3:
46          try:
47              cursos=dao.ListarCursos()
48              if len(cursos)>0:
49                  curso = None
50                  if curso:
51                      dao.actualizaCurso(curso)
52                  else:
53                      print("Código de curso a actualizar no encontrado...\n")
54              else:
55                  print("No se encontraron cursos...")
56          except:
57              print("Ocurrió un error...")
```

Ahora vamos al fichero conexión.py

```
39      def actualizarCurso(self, curso):
40          if self.conexion.is_connected():
41            try:
42                cursor=self.conexion.cursor()
43                sql = "UPDATE curso SET nombre = '{0}', creditos = {1} WHERE codigo = '{2}'"
44                cursor.execute(sql.format(curso[1], curso[2], curso[0]))
45                self.conexion.commit()
46                print("¡Curso actualizado!\n")
47            except Error as ex:
48                print("Error al intentar la conexión: {0}".format(ex))
```

Agregamos el método actualizarCurso(self, curso).

Vamos al fichero funciones.py creamos la función pedirDatosActualizacion(cursos)

```
37  def pedirDatosActualizacion(cursos):
38      ListarCursos(cursos)
39      existeCodigo=False
40      codigoEditar=input("Ingrese el código del curso a editar: ")
41      for cur in cursos:
42          if cur[0] == codigoEditar:
43              existeCodigo=True
44              break
45
46      if existeCodigo:
47          nombre=input("Ingrese nombre a modificar: ")
48
49          creditosCorrectos = False
50          while (not creditosCorrectos):
51              creditos=input("Ingrese créditos a modificar:")
52              if creditos.isnumeric():
53                  if(int(creditos)>0):
54                      creditosCorrectos = True
55                      creditos = int(creditos)
```

```
56                else:
57                    print("Los créditos deben ser mayor a 0")
58            else:
59                print("Créditos incorrectos: Debe ser un número únicamente.")
60        else:
61            curso = (codigoEditar, nombre, creditos)
62    else:
63        curso = None
64    return curso
```

Vamos al fichero principal.py para modificar lo siguiente.

```
45    elif opcion == 3:
46        try:
47            cursos=dao.ListarCursos()
48            if len(cursos)>0:
49                curso = funciones.pedirDatosActualizacion(cursos)
50                if curso:
51                    dao.actualizarCurso(curso)
52                else:
53                    print("Código de curso a actualizar no encontrado...\n")
54            else:
55                print("No se encontraron cursos...")
56        except:
57            print("Ocurrió un error...")
```

Vamos a ejecutar:

Seleccionamos la opción 3.

```
Cursos:

1. Código: 567001 | Nombre: Disedño gráfico (4 créditos)
2. Código: 872914 | Nombre: Física General (5 créditos)
3. Código: 910212 | Nombre: Introducción a la Ingeniería (4 créditos)
4. Código: 567813 | Nombre: Investigación (4 créditos)
5. Código: 365491 | Nombre: Matemática Discreta (4 créditos)
6. Código: 103156 | Nombre: Programación Intermedia (5 créditos)
7. Código: 765321 | Nombre: Redes y Conectividad (6 créditos)
8. Código: 786401 | Nombre: Teoría de Sistemas (4 créditos)

Ingrese el código del curso a editar:
```

Vamos a ingresar un código que no existe.

```
Ingrese el código del curso a editar: 222333
Código de curso a actualizar no encontrado...
```

Seleccionamos de nuevo la opción 3 para introducir un código existente.

Vamos a cambiar los datos del curso Física General.

Como nombre será Ciencias Físicas y como Crédito 6.

```
Ingrese el código del curso a editar: 872914
Ingrese nombre a modificar: Ciencias Físicas
Ingrese créditos a modificar:6
¡Curso actualizado!
```

Vamos a realizar otra actualización.

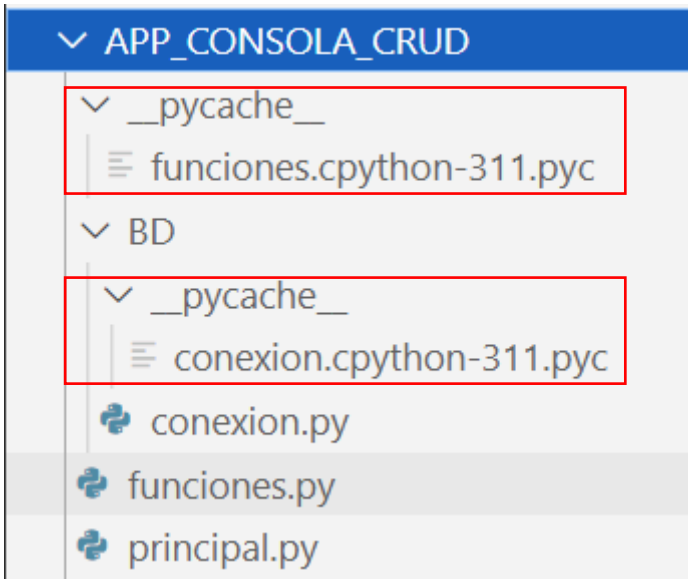
```
Ingrese el código del curso a editar: 567813
Ingrese nombre a modificar: Investigación y Deducción
Ingrese créditos a modificar:5
¡Curso actualizado!
```

Vamos a seleccionar la opción 1 para mostrar todos los registros:

```
1. Código: 872914 | Nombre: Ciencias Físicas (6 créditos)
2. Código: 567001 | Nombre: Disedño gráfico (4 créditos)
3. Código: 910212 | Nombre: Introducción a la Ingeniería (4 créditos)
4. Código: 567813 | Nombre: Investigación y Deducción (5 créditos)
5. Código: 365491 | Nombre: Matemática Discreta (4 créditos)
6. Código: 103156 | Nombre: Programación Intermedia (5 créditos)
7. Código: 765321 | Nombre: Redes y Conectividad (6 créditos)
8. Código: 786401 | Nombre: Teoría de Sistemas (4 créditos)
```

Resumen:

Esta es la estructura que tiene que tener nuestro proyecto:



Las partes enmarcadas se generan automáticamente para una mayor eficiencia del programa.

Código final de conexión.py:

```
import mysql.connector
from mysql.connector import Error

class DAO(): # Data Access Object
    def __init__(self):
        try:
            self.conexion=mysql.connector.connect(
                host='localhost',
                port=3306,
                user='root',
                password='',
                db='universidad'
            )
        except Error as ex:
            print("Error al intentar la conexión: {0}".format(ex))

    def ListarCursos(self):
        if self.conexion.is_connected():
            try:
                cursor=self.conexion.cursor()
                cursor.execute("SELECT * FROM curso ORDER BY nombre ASC")
                resultados=cursor.fetchall()
                return resultados
            except Error as ex:
                print("Error al intentar la conexión: {0}".format(ex))


    def registrarCurso(self, curso):
        if self.conexion.is_connected():
            try:
                cursor=self.conexion.cursor()
                sql = "INSERT INTO curso (Codigo, Nombre, Creditos)
VALUES ('{0}','{1}', {2})"
                cursor.execute(sql.format(curso[0], curso[1], curso[2]))
                self.conexion.commit()
                print("¡Curso registrado!\n")
            except Error as ex:
                 print("Error al intentar la conexión: {0}".format(ex))

    def actualizarCurso(self, curso):
         if self.conexion.is_connected():
            try:
                cursor=self.conexion.cursor()
                sql = "UPDATE curso SET nombre = '{0}', creditos = {1}
WHERE codigo = '{2}'"
                cursor.execute(sql.format(curso[1], curso[2], curso[0]))
                self.conexion.commit()
```

```
                print("¡Curso actualizado!\n")
            except Error as ex:
                 print("Error al intentar la conexión: {0}".format(ex))

    def eliminarCurso(self, codigoCursoEliminar):
        if self.conexion.is_connected():
            try:
                cursor=self.conexion.cursor()
                sql = "DELETE FROM curso WHERE codigo = '{0}'"
                cursor.execute(sql.format(codigoCursoEliminar))
                self.conexion.commit()
                print("¡Curso eliminado!\n")
            except Error as ex:
                 print("Error al intentar la conexión: {0}".format(ex))
```

Codigo final de funciones.py:

```
def ListarCursos(cursos):
    print("\nCursos: \n")
    contador = 1
    for cur in cursos:
        datos = "{0}. Código: {1} | Nombre: {2} ({3} créditos)"
        print(datos.format(contador, cur[0], cur[1], cur[2]))
        contador = contador + 1
    print(" ")

def perdirDatosRegistro():
    codigoCorrecto = False
    while(not codigoCorrecto):
        codigo=input("Ingreso código: ")
        if len(codigo)==6:
            codigoCorrecto = True
        else:
            print("Código incorrecto: Debe tener 6 dígitos.")

    nombre=input("Ingrese nombre: ")

    creditosCorrectos = False
    while (not creditosCorrectos):
        creditos=input("Ingrese créditos:")
        if creditos.isnumeric():
            if(int(creditos)>0):
                creditosCorrectos = True
                creditos = int(creditos)
            else:
                print("Los créditos deben ser mayor a 0")
        else:
            print("Créditos incorrectos: Debe ser un número únicamente.")
```

```
    curso=(codigo,nombre,creditos)
    return curso

def pedirDatosActualizacion(cursos):
    ListarCursos(cursos)
    existeCodigo=False
    codigoEditar=input("Ingrese el código del curso a editar: ")
    for cur in cursos:
        if cur[0] == codigoEditar:
            existeCodigo=True
            break

    if existeCodigo:
        nombre=input("Ingrese nombre a modificar: ")

        creditosCorrectos = False
        while (not creditosCorrectos):
            creditos=input("Ingrese créditos a modificar:")
            if creditos.isnumeric():
                if(int(creditos)>0):
                    creditosCorrectos = True
                    creditos = int(creditos)
                else:
                    print("Los créditos deben ser mayor a 0")
            else:
                print("Créditos incorrectos: Debe ser un número
únicamente.")
        else:
            curso = (codigoEditar, nombre, creditos)
    else:
        curso = None
    return curso


def pedirDatosEliminacion(cursos):
    ListarCursos(cursos)
    existeCodigo=False
    codigoEliminar=input("Ingrese el código del curso a eliminar: ")
    for cur in cursos:
        if cur[0] == codigoEliminar:
            existeCodigo=True
            break

    if not existeCodigo:
        codigoEliminar=""

    return codigoEliminar
```

Código final de principal.py:

```
from BD.conexion import DAO
import funciones
import os
def menuPrincipal():
    continuar = True
    while continuar:
        opcionCorrecta = False
        while(not opcionCorrecta):
            print("================= MENÚ PRINCIPAL =================")
            print("                  1.- Listar cursos")
            print("                  2.- Registrar curso")
            print("                  3.- Actualizar curso")
            print("                  4.- Eliminar curso")
            print("                  5.- Salir")
            print("==================================================")
            opcion = int(input("Seleccione una opción: "))
            os.system("cls")
            if opcion < 1 or opcion > 5:
                print("Opción incorrecta, ingrese nuevamente...")
            elif opcion == 5:
                continuar = False
                print("¡Gracias por usar este sistema!")
                break
            else:
                opcionCorrecta=True
                ejecutarOpcion(opcion)

def ejecutarOpcion(opcion):
    dao = DAO()
    if opcion == 1:
        try:
            cursos = dao.ListarCursos()
            if len(cursos)>0:
                funciones.ListarCursos(cursos)
            else:
                print("No se encontraron cursos...")
        except:
            print("Ocurrió un error...")
    elif opcion == 2:
        curso = funciones.perdirDatosRegistro()
        try:
            dao.registrarCurso(curso)
        except:
            print("Ocurrió un error...")
    elif opcion == 3:
        try:
```

