6 PROYECTOS BÁSICOS DE PYTHON

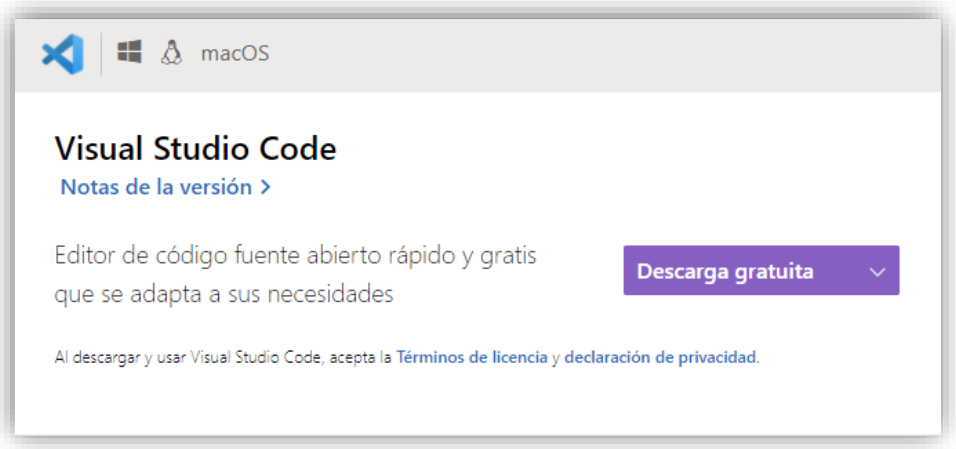
Por freeCodeCamp

Este tutorial lo he realizado sobre un magnifico tutorial realizado por Estefania Cassingena Navone.

PERE MANEL VERDUGO ZAMORA
pereverdugo@gmail.com

Para este tutorial vamos a trabajar con Visual Code.

https://visualstudio.microsoft.com/es/downloads/



En este tutorial vamos a crear 6 proyectos prácticos con Python:

- Historias locas.
- Adivina el número.
- El ordenador tiene que adivinar el número que piensas.
- Piedra, papel o tijera.
- El ahorcado.
- El lago ritmo de búsqueda binaria.

Historias locas (Mad Libs)

Tenemos un texto y en él hay párrafos en blanco que se rellenarán haciendo unas historias locas.

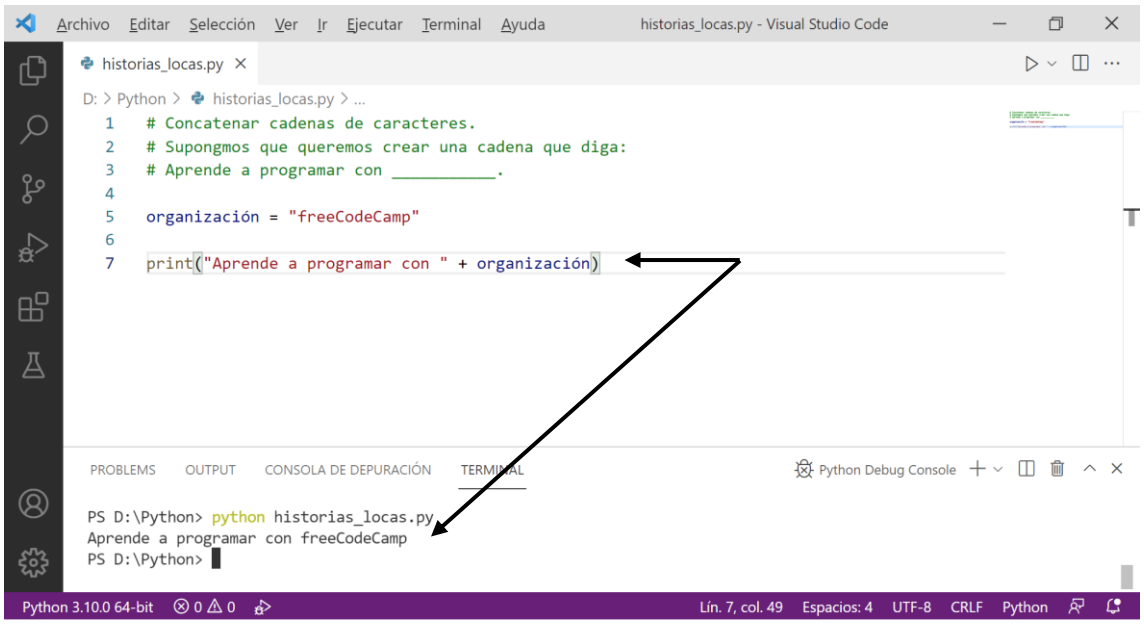
¡Programar es _ _ _ _ _! Siempre me emociona porque me encanta _ _ _ _ _ problemas. ¡Aprenda a _ _ _ _ _ con freeCodeCamp y alcanzar tus _ _ _ _ _!”

¿Cómo concatenar?

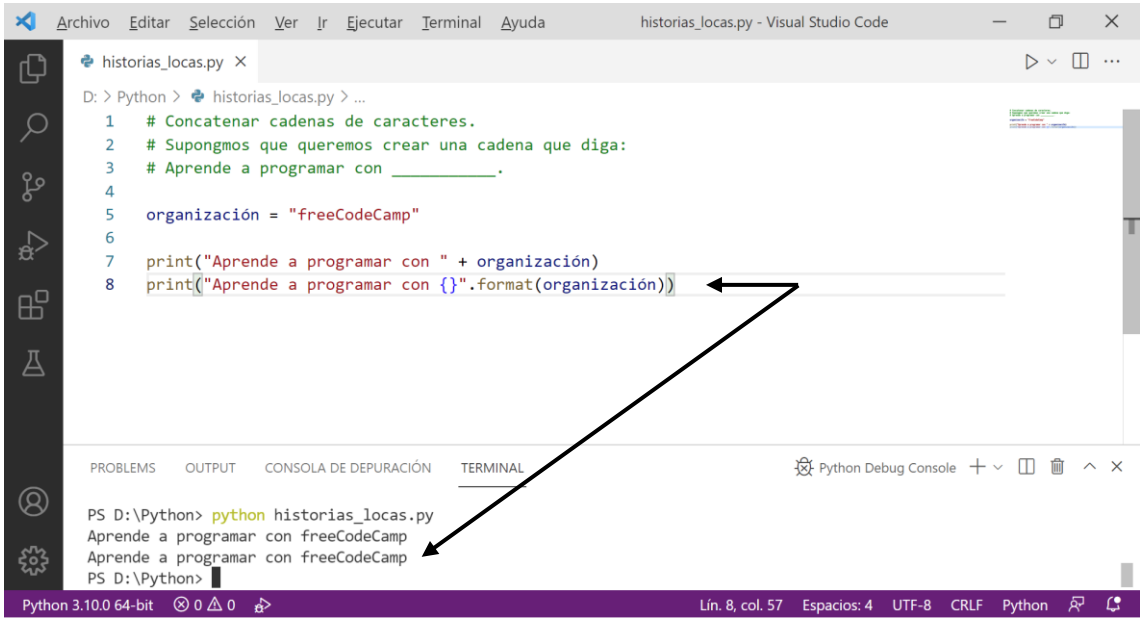
Primero vamos a escribir nuestro código, a continuación lo vamos a guardar en nuestra carpeta de proyectos de Python.

Para ver el resultado en el terminal, parte inferior, si no lo ves pulsaremos (Ctrol + ñ) vamos a ejecutar el proyecto.

python historias_locas.py, este lenguaje para su ejecución necesita un intérprete Python seguido del proyecto que termina con una extensión .py.