```
            cursos=dao.ListarCursos()
            if len(cursos)>0:
                curso = funciones.pedirDatosActualizacion(cursos)
                if curso:
                    dao.actualizarCurso(curso)
                else:
                    print("Código de curso a actualizar no
encontrado...\n")
            else:
                print("No se encontraron cursos...")
        except:
            print("Ocurrió un error...")
    elif opcion == 4:
        try:
            cursos=dao.ListarCursos()
            if len(cursos)>0:
                codigoEliminar=funciones.pedirDatosEliminacion(cursos)
                if not(codigoEliminar ==""):
                    dao.eliminarCurso(codigoEliminar)
                else:
                    print("Código de curso no encontrado...\n")
            else:
                print("No se encontraron cursos...")
        except:
            print("Ocurrió un error...")
    else:
        print("Opción no válida...")

menuPrincipal()
```

52.- Funciones Lambda en Python ¿Cómo funcionan?

Vamos con un nuevo proyecto que la llamaremos lambda.py

Las funciones lambda son funciones anónimas que sirven para abreviar o refundir una función normal en una expresión mucho más sencilla de leer.

Partiendo del siguiente ejemplo:

```
1    def sumar(n1, n2):
2        return n1 + n2
3
4    print(sumar(12,15))
```

Con el siguiente resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/
Programs/Python/Python311/python.exe f:/CursoPytho
n/lambda.py
27
PS F:\CursoPython>
```

```
1    sumar = lambda n1, n2 : n1 + n2
2
3    print(sumar(5,7))
```

Con el siguiente resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/
Programs/Python/Python311/python.exe f:/CursoPytho
n/lambda.py
12
PS F:\CursoPython>
```

Si una función consta de más de una línea, no podremos reconvertirla en una función lambda.

Toda función lambda se puede convertir a una función normal, no viceversa.

Las funciones lambda son funciones anónimas que sirven par abreviar o resumir una función normal, para convertirla en una expresión más simple.

```
5    resultado = sumar(12,7)
```

El resultado de una función también se la podemos asignar a una variable.

```
6    print(resultado)
```

A continuación podremos imprimir el valor de la variable resultado.

Vamos a ver otro ejemplo:

```
1    elevarCuadrado = lambda numero: numero * numero
2    print(elevarCuadrado(5))
```

Este será el resultado:

53.- Función Filter en Python | Para que sirve y cómo funciona

Es una de las funciones de orden superior (programación funcional).

Lo que hace es verificar que los elementos de una lista cumplan una determinada condición, devolviendo un objeto iterable (iterador) con los elementos que cumplieron esta condición predeterminada.

```
1   edades=[12, 11, 24, 36, 8, 6, 10, 41, 32, 58, 14, 50, 7]
2
3   # Queremos obtener las edades que son a partir de 18 años.
4
5   def mayorEdad(edad):
6       return edad >= 18
7
8   print(filter(mayorEdad, edades))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/pytho
n.exe f:/CursoPython/filter.py
<filter object at 0x00000172BE451900>
PS F:\CursoPython>
```

El resultado es un objeto de tipo Filter, muestra la dirección de memoria donde está alojado, pero esta información no se puede leer.

```
10  edadesMayoresEdad=list(filter(mayorEdad, edades))
11  print(edadesMayoresEdad)
```

Agregamos las siguientes líneas el resultado de los valores obtenidos los pasamos a una variable llamada edadesMayoresEdad de tipo lista, este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

n.exe f:/CursoPython/filter.py
<filter object at 0x00000172BE451900>
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
n.exe f:/CursoPython/filter.py
<filter object at 0x000001F3FC061900>
[24, 36, 41, 32, 58, 50]
PS F:\CursoPython>
```

Ejercicio práctico:

Vamos a adaptarlo a una función lambda.

```
1   edades=[12, 11, 24, 36, 8, 6, 10, 41, 32, 58, 14, 50, 7]
2
```

```python
# Queremos obtener las edades que son a partir de 18 años.


edadesMayoresEdad=list(filter(lambda edad: edad>18, edades))

print(edadesMayoresEdad)
```

Este será el resultado:

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/filter.py
[24, 36, 41, 32, 58, 50]
PS F:\CursoPython>
```

```python
class Persona:
    def __init__(self, nombre, edad):
        self.nombre= nombre
        self.edad=edad

    def __str__(self):
        return "{0} tiene {1} años.".format(self.nombre, self.edad)

personas=[
    Persona("Alberto", 32),
    Persona("Ana", 16),
    Persona("Andy", 27),
    Persona("Jesús", 25),
    Persona("Cecilia", 19),
    Persona("Laura", 30),
]

personasMayoresEdad=list(filter(lambda per:per.edad>=18, personas))
print(personasMayoresEdad)
```

Si ejecutamos:

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/filter.py
[<__main__.Persona object at 0x0000025CA68C7350>, <__main__.Persona object
at 0x0000025CA68C7850>, <__main__.Persona object at 0x0000025CA68C7890>, <_
_main__.Persona object at 0x0000025CA68C78D0>, <__main__.Persona object at
0x0000025CA68C7910>]
PS F:\CursoPython>
```

Tenemos una lista de objetos de tipo Persona

Vamos a borrara la línea 19 para escribir los siguiente:

Agregamos las siguientes líneas:

```
20    for per in personasMayoresEdad:
21        print(per)
```

Vamos a ejecutar:

Aquí nos muestra los que son mayores de edad, a partir de 18 años.

54.- Función Map de Python ¿Para que sirve y cómo funciona?

Vamos a crear un nuevo archivo llamado map.py

Map aplica una función a cada elemento de una lista iterable, devolviendo otra lista.

```
1   def elevarCuadrado(num):
2       return pow(num, 2)
3
4   # numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
5   numeros = list(range(1,11))
6   print(numeros)
7
8   numerosElevados=list(map(elevarCuadrado, numeros))
9   print(numerosElevados)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/map.py
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
PS F:\CursoPython>
```

55.- Sets (Conjuntos) y sus métodos | ¿Para qué sirven los Sets en Python?

```
1    canasta={'manzana','platano','pera','manzana','naranja','pera'}
2    print(canasta)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/sets.py
{'manzana', 'platano', 'pera', 'naranja'}
PS F:\CursoPython>
```

Nos muestra los elementos per los que se repiten solo se muestra uno.

Los Sets son colecciones desordenadas de objetos únicos.

Vamos a ver otro ejemplo con números:

```
1    numeros={1,3,5,8,3,4,12,1}
2    print(numeros)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/sets.py
{1, 3, 4, 5, 8, 12}
PS F:\CursoPython>
```

Los elementos repetidos no están.

```
1    a=set('abracadabra')
2    print(a)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/sets.py
{'b', 'd', 'a', 'r', 'c'}
PS F:\CursoPython>
```

Algunos set de denominan mutables, se pueden añadir nuevos elementos.

```
3    a.add('g')
4    print(a)
```

Agregamos las siguientes líneas, este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/sets.py
{'d', 'c', 'a', 'b', 'r'}
{'d', 'c', 'a', 'g', 'b', 'r'}
PS F:\CursoPython>
```

Hemos agregado un elemento.

Si intentamos añadir un elemento que ya tiene, este no se agregará, no puede estar duplicado.

Hay sets de tipo inmutables (no se pueden añadir nuevos elementos).

```
1    b=frozenset('perro',)
2    print(b)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/sets.py
frozenset({'e', 'p', 'o', 'r'})
PS F:\CursoPython>
```

Agregamos las siguientes líneas.

```
4    b.add('k')
5    print(b)
```

Este será el resultado:

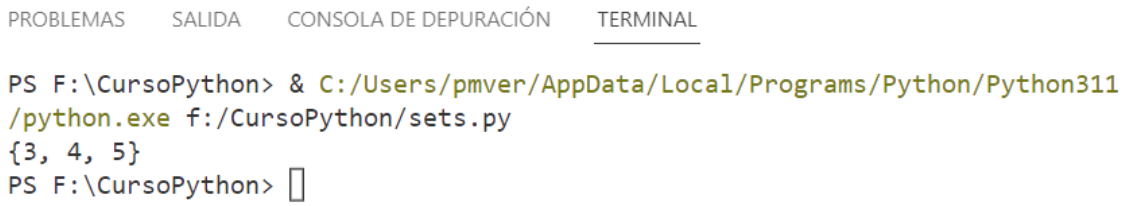
```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/sets.py
frozenset({'r', 'p', 'e', 'o'})
Traceback (most recent call last):
  File "f:\CursoPython\sets.py", line 4, in <module>
    b.add('k')
    ^^^^^
AttributeError: 'frozenset' object has no attribute 'add'
PS F:\CursoPython>
```

Nos mostrará un error diciendo que este objeto no tiene el atributo add.

Intersecciones:

```
1    miSet={1,2,3,4,5}.intersection({3,4,5,6})
2    print(miSet)
```

Este será el resultado:



Vemos solo los elementos 3, 4 y 5 son los elementos que se son comunes.

También se puede representar de la siguiente forma:

```
1    miSet={1,2,3,4,5} & ({3,4,5,6})
```

El resultado será el mismo.

Uniones:

```
1    miSet={1, 2, 3, 4, 5}.union({3, 4, 5, 6})
2    print(miSet)
```

Este será el resultado:

Tenemos la unión de los dos elementos quitando los duplicados.

```
1    miSet={1, 2, 3, 4, 5}|({3, 4, 5, 6})
```

Esto hace exactamente lo mismo.

Diferente:

```
1    miSet={1, 2, 3, 4}.difference({2, 3, 5})
2    print(miSet)
```

Este será el resultado:

```
{1, 4}
PS F:\CursoPython>
```

Muestra los elementos que son diferentes.

```
1    miSet={1, 2, 3, 4}-({2, 3, 5})
```

Hace exactamente lo mismo.

Diferencia simétrica:

```
1    miSet={1, 2, 3, 4}.symmetric_difference({2, 3, 5})
2    print(miSet)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/sets.py
{1, 4, 5}
PS F:\CursoPython>
```

Son los elementos que no existen que no existen en el otro grupo de elementos.

El 1 y 4 no están en el segundo bloque de elementos y el 5 no está en el primer bloque de elementos.

Contiene:

```
1    miSet={1,2}.issuperset({1,2,3})
2    print(miSet)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/sets.py
False
PS F:\CursoPython>
```

En cambio.

```
1    miSet={1,2,3}.issuperset({1,2})
2    print(miSet)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL
```

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/sets.py
True
PS F:\CursoPython>
```

Le estamos diciendo si los elementos del primer bloque están en el segundo bloque será True, de lo contrario será False.

Para comparar el segundo bloque del primero.

```
1    miSet={1,2,3}.issubset({1,2})
2    print(miSet)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/sets.py
False
PS F:\CursoPython>
```

En cambio.

```
1    miSet={1,2}.issubset({1,2,3})
2    print(miSet)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/CursoPython/sets.py
True
PS F:\CursoPython>
```

issuperset → es superconjunto

issubset → es un subconjunto

56.- Módulo Datetime: Manejo de Fechas y Horas con Python

```
1    import datetime
2    fechaActual = datetime.datetime.now()
3    print(fechaActual)
```

Este será el resultado:

Muestra la fecha y hora actual.

Otra forma de importar la librería y el modo de uso.

```
1    from datetime import datetime
2    fechaActual = datetime.now()
3    print(fechaActual)
```

El resultado será el mismo.

```
1    import datetime
2    fecha=datetime.datetime(2020,11,5)
3    print(fecha)
```

Este será el resultado:

Me permite construir una feche a partir de una parámetros.

```
1    import datetime
2    fecha=datetime.datetime(2020,11,5,10,35,21)
3    print(fecha)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejoFechas.py
2020-11-05 10:35:21
PS F:\CursoPython>
```

Pasándole además los parámetros de horas, minutos y segundos.

```
1   import datetime
2   fechaActual=datetime.datetime.now()
3   fechaActual2=datetime.datetime.strftime(fechaActual, '%d/%m/%Y - %H:%M:%S')
4   print(fechaActual2)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejoFechas.py
08/02/2023 - 12:12:57
PS F:\CursoPython>
```

```
1   import datetime
2   fechaActual=datetime.datetime.now()
3   fechaActual2=datetime.datetime.strftime(fechaActual, '%b %d %Y %H:%M:%S')
4   print(fechaActual2)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejoFechas.py
Feb 08 2023 12:19:07
PS F:\CursoPython>
```

```
1   import datetime
2   fechaActual=datetime.datetime.now()
3   fechaActual2=datetime.datetime.strftime(fechaActual, '%B % %Y %H:%M:%S')
4   print(fechaActual2)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejoFechas.py
February 08 2023 12:24:42
PS F:\CursoPython>
```

```
1   import datetime
2   fechaTexto='Dec 06 2020 12:56:11'
3   fechaActual=datetime.datetime.strptime(fechaTexto,'%b %d %Y %H:%M:%S')
4   print(fechaActual)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejoFechas.py
2020-12-06 12:56:11
PS F:\CursoPython>
```

```
1   import datetime
2   fechaActual=datetime.datetime.now()
3   dia=datetime.datetime.strftime(fechaActual,'%d')
4   print(dia)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejoFechas.py
08
PS F:\CursoPython>
```

Lo vamos a convertir en entero.

```
1   import datetime
2   fechaActual=datetime.datetime.now()
3   dia=int(datetime.datetime.strftime(fechaActual,'%d'))
4   print(dia)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejoFechas.py
8
PS F:\CursoPython>
```

Formato entero.

```
1   import datetime
2   fechaActual=datetime.datetime.now()
3   horaActual=datetime.datetime.strftime(fechaActual,'%H:%M:%S')
4   print(horaActual)
```

Este será el resultado de la hora actual.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejoFechas.py
12:39:25
PS F:\CursoPython>
```

```
1   import datetime
2   fechaActual=datetime.datetime.now()
3   fechaPasada=datetime.datetime(2020, 10, 23)
4   diferencia = fechaActual - fechaPasada
5   print(diferencia)
6   print(diferencia.days)
7   print(diferencia.total_seconds())
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejoFechas.py
838 days, 12:45:41.916294
838
72449141.916294
PS F:\CursoPython>
```

```
1   import datetime
2   dia_delta = datetime.timedelta(days=5)
3   fechaInicial = datetime.date.today()
4   print(fechaInicial)
5   fechaFutura = fechaInicial + dia_delta
6   print(fechaFutura)
```

Este será el resultado al sumar 5 días a una fecha.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejoFechas.py
```

```
2023-02-08
2023-02-13
PS F:\CursoPython>
```

Si ponemos un valor negativo lo que hará será restar días.

```
1    import datetime
2    fecha = datetime.datetime.now().isoformat()
3    print(fecha)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejoFechas.py
2023-02-08T12:55:53.474698
PS F:\CursoPython>
```

Te muestra la fecha con este formato.

En el siguiente enlace encontrarás más información:

https://strftime.org/

57.- Módulo Math: Operaciones Matemáticas Avanzadas con Python

round()

```
1    import math
2
3    x = 1.553
4    print(round(x))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
2
PS F:\CursoPython>
```

```
4    print(round(x,1))
```

Redondeamos con un decimal.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
1.6
PS F:\CursoPython>
```

math.ceil(x) → redondea hacia arriba.

math.floor(x) → redondea hacia abajo.

```
1    import math
2
3    x = 1.553
4    print(math.ceil(x))
5    print(math.floor(x))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
2
1
PS F:\CursoPython>
```

```
1    import math
2
3    x = 1.553
4    print(math.trunc(x))
```

Este será el resultado:

Nos retorna la parte entera de un número.

```
1    import math
2    numeros = [1, 2, 3, 4, 5]
3    print(math.fsum(numeros))
```

Este será el resultado:

Suma los elementos de una lista, retorna un valor flotante pero utilizando un int() lo pasamos a número entero.

```
1    import math
2    print(math.fabs(-4))
```

Este será el resultado:

Retorna el valor absoluto de un número, si este es negativo lo retorna en positivo y si este es positivo se queda en positivo.

```
1    import math
2    print(math.fmod(17,6))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
5.0
PS F:\CursoPython>
```

Es el módulo de dividir 17 entre 6 tiene un residuo de 5.

```
1   import math
2   print(math.exp(2))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
7.38905609893065
PS F:\CursoPython>
```

El épsilon es uno de los números irracionales y los números transcendentes más importantes, aparece en diversas ramas de las matemáticas, su valor aproximado es de 2,71828.

```
1   import math
2   print(math.exp(2))
3   print(2.71828**2)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
7.38905609893065
7.3890461584
PS F:\CursoPython>
```

```
1   import math
2   print(math.pi)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
```

```
3.141592653589793
PS F:\CursoPython>
```

Nos retorna el valor de π.

```
1    import math
2    print(math.pow(5,6))
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
15625.0
PS F:\CursoPython>
```

Es lo mismo que

```
print(5**6)
```

```
1    import math
2    print(math.sqrt(25))
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
5.0
PS F:\CursoPython>
```

Calcula la raíz cuadrada de un número.

```
1    import math
2
3    h=math.hypot(1.5,1.5)
4    print(h)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
2.1213203435596424
PS F:\CursoPython>
```

Calcula la hipotenusa. $H^2 = C1^2 + C2^2$

Esto calcula la hipotenusa con una función.

```python
import math

def hipotenusa(c1, c2):
    sc = c1**2 + c2**2
    return math.sqrt(sc)

print(hipotenusa(1.5,1.5))
```

El resultado será el mismo.

```python
import math

r1 = math.radians(45)
r2 = 45/57.295
print(r1, r2)
```

Este será el resultado:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
0.7853981633974483 0.785408848939698
PS F:\CursoPython>
```

Convierte grados a radianes, sabiendo que un radian es igual a 57.295 grados.

```python
import math

print(math.sin(67))
```

Este será el resultado:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
-0.8555199789753223
PS F:\CursoPython>
```

```python
import math

print(math.sin(math.radians(30)))
```

Este será el resultado:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
```

```
0.49999999999999994
PS F:\CursoPython>
```

Hemos calculado 30 grados, los hemos pasado a radianes y sobre el valor obtenido hemos calculado el seno.

El último ejemplo.

```
1    import math
2
3    print(math.remainder(16,2))
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/matematicas.py
0.0
PS F:\CursoPython>
```

Nos dice 0.0 esto nos dice que estos dos números son divisibles porque 16 entre 2 da residuo 0.

```
print(math.remainder(16, 3))
```

Me retorna el valor 1 porque 16 entre 3 me devuelve 5 y con un residuo de 1.

58.- List Comprehension (Compresión de Listas) en Python

```
1   import math
2   # cuadrado de 1, 2, 3, 4 y 5
3   numeros = [1, 4, 9, 16, 25]
4   raices=[]
5   for n in numeros:
6       raices.append(int(math.sqrt(n)))
7
8   print(raices)
```

Este será el resultado:

Tenemos una lista de los cuadrados de 1, 2, 3, 4 y 5. A continuación creamos una segunda lista llamada raíces que estará vacía.

Con un ciclo for recorremos toda la lista números, de cada elemento le calculamos su raíz con valor entero que se lo agregamos a la lista raíces.

Terminadas las iteraciones del bucle mostramos el contenido de raíces.

```
1   import math
2   numeros = [1, 4, 9, 16, 25]
3   raices=[int(math.sqrt(x)) for x in numeros]
4
5   print(raices)
```

Otra forma de realizar el ejercicio anterior y obteniendo el mismo resultado.

La lista interna es asignada a la variable cuando todos los elementos han sido procesados.

Vamos a ver otro ejemplo:

```
2   numeros = [1, 4, 9, 16, 25]
3
4   v=[x if (x>10) else '*' for x in numeros]
5   print(v)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/compresionListas.py
['*', '*', '*', 16, 25]
PS F:\CursoPython>
```

En la línea 4 a la variable v de tipo lista le decimas que si x en mayor a 10 asigne a la lista el valor, de lo contrario que asigne un * en el recorrido de toda la lista mediante el for.

Vamos con otro ejemplo:

```
3    l=[c.upper() for c in 'UskoKruM2010']
4    print(l)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/compresionListas.py
['U', 'S', 'K', 'O', 'K', 'R', 'U', 'M', '2', '0', '1', '0']
PS F:\CursoPython>
```

En la línea 3 a la variable l de tipo lista añade cada carácter en mayúsculas como elemento de la lista.

Otro ejemplo:

```
1    a=[l if l in 'aeiou' else '*' for l in 'murcielago']
2    print(a)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/compresionListas.py
['*', 'u', '*', '*', 'i', 'e', '*', 'a', '*', 'o']
PS F:\CursoPython>
```

A al variable ‘a’ de tipo lista le vamos a asignar el carácter si es vocal de lo contrario un * durante el recorrido de la cadena ‘murcielago’ mediante un ciclo for.

59.- Enums en Python | Python Enumerations

```
1   from enum import Enum
2
3   class Color(Enum):
4       rojo = '#ff0000'
5       verde = '#008000'
6       azul = '#0000ff'
7
8   print(Color.rojo)
```

Este será el resultado:


```
1   from enum import Enum
2
3   class Color(Enum):
4       rojo = '#ff0000'
5       verde = '#008000'
6       azul = '#0000ff'
7
8   print(Color.rojo.value)
```

Este será el resultado:

Cambia a verde para comprobar el valor optenido.

```
8   print(Color.verde.value)

#008000
```

Agregamos la siguiente línea:

```
9    print(Color('#008000'))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/enums.py
#008000
Color.verde
PS F:\CursoPython>
```

Puedo obtener el nombre o llave.