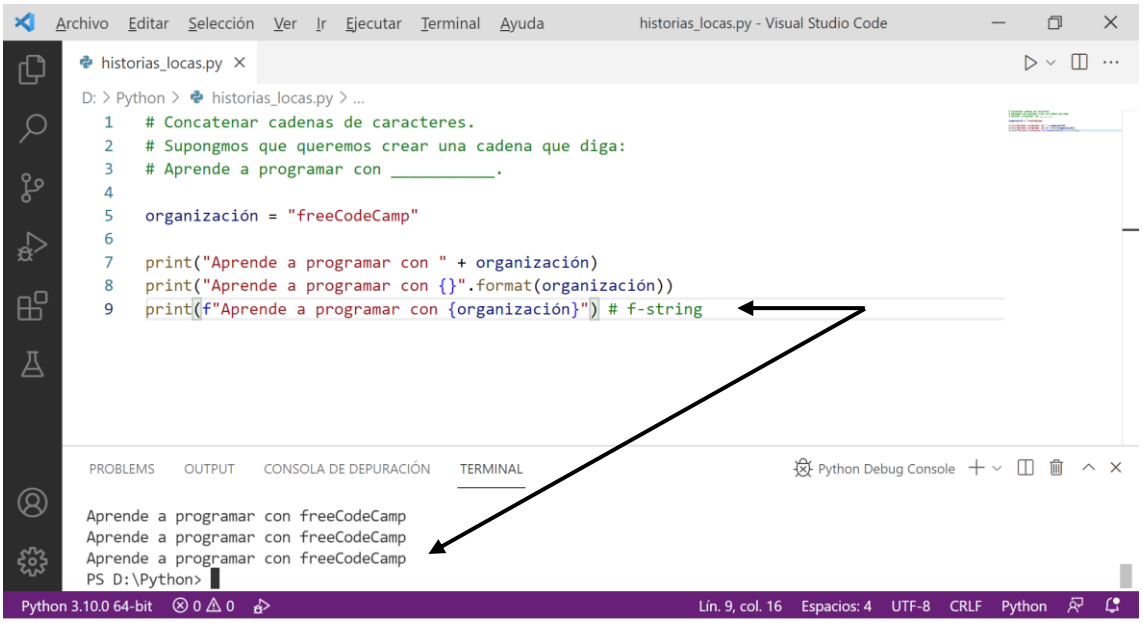


Otra forma de concatenar.



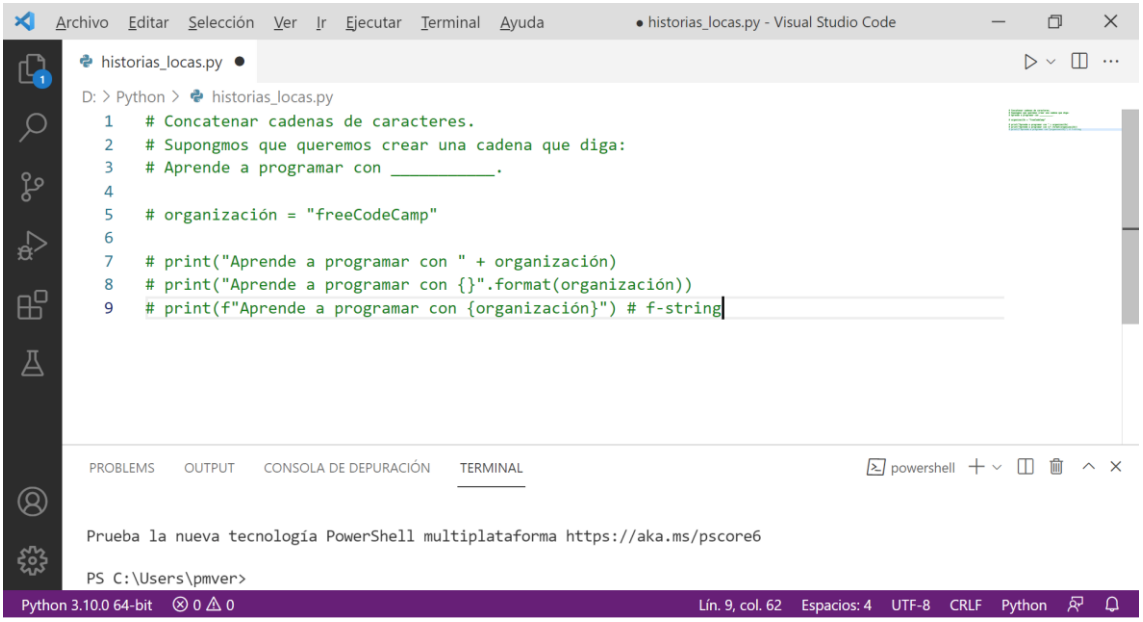
Otra forma.

Para mí la más recomendable (f-String).