Agregamos la siguiente línea.

```
10    print(Color['rojo'])
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/enums.py
#008000
Color.verde
Color.rojo
PS F:\CursoPython>
```

Modifica la línea 10.

```
10    print(Color['rojo'].value)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/enums.py
#008000
Color.verde
#ff0000
PS F:\CursoPython>
```

Vamos a ver como recuperar todos los colores:

```
1    from enum import Enum
2
3    class Color(Enum):
```

```
4       rojo = '#ff0000'
5       verde = '#008000'
6       azul = '#0000ff'
7
8   lista=[c for c in Color]
9   print(lista)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/enums.py
[<Color.rojo: '#ff0000'>, <Color.verde: '#008000'>, <Color.azul: '#0000ff'>]
PS F:\CursoPython>
```

Si lo queremos convertir en una lista:

```
8   lista=list([c for c in Color])
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/enums.py
[<Color.rojo: '#ff0000'>, <Color.verde: '#008000'>, <Color.azul: '#0000ff'>]
PS F:\CursoPython>
```

Agregamos la siguiente línea:

```
10   print(len(lista))
```

Este será el resultado:

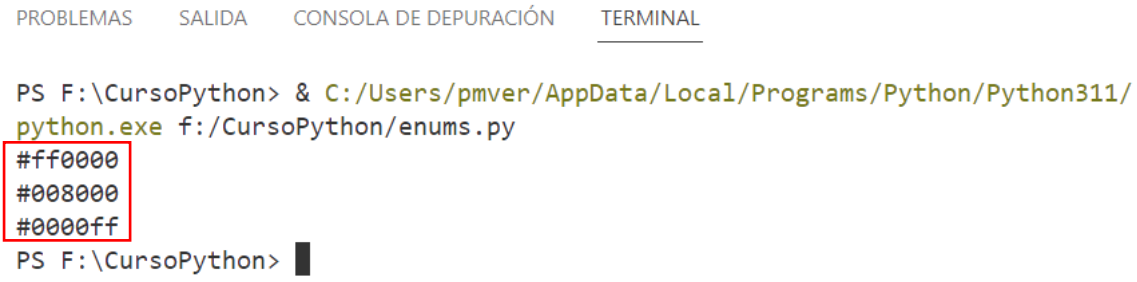
```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/enums.py
[<Color.rojo: '#ff0000'>, <Color.verde: '#008000'>, <Color.azul: '#0000ff'>]
3
PS F:\CursoPython>
```

Nos dice el número de elementos de la lista.

```
1   from enum import Enum
2
3   class Color(Enum):
```

```
    rojo = '#ff0000'
    verde = '#008000'
    azul = '#0000ff'

lista = list([c for c in Color])

for a in lista:
    print(a.value)
```

Para acceder a todos los valores de la lista, este será el resultado:



60.- Manejo de Archivos | Modos de apertura: Lectura, Escritura y Adjuntar

En este capítulo vamos a ver el manejo de archivos de texto para la lectura, escritura y adjuntar.

Vamos a crear una nueva carpeta llamada ManejoArchivos.

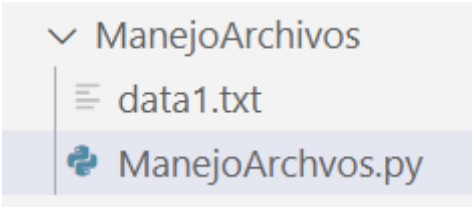
Creamos un archivo llamado data1.txt.

ManejoArchivos > data1.txt

```
1    UskoKruM2010
```

Ponemos el siguiente texto.

Vamos a crear otro archivo llamado ManejoArchivos.py.



Ahora en el ManejoArchivos vamos a escribir el siguiente código:

```
1    # Abrimos el archivo en modo lectura
2    file=open('ManejoArchivos\data1.txt', 'r')
3    print(file)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
<_io.TextIOWrapper name='ManejoArchivos\\data1.txt' mode='r' encoding='cp125
2'>
PS F:\CursoPython>
```

Con este mensaje nos dice que ya tenemos acceso al archivo.

```
1    # Abrimos el archivo en modo lectura
2    file=open('ManejoArchivos\data1.txt', 'r')
3    print(file)
4    file.close()  # Cerrar el archivo
```

Antes de finalizar el programa siempre en bueno cerrar el archivo.

Vamos a leer el contenido del archivo.

```
1    # Abrimos el archivo en modo lectura
2    file=open('ManejoArchivos\data1.txt', 'r')
3    print(file)
4    lineas=file.readlines()
5    print(lineas)
6    file.close()  # Cerrar el archivo
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
<_io.TextIOWrapper name='ManejoArchivos\\data1.txt' mode='r' encoding='cp125
2'>
['UskoKruM2010']
PS F:\CursoPython>
```

Vamos a agregar una línea nueva en el archivo de texto.

```
1    UskoKruM2010
2    Suscríbete
```

Guardamos los cambios del archivo data1.txt y desde el archivo ManejoArchivos.py lo ejecutamos de nuevo.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
<_io.TextIOWrapper name='ManejoArchivos\\data1.txt' mode='r' encoding='cp125
2'>
['UskoKruM2010\n', 'SuscrÃ\xadbete']
PS F:\CursoPython>
```

Nos muestra las dos líneas. El \n es un salto de línea.

Como hemos escrito la palabra Suscríbete con acento y no hemos especificado la codificación de caracteres para nosotros que es UTF-8.

Para solucionarlo vamos a modificar la línea 2.

```
2    file=open('ManejoArchivos\data1.txt', 'r', encoding="utf-8")
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
```

```
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
<_io.TextIOWrapper name='ManejoArchivos\\data1.txt' mode='r' encoding='utf-8
'>
['UskoKruM2010\n', 'Suscríbete']
PS F:\CursoPython>
```

La palabra ya aparece correctamente acentuada.

Para el segundo ejemplo vamos a crear otro archivo de texto llamado data2.txt con el siguiente texto.

```
1   Java
2   Python
3   C#
4   JavaScript
5   PHP
6   Kotlin
7   Ruby
```

Vamos al manejoArchivos.py

```
1   with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
```

Vamos a utilizar 'with" es para indicar que le archivo que hemos abierto en modo lectura vamos a utilizarlo como un alias.

```
1   with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
2       lineas = archivo.readlines()
3       print(lineas)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
['Java\n', 'Python\n', 'C#\n', 'JavaScript\n', 'PHP\n', 'Kotlin\n', 'Ruby']
PS F:\CursoPython>
```

Podemos ver los lenguajes de programación con sus respectivos saltos de línea \n.

```
1   with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
2       lineas = archivo.readlines()
3       print(lineas)
4
5   # Estamos fuera de with (Ya se ha cerrado el fichero data2.txt)
6   print(lineas)  # Pero los valores se mamntienen en la variable.
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL
```

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
['Java\n', 'Python\n', 'C#\n', 'JavaScript\n', 'PHP\n', 'Kotlin\n', 'Ruby']
['Java\n', 'Python\n', 'C#\n', 'JavaScript\n', 'PHP\n', 'Kotlin\n', 'Ruby']
PS F:\CursoPython>
```

Vamos a ver otra forma de recorrer la variable líneas.

Agregamos las siguiente líneas:

```
8    for l in lineas:
9        print(l)
```

Vamos a ver el resultado:

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
['Java\n', 'Python\n', 'C#\n', 'JavaScript\n', 'PHP\n', 'Kotlin\n', 'Ruby']
['Java\n', 'Python\n', 'C#\n', 'JavaScript\n', 'PHP\n', 'Kotlin\n', 'Ruby']
Java

Python

C#

JavaScript

PHP

Kotlin

Ruby
PS F:\CursoPython>
```

¿Porque los vemos tan separados de un curso a otro? Es debido al \n que provoca un salto de línea.

Para esto vamos a modificar la línea 9.

```
9        print(l.replace('\n','' ))
```

Este será el resultado:

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
['Java\n', 'Python\n', 'C#\n', 'JavaScript\n', 'PHP\n', 'Kotlin\n', 'Ruby']
['Java\n', 'Python\n', 'C#\n', 'JavaScript\n', 'PHP\n', 'Kotlin\n', 'Ruby']
```

```
Java
Python
C#
JavaScript
PHP
Kotlin
Ruby
PS F:\CursoPython>
```

Vamos a ver otro ejemplo:

```
1  with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
2      contenido=archivo.read()
3      lineas=contenido.split('\n') # Quitar saltos de línea
4      print(lineas)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
['Java', 'Python', 'C#', 'JavaScript', 'PHP', 'Kotlin', 'Ruby']
PS F:\CursoPython>
```

Ya lo tenemos en modo de lista para poder trabajar con ellos.

Vamos con otro ejemplo:

```
1  with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
2      pos=archivo.tell()
3      print(pos)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
0
PS F:\CursoPython>
```

Al principio cuando abrimos el archivo en modo de lectura nos encontramos en la posición 0.

Vamos a realizar algunas modificaciones:

```
1  with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
2      contenido = archivo.read()
3      lineas = contenido.split('\n')
4      pos=archivo.tell()
5      print(pos)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
47
PS F:\CursoPython>
```

En el momento de hacer la lectura el cursor de ubicó al final del archivo.

```
1  with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
2      contenido = archivo.read()
3      lineas = contenido.split('\n')
4      pos=archivo.tell()
5      print("El archivo tiene {0} caracteres de longitud".format(pos))
```

Si modificamos la línea 5 este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
El archivo tiene 47 caracteres de longitud
PS F:\CursoPython>
```

Vamos con otro ejemplo:

```
with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
    archivo.seek(7)
    pos=archivo.tell()
    print(pos)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
7
PS F:\CursoPython>
```

Nos indica que estamos en la posición 7.

```
1  with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
2      archivo.seek(7)
3      pos=archivo.tell()
4      print(pos)
5      contenido=archivo.read()
6      lineas=contenido.split('\n')
7      print(lineas)
```

Que pasa si leemos desde esta posición.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
7
['ython', 'C#', 'JavaScript', 'PHP', 'Kotlin', 'Ruby']
PS F:\CursoPython>
```

Estamos iniciando la lectura a partir de la posición 7.

Otro ejemplo:

```
1  with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
2      siguientes4=archivo.read(4)
3      print(siguientes4)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
Java
PS F:\CursoPython>
```

Muestra los 4 primeros caracteres.

```
1  with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
2      siguientes7=archivo.read(7)
3      print(siguientes7)
```

Hemos cambiado el 4 por un 7.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
Java
Py
PS F:\CursoPython>
```

Podemos pedir cuantos caracteres queremos leer.

Vamos con otro ejemplo:

```
1  with open('ManejoArchivos\data2.txt', 'r', encoding="utf-8") as archivo:
2      print(type(archivo.read()))
```

Si abrimos en modo ‘r’, este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
```

```
<class 'str'>
PS F:\CursoPython>
```

Vemos una clase String.

```
1  with open('ManejoArchivos\data2.txt', 'rb') as archivo:
2      print(type(archivo.read()))
```

Si lo abrimos en modo 'rb', este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
<class 'bytes'>
PS F:\CursoPython>
```

Vemos una clase que la lee en bytes.

Estos son los modos a los que tenemos acceso:

Caracter	Significado
'r'	abierto para lectura (predeterminado)
'w'	abierto para escritura, truncando el archivo primero
'x'	abierto para creación exclusiva, fallando si el archivo ya existe
'a'	abierto para escritura, agregando al final del archivo si existe
'b'	modo binario
't'	modo de texto (predeterminado)
'+'	abierto para actualizar (lectura y escritura)

Vamos a ver un ejemplo de escritura:

```
1  with open('ManejoArchivos\data3.txt', 'w', encoding="utf-8") as archivo:
2      archivo.write('Oscar\nAlejandro\nFlavio')
```

Vamos a ejecutar, este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/ManejoArchivos/ManejoArchivos.py
PS F:\CursoPython>
```

Por consola no ha dicho nada, pero nos ha creado un archivo llamado data3.txt con el siguiente contenido.

ManejoArchivos > data3.txt

```
1   Oscar
2   Alejandro
3   Flavio
```

Si volvemos a ejecutar el programa anterior modificando la línea 2 con otro contenido este se va a sobre escribir.

```
1   with open('ManejoArchivos\data3.txt', 'w', encoding="utf-8") as archivo:
2       archivo.write('Pere Manel\nVerdugo\nZamora')
```

Tu lo puedes modificar con tu nombre y apellidos.

Este será el contenido del archivo data3.txt.

ManejoArchivos > data3.txt

```
1   Pere Manel
2   Verdugo
3   Zamora
```

Para poder añadir al final del archivo:

```
1   with open('ManejoArchivos\data3.txt', 'a', encoding="utf-8") as archivo:
2       archivo.write('\nCarlos\nMartínez\nRuiz')
```

Vamos a ejecutar y este será el resultado:

ManejoArchivos > data3.txt

```
1   Pere Manel
2   Verdugo
3   Zamora
4   Carlos
5   Martínez
6   Ruiz
```

Hemos agregado contenido al final de este.

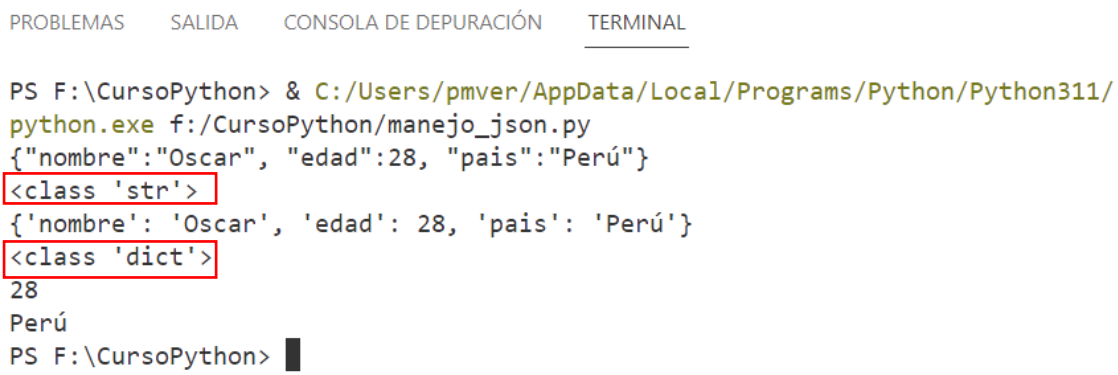
61.- JSON: Escritura y Lectura de Datos en JSON(JavaScript Object Notation)

Definición de JSON → JavaScript Object Notation lo vamos a utilizar tanto para la lectura como la escritura de datos.

Vamos a crear un nuevo archivo ce Python.

```
1   import json
2   json_str='{"nombre":"Oscar", "edad":28, "pais":"Perú"}'
3   print(json_str)
4   print(type(json_str))
5
6   python_dict=json.loads(json_str)
7   print(python_dict)
8   print(type(python_dict))
9
10  print(python_dict['edad'])
11  print(python_dict['pais'])
```

Este será el resultado:



Línea 1 importamos la clase json

Línea 2 Asignamos a la variable json_str el valor de un string que va entre comillas simples.

Línea 3 Mostramos por consola en contenido de la variable json_str.

Línea 4 Mostramos por consola que tipo de objeto es json_str, nos dice <class 'str'>.

Línea 6 con la sentencia json.loads(de la variable json_str) a diccionario que se lo asignamos a la variable python_dict.

Línea 7 Mostramos el contenido de la variable python_dict por consola, y observamos que ahora tiene un aspecto de diccionario.

Línea 8 nos dice de que tipo de objeto es Python_dict, nos dice <class 'dict'>.

Ahora como la variable Python_dict es un diccionario podemos consultar por su edad y país pasando como parámetro la clave para que nos retorne el valor.

Ahora vamos a realizar el proceso inverso con respecto al ejemplo anterior.

```
1   import json
2   data={
3       "youtuber":"UskoKruM2010",
4       "nombre":"Oscar",
5       "edad":28
6   }
7   json_data=json.dumps(data)
8   print(type(json_data))
9   print(json_data)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejo_json.py
<class 'str'>
{"youtuber": "UskoKruM2010", "nombre": "Oscar", "edad": 28}
PS F:\CursoPython>
```

En la línea 2 creamos un diccionario que se lo asignamos a la variable data.

En la línea 7 convertimos el diccionario a un String (objeto json) asignándoselo a la variable json_data.

En la línea 8 imprimimos el tipo de objeto que es json_data que es <class ‘str’>. Objeto json

En la línea 9 imprimimos el contenido de la variable json_data.

```
1   import json
2   data={
3       "youtuber":"UskoKruM2010",
4       "nombre":"Oscar",
5       "edad":28,
6       "cursos":["PHP", "Python", "JavaScript", "C#", "Node.js"]
7   }
8   json_data=json.dumps(data)
9   print(type(json_data))
10  print(json_data)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejo_json.py
```

```
<class 'str'>
{"youtuber": "UskoKruM2010", "nombre": "Oscar", "edad": 28, "cursos": ["PHP"
, "Python", "JavaScript", "C#", "Node.js"]}
PS F:\CursoPython>
```

La variable json_data se ha convertido en objeto JSON, tal como se muestra en la consola.

Tipos de datos en Python y sus equivalentes en JSON.

Python	JSON
dict	Object
list	Array
tuple	Array
str	String
int	Number
float	Number
True	true
False	false
None	null

Vamos a ver otro ejemplo:

```
1   import json
2   data={
3       "youtuber":"UskoKruM2010",
4       "nombre":"Oscar",
5       "edad":28,
6       "cursos":["PHP", "Python", "JavaScript", "C#", "Node.js"]
7   }
8   json_data=json.dumps(data,indent=4, separators=(", ", " : "))
9   print(json_data)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejo_json.py
{
    "youtuber" : "UskoKruM2010",
    "nombre" : "Oscar",
    "edad" : 28,
    "cursos" : [
        "PHP",
        "Python",
        "JavaScript",
        "C#",
        "Node.js"
    ]
}
PS F:\CursoPython>
```

Pero además si lo queremos ordenado vamos a modificar la línea 8.

```
8    json_data=json.dumps(data,indent=4, separators=(", ", " : "), sort_keys=True)
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejo_json.py
{
    "cursos" : [
        "PHP",
        "Python",
        "JavaScript",
        "C#",
        "Node.js"
    ],
    "edad" : 28,
    "nombre" : "Oscar",
    "youtuber" : "UskoKruM2010"
}
PS F:\CursoPython>
```

Otro ejemplo:

```
1    import json
2    json_data=json.JSONEncoder().encode({"lenguajes":["Python","JavaScript"]})
3    print(json_data)
4    print(type(json_data))
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejo_json.py
{"lenguajes": ["Python", "JavaScript"]}
<class 'str'>
PS F:\CursoPython>
```

La variable json_data es de tipo objeto JSON.

Vamos a agregar las siguientes líneas al ejemplo anterior:

```
6    python_dict=json.JSONDecoder().decode(json_data)
7    print(python_dict)
8    print(type(python_dict))
```

Este será el resultado:

```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejo_json.py
```

```
{"lenguajes": ["Python", "JavaScript"]}
<class 'str'>
{'lenguajes': ['Python', 'JavaScript']}
<class 'dict'>
PS F:\CursoPython>
```

Un objeto de tipo diccionario lo hemos convertido JSON y después de JSON lo hemos convertido a tipo diccionario.

También podemos procesar un archivo txt y procesarlo de Python.

Vamos a trabajar con JSON desde una clase personalizada.

```
1   import json
2
3   class Curso():
4       def __init__(self,codigo, nombre, creditos ):
5           self.codigo = codigo
6           self.nombre = nombre
7           self.creditos = creditos
8
9   curso_1=Curso("9841", "Lenguaje", 4)
10  print(curso_1)
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS F:\CursoPython> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/
python.exe f:/CursoPython/manejo_json.py
<__main__.Curso object at 0x0000016F1C9C7210>
PS F:\CursoPython>
```

Podemos ver la referencia pero no la información.

```
1   import json
2
3   class Curso():
4       def __init__(self,codigo, nombre, creditos ):
5           self.codigo = codigo
6           self.nombre = nombre
7           self.creditos = creditos
8
```

```
 9    curso_1=Curso("9841", "Lenguaje", 4)
10    print(curso_1)
11    json_object_data=json.dumps(curso_1.__dict__)
12    print(json_object_data)
```

Este será el resultado:

En la línea 11 .__dict__ es para convertirlo en diccionario.

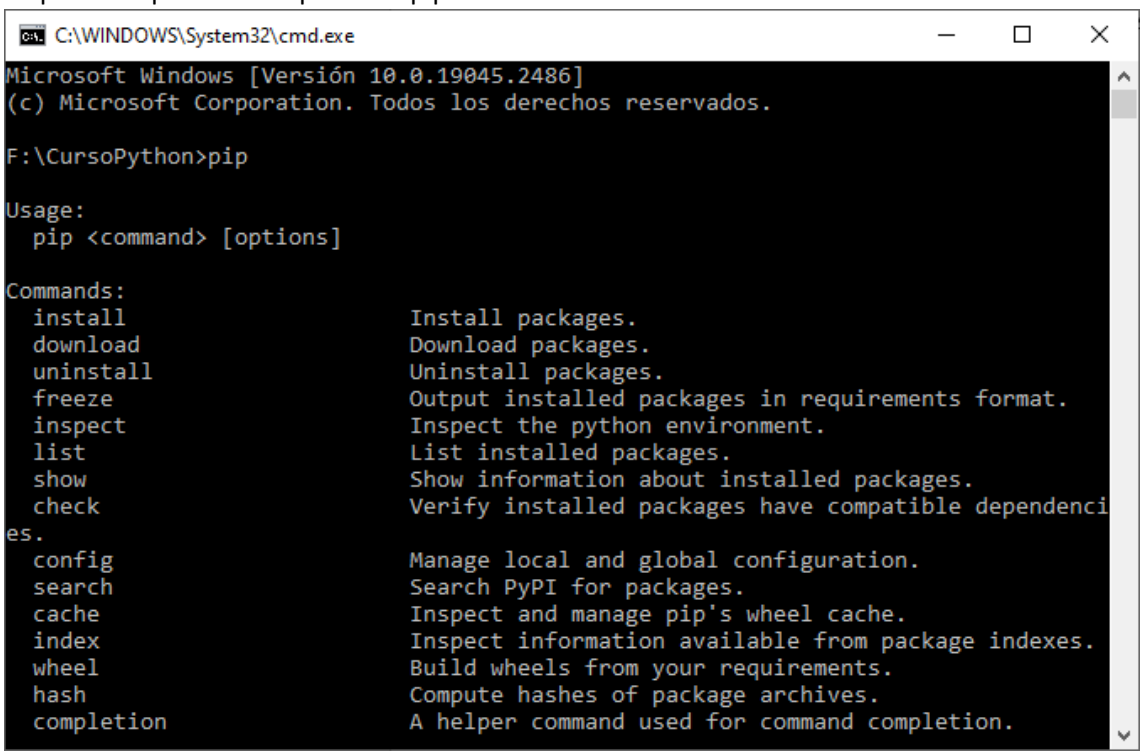
Vamos a agregar el siguiente código:

```
14    python_dict = json.loads(json_object_data)
15    print(python_dict)
16    print(type(python_dict))
```