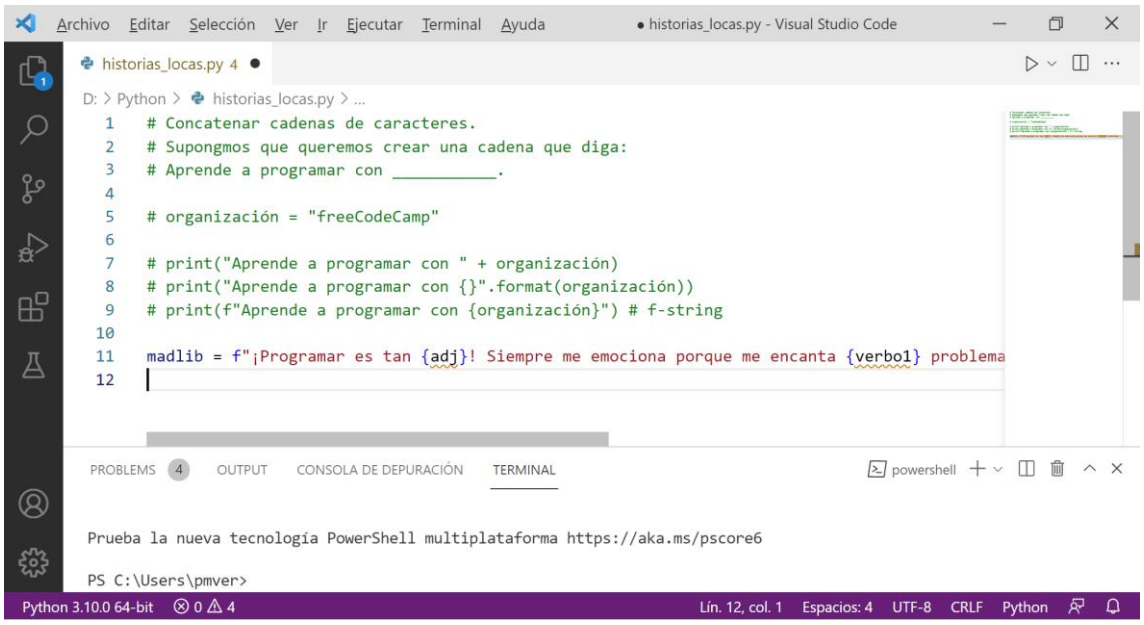


Para comentar el código anterior lo seleccionaremos y haremos Ctrol +/.

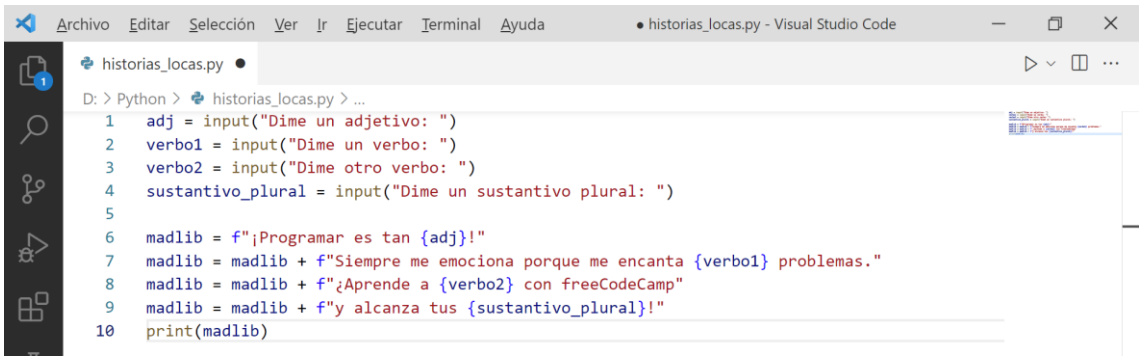
Ahora todo el código lo vamos a comentar con “Ctrol + k” y “Ctrol + c”



Vamos a escribir nuestra historia.



Como podrás observar al ser una línea muy larga la hemos concatenado de la siguiente forma:

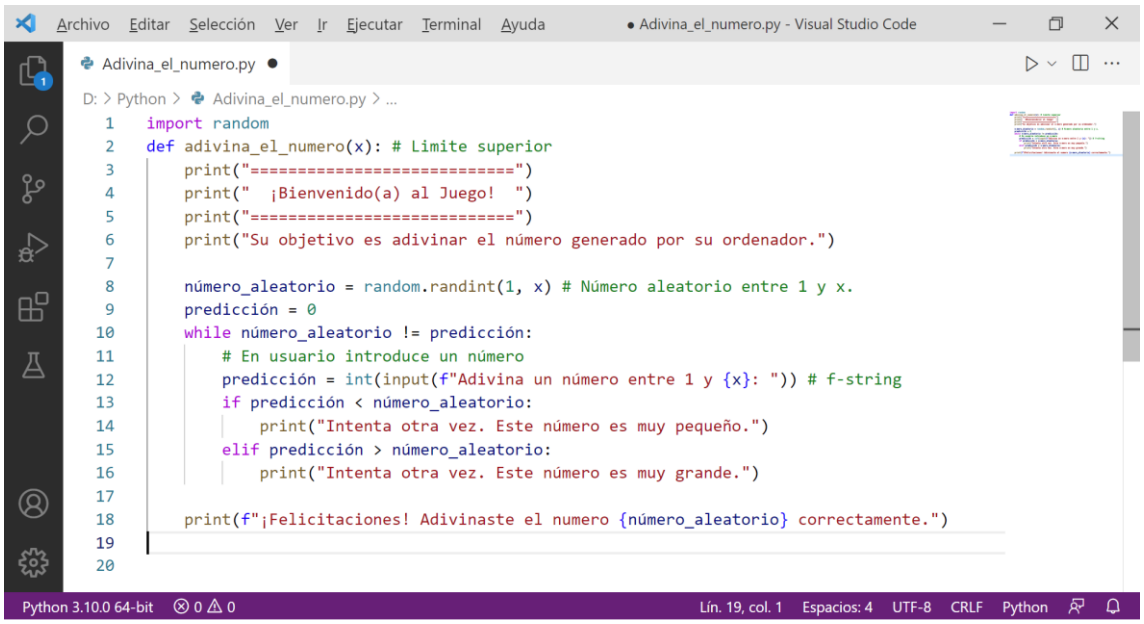


Guardamos los cambios y desde el terminar veremos los resultados.