Este será el resultado:

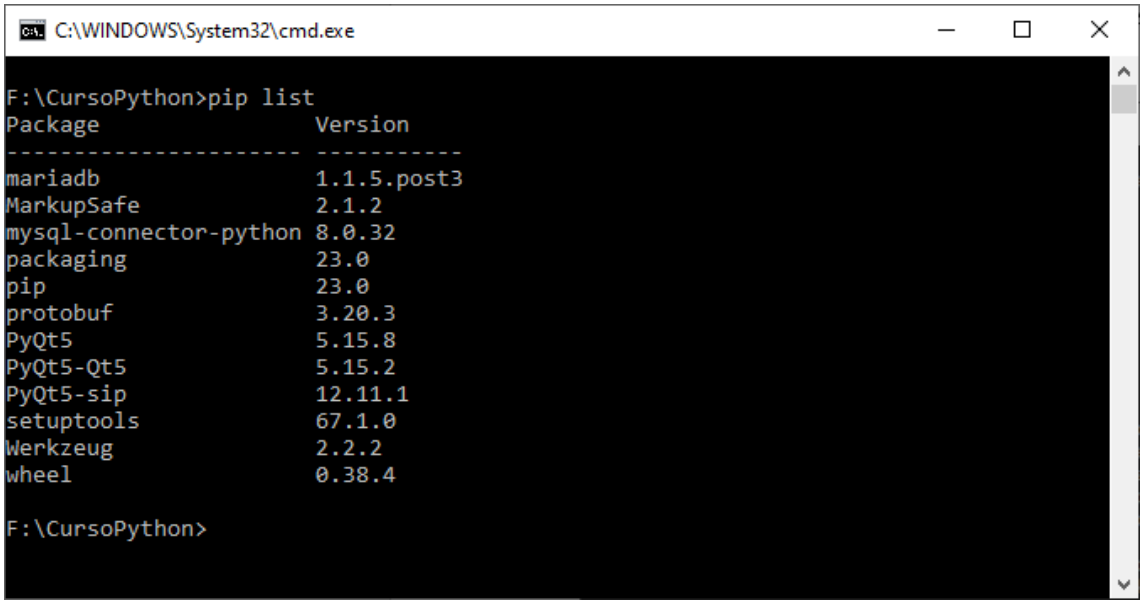
62.- PIP (Package Installer for Python): Administrador de Paquetes de Python

PIP: Instalador de paquetes para Python.
La forma de usarlas es a través de la línea de comando de la consola o también utilizando el Cmd símbolo de sistema de Windows.
Para los ejemplos vamos a utilizar la ventana Cmd del sistema.
Lo primero que vamos a poner es pip.



Se listan una serie de comandos.

Ahora vamos a aprobar pip list

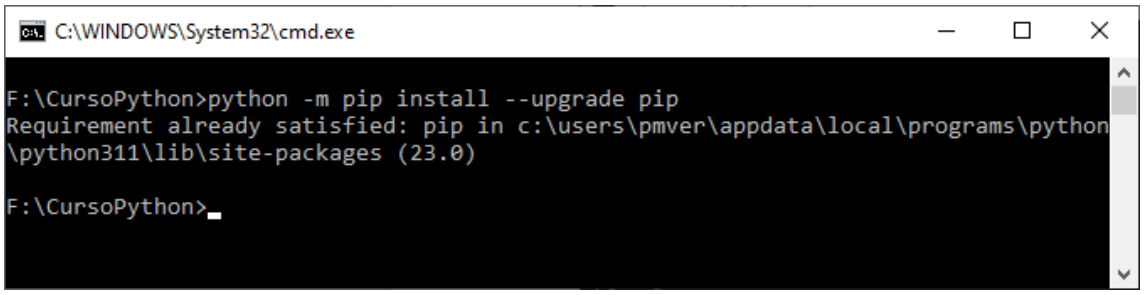


Podemos ver todos los paquetes que tenemos instalados y disponibles de manera global dentro del entorno que tengo para el trabajo con Python.

Para realizar un actualización de la herramienta pip vamos a escribir los siguiente:

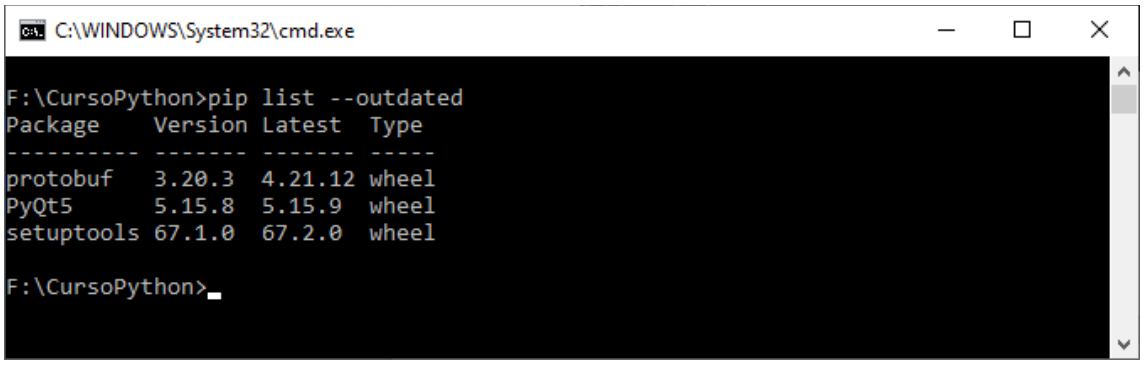
Python -m pip install - - upgrade pip.

Hay que esperar un momento a que se desinstale e instale la versión actual.



Podemos ver aquellos paquetes que están desactualizados.

pip list –outdated.



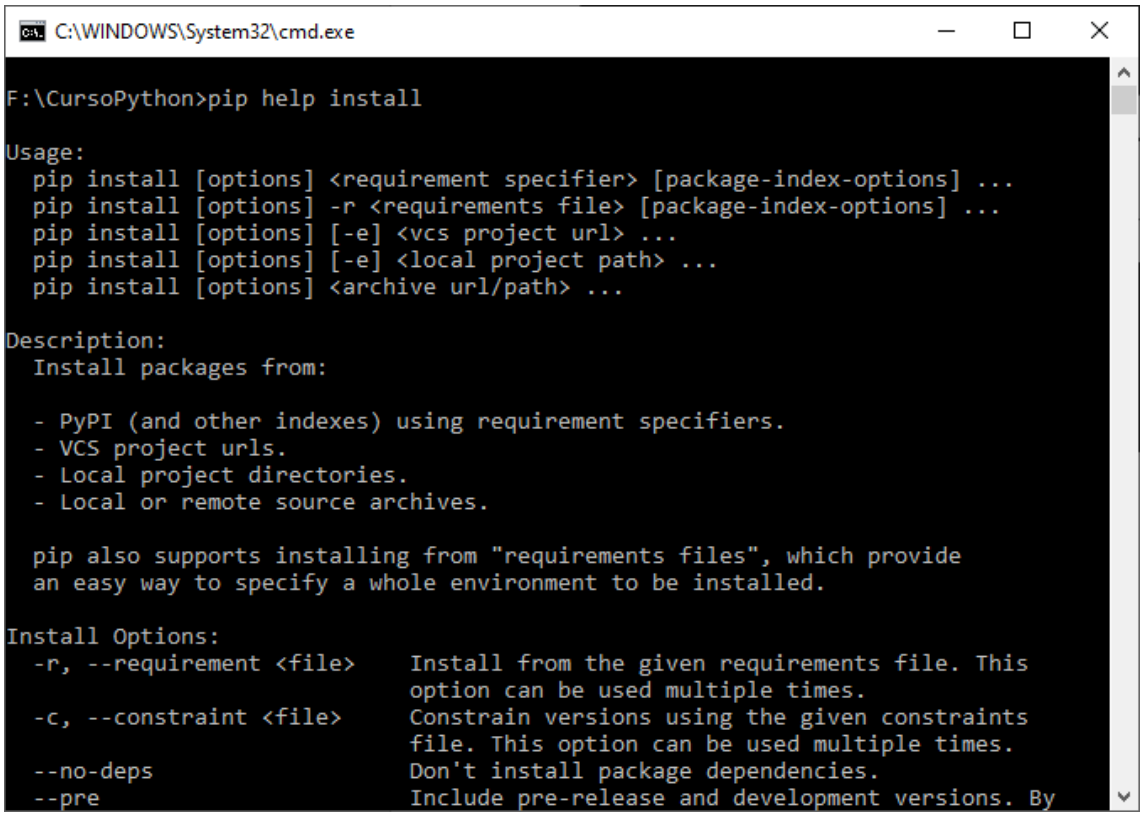
Ahora queremos ver más información acerca de Django.

pip show Django

Yo no lo tengo para instalarlo:

Para obtener inforación de todo los comandos:

pip help install.


C:\WINDOWS\System32\cmd.exe
F:\CursoPython>pip help install
Usage:
pip install [options] <requirement specifier> [package-index-options] ...
pip install [options] -r <requirements file> [package-index-options] ...
pip install [options] [-e] <vcs project url> ...
pip install [options] [-e] <local project path> ...
pip install [options] <archive url/path> ...
Description:
Install packages from:
- PyPI (and other indexes) using requirement specifiers.
- VCS project urls.
- Local project directories.
- Local or remote source archives.
pip also supports installing from "requirements files", which provide
an easy way to specify a whole environment to be installed.
Install Options:
-r, --requirement <file> Install from the given requirements file. This
option can be used multiple times.
-c, --constraint <file> Constrain versions using the given constraints
file. This option can be used multiple times.
--no-deps Don't install package dependencies.
--pre Include pre-release and development versions. By

Práctica: Módulos y Paquetes en Python | Importación de módulos, paquetes y subpaquetes.

Vamos a crear una carpeta llamada modulos_python

Un módulo es un archivo de Python cuyos objetos, funciones, clases, excepciones, etc. pueden ser accedidos desde otro archivo de Python.

Se trata simplemente de una forma de organizar grandes códigos dentro de un proyecto.

Una vez ejecutado visual Code Studio, abrimos la carpeta modulos_python y a continuación vamos a crear un archivo llamado módulos.py.

Creamos otro archivo llamado modulos1.py.