Si ejecutas el programa y a un amigo le dices que conteste a las preguntas verás que saldrá una frase sin pies ni cabeza.

Adivina el número

Vamos a tener que decir si un número generado por el ordenador es mayor, menor o igual.



Importamos el método random (para calcular el valor aleatorio).

Creamos una función llamada adivina_el_numero(x) que admite un parámetro que será el valor superior.

Las líneas 3, 4, 5 y 6 son mensajes de bienvenida e información al usuario.

Línea 8 a la variable número_aleatorio le vamos a asignar un valor al azar entre 1 y en valor que le pasemos a la función.

Línea 9 definimos la variable predicción a 0, este valor en principio es un valor que no va a salir ya que el valor mínimo es de 1.

Línea 10 utilizamos la instrucción ‘while’ mientras el valor de número_aleatorio sea distinto del valor de predicción estaremos un bucle.

Línea 12 a la variable predicción le pasamos el valor en modo entero ‘int’ de la información que el usuario introduce por teclado ‘input’ y además incluimos desde el 1 hasta {x} el valor más alto que hemos pasado al llamar a esta función.

Línea 13 si predicción en menor a número_aleatorio en la línea 14 nos dirá que el número es demasiado pequeño.

Línea 15 si la condición anterior no se cumple compara esta nueva condición ‘elif’ si no en la línea 16 compara si predicción es mayor que número_aleatorio, en este caso el mensaje será que el número es demasiado grande.

En la línea 18 cuando acierte el número aparecerá un mensaje de felicitaciones diciendo que adiviné le número.

Ahora solo nos queda llamar a la función pasándole el valor máximo.

Adivina_el_numero(100)

PROBLEMAS OUTPUT CONSOLA DE DEPURACIÓN TERMINAL

```
============================
  ¡Bienvenido(a) al Juego!
============================
Su objetivo es adivinar el número generado por su ordenador.
Adivina un número entre 1 y 100: 50
Intenta otra vez. Este número es muy grande.
Adivina un número entre 1 y 100: 25
Intenta otra vez. Este número es muy pequeño.
Adivina un número entre 1 y 100: 40
Intenta otra vez. Este número es muy grande.
Adivina un número entre 1 y 100: 30
Intenta otra vez. Este número es muy pequeño.
Adivina un número entre 1 y 100: 35
Intenta otra vez. Este número es muy pequeño.
Adivina un número entre 1 y 100: 38
Intenta otra vez. Este número es muy grande.
Adivina un número entre 1 y 100: 37
¡Felicitaciones! Adivinaste el numero 37 correctamente.
PS D:\python>
```

Adivina el número (Computadora)

```
1   import random
2   def adivina_el_numero_computadora(x):
3       print("==========================")
4       print(" ¡Bienvenido(a) al juego! ")
5       print("==========================")
6       print(f"Selecciona un número entre 1 y {x} para que la computadora intente adivinarlo.
        .. ")
7       limite_inferior = 1
8       limite_superior = x
9       respuesta = ""
10      while respuesta != "c":
11          # Generar predicción
12          if limite_inferior != limite_superior: #[1, 10]
13              predicción = random.randint(limite_inferior, limite_superior)
14          else:
15              predicción = limite_inferior # también podría ser el límite superior.
16          # obtener una respuesta del usuario
17          respuesta = input(f"Mi predicción es {predicción}. Si es muy alta, ingresa (A).
            Si es baja, ingresa (B). Si es correcta, ingresa (C)").lower()
18          if respuesta == "a":
19              limite_superior = predicción -1
20          elif respuesta == "b":
21              limite_inferior = predicción + 1
22      print(f"¡Siii! El ordenador adivinó tu número correctamente: {predicción}")
23
24  adivina_el_numero_computadora(100)
```

Línea 1 importamos el módulo random, que nos permitirá obtener un valor aleatorio.

Línea 2 definimos una función llamada adivina_el_numero_computadora(x) con un parámetro (x) para aportar el valor máximo.

Líneas 3, 4, 5 y 6 mensaje de bienvenida e información para el usuario para que piense un número entre 1 y (x).

Línea 7 el limite_inferior le asignamos el valor de 1.

Línea 8 el limite_superior le asignamos el valor de x el que pasamos cuando llamamos a la función.

Línea 9 a la variable Respuesta le asignamos el valor “ “ vacío.

Línea 10 Mientras respuesta sea distinta de “c” estaremos en bucle.

Línea 12 Si limite_inferior es distinto a limite_supeiror entonces.

Línea 13 predicción asume un valor aleatorio entre el limite_inferior y el limie_superior.

Línea 14 si no

Línea 15 predicción es igual al límite_inferior, también podría ser el limite_superior.

Línea 17 La variable respuesta recibirá la letra ‘a’ si el valor mostrado es bajo, la letra ‘b’ si el valor mostrado es alto y la letra ‘c’ si ha acertado el número, con .lower() nos aseguramos que las retorna en minúsculas.

Línea 18 Si respuesta es ‘a’ sabemos que el número mostrado en menor.

Línea 19 Por este motivo le decimos que la variable limite_superior sea igual al valor de predicción – 1 (el valor que te ha dado el ordenador menos 1)

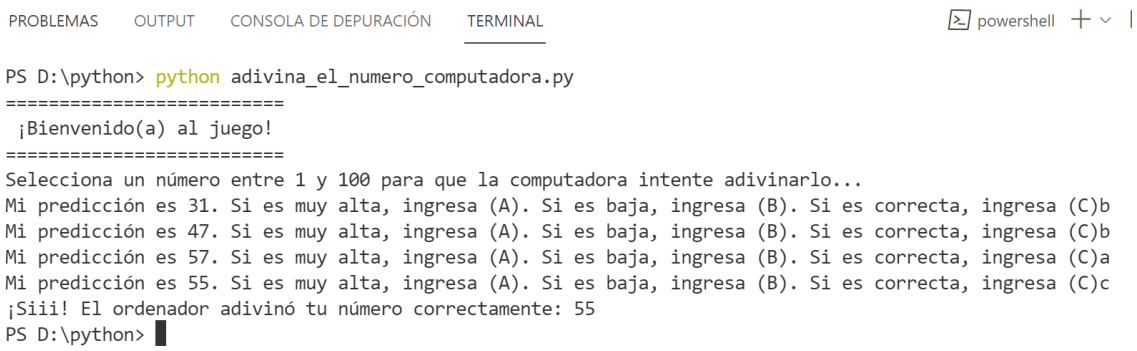
Línea 20 Si no, otra comparación si la respuesta es igual a ‘b’ significa que el numero que mostrado nuestro ordenador es mayor.

Línea 21 Por este motivo le decimos que limite_inferior sea igual al valor de la predicción más 1.

Línea 22 fuera del bucle while, hemos salido porque nuestra respuesta ha sido la ‘c’ el ordenador adivinó el número y muestra su valor.

Línea 24 llamamos a la función adivina_el_numero_computadora(100), le damos como valor máximo 100.

Vamos a ejecutar y he pensado con el número 55.



Piedra, Papel o Tijera

```
1   import random
2
3
4   def jugar():
5       usuario = input("Escoge una opción: 'pi' para pieda, 'pa' para papel y 'ti' para
        tijera.\n").lower()
6       computadora = random.choice(['pi', 'pa', 'ti'])
7
8       if usuario == computadora:
9           return '¡Empate!'
10
11      if ganó_el_jugador(usuario, computadora):
12          return '¡Ganaste!'
13
14      return '¡Perdiste!'
15
16
17  def ganó_el_jugador(jugador, oponente):
18      # Retorna True (verdadero) si gana el jugador.
19      # Piedra gana a Tijera (pi > ti).
20      # Tijera gana a Papel (ti > pa).
21      # Papel gana a la Piedra (pa > pi).
22      if((jugador == 'pi' and oponente == 'ti') or (jugador == 'ti' and oponente == 'pa')
        or (jugador == 'pa' and oponente == 'pi')):
23          return True
24      else:
25          return  False
26
27  print(jugar())
```

Línea 1 importamos el módulo random.

Línea 4 definimos la función jugar() esta no tiene parámetros.

Línea 5 En la variable usuario almacenamos un valor que el jugador introducirá por teclado, las respuestas tienen que ser ‘pi’ para piedra, ‘pa’ para paple y ‘ti’ para tijeras, la función .lower() hace que la respuesta siempre sea en minúsculas dependiendo de cómo tengas el teclado.

Línea 6 Computadora asume un valor aleatorio de la lista [‘pi’, ‘pa’, ‘ti’].

Línea 8 si usuario y computadora son iguales

Línea 9 Retorne el mensaje ‘¡Empate!’.

Línea 11 Si gano_el_jugador(usuario, computadora).

Línea 12 retorna ‘¡Ganaste!’.