En modulos1.py vamos a escribir el siguiente código:

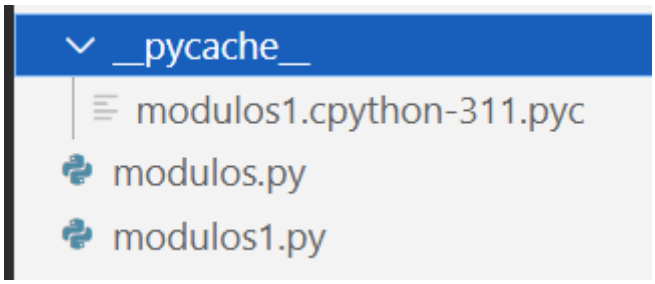
```
1   def sumar(a,b):
2       return a+b
3
4   def resta(a,b):
5       return a-b
6
7   def multiplicar(a,b):
8       return a*b
9
10  def dividir(a,b):
11      return a/b
```

Ahora queremos utilizarlos desde el archivo modulos.py

```
1   import modulos1
2
3   print(modulos1.multiplicar(7,8))
```

Si ejecutamos este será el resultado:

El resultado será 56.



Podréis ver que se ha creado la carpeta __pycache__ con un archivo compilado de Python llamado modulos1.pcython-311.pyc.

En este ejemplo hemos hecho la importación de un módulo que se encuentra en la misma ruta.

```
1    import modulos1
2
3    print(modulos1.sumar(7,8))
4    print(modulos1.restar(7,8))
5    print(modulos1.multiplicar(7,8))
6    print(modulos1.dividir(7,8))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   TERMINAL   ···                Python + ···

PS F:\modulos_python> & C:/Users/pmver/AppData/Local/Programs/Python/Pyth
on311/python.exe f:/modulos_python/modulos.py
15
-1
56
0.875
PS F:\modulos_python>
```

¿Qué tenemos que hacer para evitar tener que escribir cada vez modulos1.

```
1    from modulos1 import suma
```

Podemos importar una función.

```
1    from modulos1 import suma, resta
```

Importar varias funciones

```
1    from modulos1 import *
```

Podemos importar todas las funciones del archivo modulos1.py

```
1   from modulos1 import *
2
3   print(sumar(7,8))
4   print(resta(7,8))
5   print(multiplicar(7,8))
6   print(dividir(7,8))
```

El resultado será el mismo que en el ejercicio anterior.

Ahora volvemos al archivo molulos1.py para definir una variable.

```
13    canal="USkoKruM2010"
```

Agrega esta línea y nos vamos al archivo modulos.py

```
1   from modulos1 import *
2
3   print(sumar(7,8))
4   print(resta(7,8))
5   print(multiplicar(7,8))
6   print(dividir(7,8))
7
8   print(canal)
```

Vamos a ejecutar:

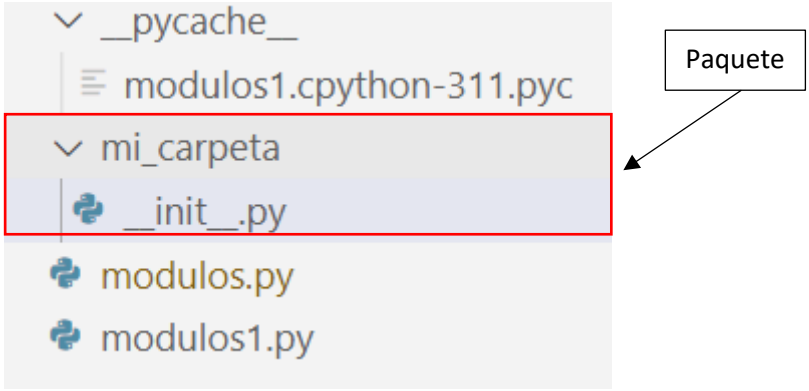
Si solo queremos importar sumar, resta y canal

```
1   from modulos1 import sumar, resta, canal
```

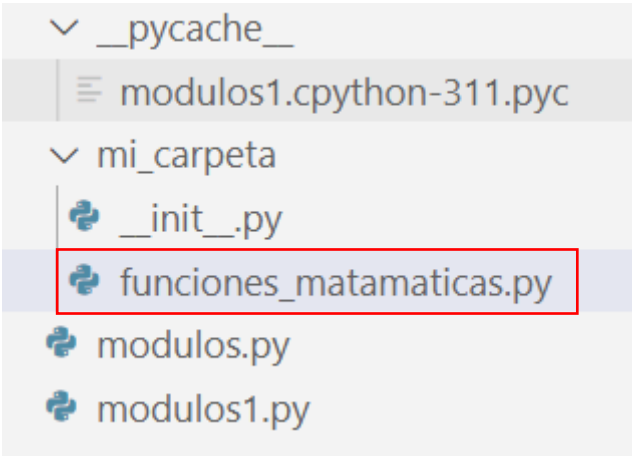
Ahora vamos a crear un paquete de Python en la cual vamos a tener funcionalidades englobadas y vamos a poder llamar a estas funcionalidades a través de su correcta importación.

¿Qué es un paquete? No es más que una carpeta que contiene varios módulos, pero tiene una particularidad, debe contener un archivo llamado __init__.py, el cual puede estar vacío para que el interprete de Python entienda que se trata de un paquete y no de una simple carpeta.

Vamos a crear una carpeta llamada mi_paquete y dentro el archivo __init__.py.



Dentro de la carpeta mi_carpeta vamos a crear un archivo llamado funciones_matematicas.py.



En funciones_matematicas.py vamos a crear la correspondiente función.

mi_carpeta > funciones_matamaticas.py > ...

```python
frase = 'Suscríbete'

def calcular_factorial(numero):
    factorial = 1
    for n in range(1, (numero+1)):
        factorial *= n
    return factorial
```

Ahora vamos a modulos.py

```python
from modulos1 import *
from mi_carpeta.funciones_matamaticas import calcular_factorial, frase

```

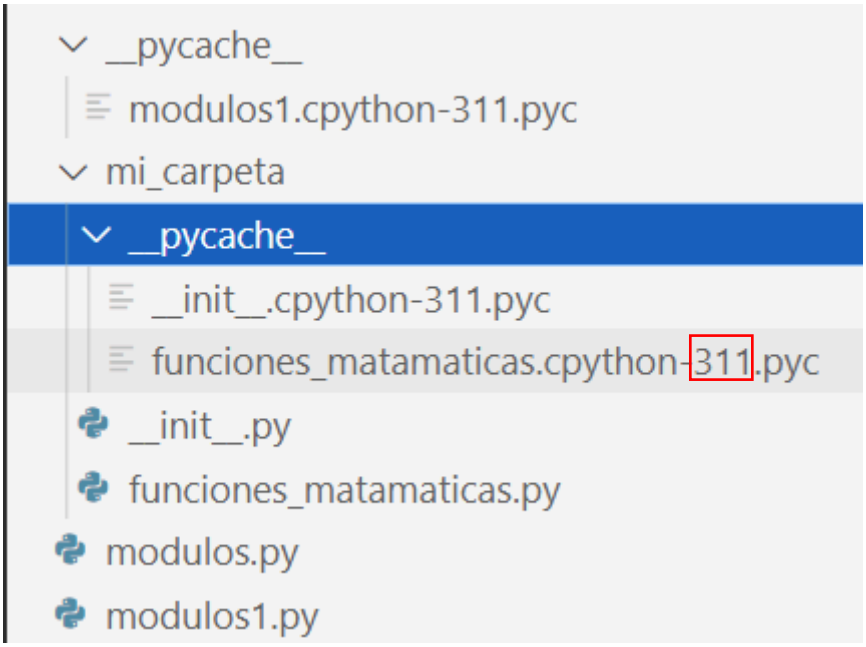
```
4   print(sumar(7,8))
5   print(resta(7,8))
6   print(multiplicar(7,8))
7   print(dividir(7,8))
8
9   print(canal)
```

Agregamos las siguientes líneas de código:

```
11   print(calcular_factorial(5))
12   print(frase)
```

Vamos a ejecutar:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL                Python

PS F:\modulos_python> & C:/Users/pmver/AppData/Local/Programs/Python/Python311/python.
exe f:/modulos_python/modulos.py
15
-1
56
0.875
USkoKruM2010
120
Suscríbete
PS F:\modulos_python>
```



Observamos que se ha creado una carpeta llamada __pycache__, con los siguientes archivos __init__.py y funciones_matematicas.cpython-311.pyc, este último en la compilación de la función funciones_matematicas del archivo funciones_matematicas.py y está compilado con la versión 3.11 de Python.

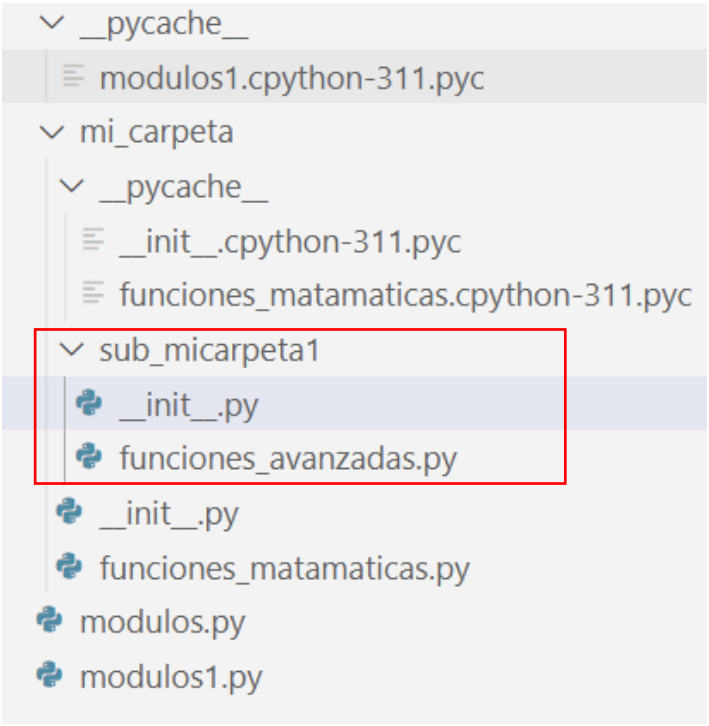
A la hora de importar podemos ponerle un alias.

```python
from modulos1 import *
import mi_carpeta.funciones_matamaticas as fun_mat

print(sumar(7,8))
print(resta(7,8))
print(multiplicar(7,8))
print(dividir(7,8))

print(canal)

print(fun_mat.calcular_factorial(5))
print(fun_mat.frase)
```

Se denomina alias

Ahora veremos como podemos crear un subpaquete, es decir un paquete dentro de otro paquete.

Vamos a crear una carpeta llamada sub_carpeta2 y un archivo dentro llamado funciones_avanzadas.py, más el archivo __init__.py.



Vamos a funciones_avanzadas.py.