Línea 14 retorna ‘¡Perdiste!’ en una función y se ejecuta un return el siguiente no lo ejecuta.

Línea 17 Creamos la función ganó_el_jugador(jugador, oponente) función con dos parámetros.

Línea 22 si jugador es igual a ‘pi’ y oponente igual a ‘ti’

- jugador es igual a ‘ti’ y oponente igual a ‘pa’
- jugador es igual a ‘pa’ y oponente igual a ‘pi’

Línea 23 retorna True.

Línea 24 si no.

Línea 25 retorna False.

Línea 27 se llama a la función jugar() sin parámetros.

Este es el resultado:

```
PROBLEMAS   OUTPUT   CONSOLA DE DEPURACIÓN   TERMINAL
Escoge una opción: 'pi' para pieda, 'pa' para papel y 'ti' para tijera.
pi
¡Ganaste!
PS D:\python> python piedra_papel_tijera.py
Escoge una opción: 'pi' para pieda, 'pa' para papel y 'ti' para tijera.
pa
¡Perdiste!
PS D:\python> python piedra_papel_tijera.py
Escoge una opción: 'pi' para pieda, 'pa' para papel y 'ti' para tijera.
ti
¡Empate!
```

El ahorcado

Para este proyecto vamos a utilizar varios archivos.

Un archivo llamado palabras.py donde almacenarás todas las palabras que vas a utilizar en el juego, evita palabras que tengan la letra 'Ñ'. Más adelante comentaremos porque.

```
palabras = ["aire", "ojos", "piel", "anteojos", "joven", "viejo", "alto", "bajo",
"pequeño", "gordo", "delgado", "bella", "azul", "verde", "negro", "sombrero", "guantes",
"corbata", "gemelos", "paraguas", "plata", "oro", "perla", "diamante", "esmeralda",
"anillo", "pulsera", "reloj", "elegante", "sencillo", "chaqueta", "traje", "camisa",
"zapatos", "pelo", "maquillaje", "peine", "dedo", "hueso", "cara", "ojo", "calor",
"ambulancia", "enfermera", "farmacia", "vitaminas", "pastillas", "dentista", "ciego",
"correr", "caminar", "regresar", "saltar", "fin", "cerrar", "nombre", "mujer", "hombre",
"soltero", "novio", "nacer", "vivir", "edad", "anciana","trabajar", "cobrar", "azafata",
"artista", "panadero", "carpintero", "cocinero", "maestro", "bombero", "juez", "modelo",
"monje", "pintor", "piloto", "secretaria", "taxista", "escritor", "jefe", "aprendiz",
"jubilado", "empleo", "industria", "taller", "tienda", "vacaciones", "sueldo"]
```

A continuación un segundo archivo llamado Ahorcado_diagrama.py que contendrá el dibujo del ahorcado para irlo mostrando según las vidas que te quedan.

```
vidas_diccionario_visual = {
    0: """
       ___________
      | /         |
      |/        ( )
      |          |
      |         / \\
      |
    """,
    1: """
       ___________
      | /         |
      |/        ( )
      |          |
      |         /
      |
     """,
    2: """
       ___________
      | /         |
      |/        ( )
      |          |
      |
      |
     """,
```

```
    3: """
        +---+
        |   |
        O   |
       /|\  |
       / \  |
            |
      =========""",
    4: """
        +---+
        |   |
        O   |
       /|\  |
       / \  |
            |
      =========""",
    5: """
        +---+
        |   |
        O   |
       /|\  |
       / \  |
            |
      =========""",
    6: """
        +---+
        |   |
        O   |
       /|\  |
       / \  |
            |
      =========""",
    7: "",
}
```

Ahora vamos con el archivo principal con el nombre Ahorcado.py.

```
1    import random
2    import string
```

Importamos las módulo random y string, el modulo random ya lo hemos utilizado en proyectos anteriores.

El módulo string nos permite trabajar con determinadas cadenas de caracteres.

```
4    from palabras import palabras
5    from Ahorcado_diagrama import vidas_diccionario_visual
```

Para poder agregar las palabras a nuestro proyecto.

Para reconocer la variable vidas_diccionario_visual una lista con el valor de vidas que te quedará en el juego con su correspondiente dibujo.

```
 7  def obtener_palabra_válida(palabras):
 8      # Seleccionar una palabra al azar de la lista
 9      # de palabras válidas.
10      palabra = random.choice(palabras)
11      while '-' in palabra or ' ' in palabras:
12          palabra = random.choice(palabras)
13      return palabra.upper()
```

La función obtener_palabra_válidad(palabras) nos permite seleccionar de la lista palabras una aleatoriamente con la instrucción ‘random.choice(palabras) que ya hemos utilizado en ejercicios anteriores.