```python
def contar_letras(frase):
    numero_letras = 0
    for l in frase:
        if l != ' ':
```

```
5                numero_letras += 1
6        return numero_letras
```

Nos vamos a modulos.py

```
1   from modulos1 import *
2   import mi_carpeta.funciones_matamaticas as fun_mat
3   #importación de un subpaquete
4   from mi_carpeta.sub_micarpeta1.funciones_avanzadas import contar_letras
5
6   print(sumar(7,8))
7   print(resta(7,8))
8   print(multiplicar(7,8))
9   print(dividir(7,8))
10
11  print(canal)
12
13  print(fun_mat.calcular_factorial(5))
14  print(fun_mat.frase)
15
16  texto = "Grácias por apoyar mi canal"
17  print(contar_letras(texto))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL   Python

PS F:\modulos_python> & C:/Users/pmver/AppData/Local/Programs/Python/Python311
/python.exe f:/modulos_python/modulos.py
15
-1
56
0.875
USkoKruM2010
120
Suscríbete
23
PS F:\modulos_python>
```

Un módulo es un archivo de Python cuyos objetos (funciones, clases, excepciones, etc.) pueden ser accedidos desde otro archivo. Se trata simplemente de una forma de organizar grandes códigos.

Un paquete es una carpeta que contiene varios módulos. Siguiendo el ejemplo anterior, podemos diseñar un paquete matematica creando una carpeta con la siguiente estructura.

Debe contener siempre un archivo __init__.py (por el momento vacío) para que Python entienda que se trata de un paquete y no de una simple carpeta.

Python tiene sus propios módulos, los cuales forman parte de su librería de módulos estándar, que también pueden ser importados.

PEP 8: importación
La importación de módulos debe realizarse al comienzo del documento, en orden alfabético de paquetes y módulos.

Primero deben importarse los módulos propios de Python. Luego, los módulos de terceros y finalmente, los módulos propios de la aplicación.

Entre cada bloque de imports, debe dejarse una línea en blanco.

Práctica: Generador de Contraseñas Aleatorias en Python

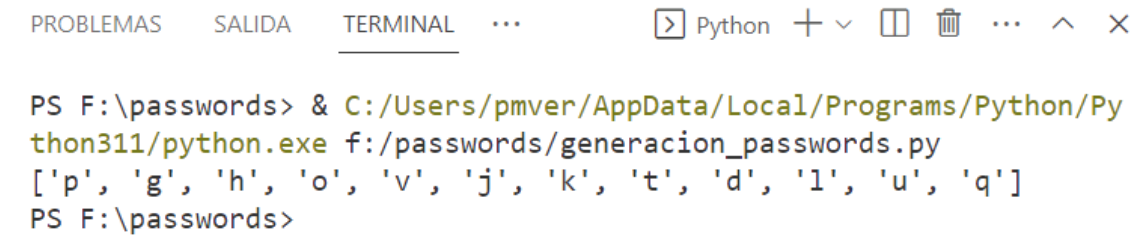
Para este proyecto vamos a crear una carpeta llamada passwords.

En ella vamos a crear un archivo llamado generación_passwords.py.

Vamos a escribir el siguiente código:

```
1    import random
2
3    minus="abcdefghijklmnopqrstuvwxyz"
4    longitud=12
5    muestra=random.sample(minus, longitud)
6    print(muestra)
```

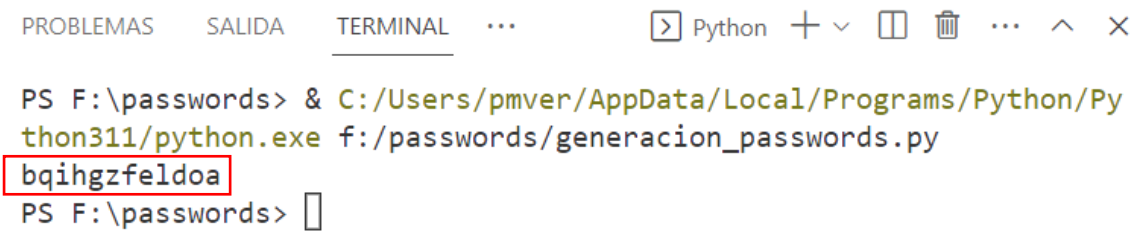
Este será el resultado:



Queremos que todos los elementos se junten.

```
1    import random
2
3    minus="abcdefghijklmnopqrstuvwxyz"
4    longitud=12
5    muestra=random.sample(minus, longitud)
6    password="".join(muestra)
7    print(password)
```

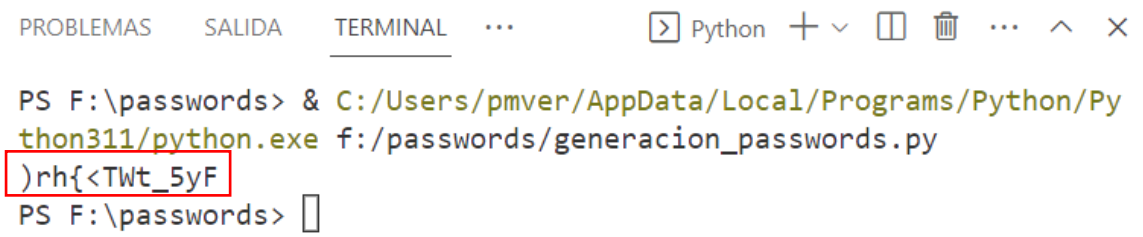
Este será el resultado:




```
1    import random
2
```

```
3   minus="abcdefghijklmnopqrstuvwxyz"
4   mayus=minus.upper()
5   numero="0123456789"
6   simbolos="@()[]{}*,;/-_¿?.¡$<#>&+%="
7   base=minus+mayus+numero+simbolos
8   longitud=12
9   muestra=random.sample(base, longitud)
10  password="".join(muestra)
11  print(password)
```

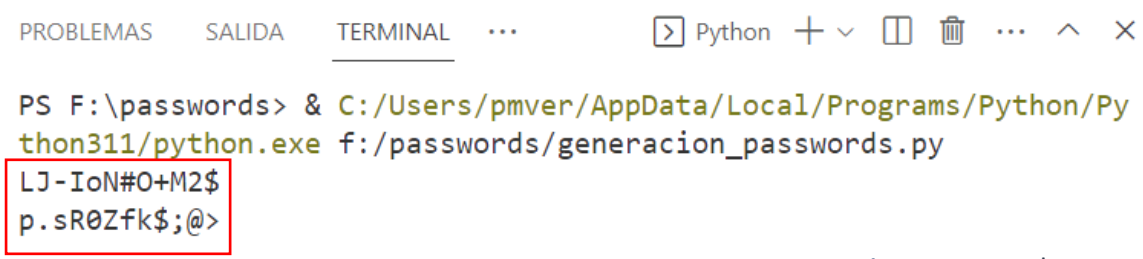
Este será el resultado:



Queremos que genere 10 passwords a la vez.

```
1   import random
2
3   minus="abcdefghijklmnopqrstuvwxyz"
4   mayus=minus.upper()
5   numero="0123456789"
6   simbolos="@()[]{}*,;/-_¿?.¡$<#>&+%="
7   base=minus+mayus+numero+simbolos
8   longitud=12
9   for _ in range(10):
10      muestra=random.sample(base, longitud)
11      password="".join(muestra)
12      print(password)
```

Este será el resultado:



```
¡tk<{LbWIo*1
DrCx=96FIS/U
p¡1O¿HEvhc$C
y]ZVM=}As,5t
yhVd/v8k[UB1
x7{blMoUNL$a
Jo+F#i2BOgZb
tsxr0;ipE7z-
PS F:\passwords>
```

Para la siguiente parte del tutorial necesitamos una librería llamada werkzeug , para saber si lo tenemos instalado haremos: pip list

```
PROBLEMAS  SALIDA  TERMINAL  ···                Python + ⌄  ···  ^

PS F:\passwords> pip list
Package                 Version
----------------------- -----------
mariadb                 1.1.5.post3
mysql-connector-python 8.0.32
packaging               23.0
pip                     23.0
protobuf                3.20.3
PyQt5                   5.15.8
PyQt5-Qt5               5.15.2
PyQt5-sip               12.11.1
setuptools              67.1.0
wheel                   0.38.4
PS F:\passwords>
```

En nuestro caso no lo tenemos instalado, lo vamos a instalar.

pip install werkzeug

```
PROBLEMAS  SALIDA  TERMINAL  ···                Python + ⌄  ···  ^  ×

PS F:\passwords> pip install werkzeug
Collecting werkzeug
  Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.7/232.7 kB 4.7 MB/s eta 0:00:00
Collecting MarkupSafe>=2.1.1
  Downloading MarkupSafe-2.1.2-cp311-cp311-win_amd64.whl (16 kB)
Installing collected packages: MarkupSafe, werkzeug
Successfully installed MarkupSafe-2.1.2 werkzeug-2.2.2
PS F:\passwords>
```

Ahora volveremos a poner pip list para ver si ya la tenemos instalada.

```
PROBLEMAS   SALIDA   TERMINAL   ···                    Python + ⌄   ···   ^

PS F:\passwords> pip list
Package                 Version
----------------------- -----------
mariadb                 1.1.5.post3
MarkupSafe              2.1.2
mysql-connector-python  8.0.32
packaging               23.0
pip                     23.0
protobuf                3.20.3
PyQt5                   5.15.8
PyQt5-Qt5               5.15.2
PyQt5-sip               12.11.1
setuptools              67.1.0
Werkzeug                2.2.2
wheel                   0.38.4
PS F:\passwords>
```

Ya lo tenemos instalado.

Vamos a seguir con el código:

```python
import random
from werkzeug.security import generate_password_hash

minus="abcdefghijklmnopqrstuvwxyz"
mayus=minus.upper()
numero="0123456789"
simbolos="@()[]{}*,;/-_¿?.¡$<#>&+%="
base=minus+mayus+numero+simbolos
longitud=12
for _ in range(10):
    muestra=random.sample(base, longitud)
    password="".join(muestra)
    password_encriptado=generate_password_hash(password)
    print("{} => {}".format(password, password_encriptado))
```

Este será el resultado:

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL                Python + ⌄   ···   ^   ×

t<%.mX-/UfrK => pbkdf2:sha256:260000$NoqsqJJRVYG0LLMd$30a6a73a77089e4a2e8317fa1364785a8e6b4014322888446db2ebd06142550f
J*8;H/y@R]%X => pbkdf2:sha256:260000$1SPNaneJ4ri0ABr4$01e9488117f92525b5a03fc3934a459cfe6a63b4b564dd7daf3df3da14647c15
vj@Q+Z#$kFf) => pbkdf2:sha256:260000$S559Hh0HhmPFDz9x$e5670bb1d00ff41dbde5fad49aab52d773b160d55547bd0b6bd0cead34a0ba95
,MHYti=7J#]j => pbkdf2:sha256:260000$umjJn64Cf8lybM3N$93fa0ac61aece7e1f6632687f1807b3368412ba265aa739f25f64124b9c55a4a
52F1KX@&z#w¿ => pbkdf2:sha256:260000$ltWx5vxROvTP2QWT$06e3e7f7ea6d473bf94f171a84b7e0698f36d3fccf60c4b5420cb6a500ed69d8
6O(5nAq@4t*m => pbkdf2:sha256:260000$Tfbe2EOFhAsahyln$604bcea085b42052ca65e88d84e854387e66d4094fb356cd07c6ac1579b39bc6
PS F:\passwords>
```

Normal Encriptado