Línea 11 mientras la palabra contenga un ‘-‘ o un ‘ ‘ espacio en blanco.
Línea 12 búscame otra palabra.
Línea 13 me la retornas en mayúsculas.

```
15  def ahorcado():
16      print("=======================================")
17      print(" ¡Bienvenido(a) al juego del Ahorcado! ")
18      print("=======================================")
19      palabra = obtener_palabra_válida(palabras)
20      letras_por_adivinar = set(palabra)
21      abecedario = set(string.ascii_uppercase)
22      letras_adivinadas =set()
23      vidas = 7
24      while len(letras_por_adivinar) > 0 and vidas > 0 :
25          # Letras adivinadas
26          # ' '.join({'A', 'B', 'C'}) -> 'A B C'
27          print(f"Te quedan {vidas} vidas y has usado estas letras: {' '.join
            (letras_adivinadas)}")
```

Definimos la función ahorcado(): sin parámetros.
Líneas 16, 17 y 18 presentación de bienvenida.
Línea 19 Palabra obtiene de la función obtener_palabra_válida(palabras) la palabra aleatoria de la lista de palabras del documento palabras.py.

Línea 20 La variable letras_por_adivinar obtine las letras de dicha palabra, pero si la palabra repite alguna letra aquí no la muestra.

Un ejemplo:

```
1    import string
2
3    print(set("Cocodrilo"))
```

Resultado:

```
{'C', 'd', 'o', 'i', 'c', 'l', 'r'}
```

Este ejemplo no tiene nada que ver con el proyecto es para que entiendas el argumento de la línea 20.

Línea 21 En la variable abecedario se almacenan todas las letras a excepción de la letra 'Ñ' por lo que comentábamos que ninguna palabra tuviera la letra 'Ñ'.

Ejemplo:

```
1    import string
2
3    print(set(string.ascii_uppercase))
```

Resultado:

```
{'V', 'W', 'J', 'U', 'E', 'A', 'D', 'O', 'L', 'G', 'T', 'C', 'B', 'F', 'M', 'N', 'Q', 'I', 'K', 'X', 'R', 'H', 'P',
 'Y', 'Z', 'S'}
```

Este ejemplo no tiene nada que ver con el proyecto es para que entiendas el argumento de la línea 21.

Línea 22 La variable letras_adivinadas al principio están vacías.
Línea 23 la variable vidas tiene el valor de 7 (como el juego del ahorcdo).
Línea 24 Mientras la longitud de letras_por_adivinar sea mayor de 0 y vidas mayor a 0 tendremos un bucle.
Línea 27 Muestra el número de vidas que me queda y las letras que he usado.

Ejemplo:

```
1    import string
2    Palabra = "Cocodrilo"
3    print('-'.join(Palabra))
```

Resultado:

```
C-o-c-o-d-r-i-l-o
```

Este ejemplo no tiene nada que ver con el proyecto es para que entiendas el argumento de la línea 27.

```
28          # Mostrar el estado actual de la palabra
29          # H - L A
30          palabra_lista = [letra if letra in letras_adivinadas else '-' for letra in
            palabra]
31          # Mostrar estado del ahorcado
32          print(vidas_diccionario_visual[vidas])
33          # Mostrar las letras separadas por un espacio.
34          print(f"Palabra: {' '.join(palabra_lista)}")
35
36          letra_usuario = input("Escoge una letra: ").upper()
```

Línea 30 La variable palabra_lista será igual a la letra está entre las adivinadas se mostrará de lo contrario se mostrará un guión, el ejemplo se encuentra en la línea 29.

Ejemplo:

```
1   vocales = ['a', 'e', 'i', 'o', 'u']
2   palabra = "abecedario"
3   texto = [letra if letra in vocales else '-' for letra in palabra]
4   print(texto)
```

Resultado:

```
['a', '-', 'e', '-', 'e', '-', 'a', '-', 'i', 'o']
```

Este ejemplo no tiene nada que ver con el proyecto es para que entiendas el argumento de la línea 30.

Línea 32 muestra el dibujo del ahorcado según las vidas que te queda, ya que es una lista con las distintas formas de según las vidas que te queda.
Línea 34 vuelve a mostrar la palabra para que sigas adivinando letras.
Línea 36 el usuario introduce una nueva letra.

```
37          # Si la letra escogida por el usuario está en el
38          # abecedario y no está en el conjunto de letras
39          # que ya se han ingresado, se añade la letra al conjunto
40          # de letras ingresadas.
41          if letra_usuario in abecedario - letras_adivinadas:
42              letras_adivinadas.add(letra_usuario)
43
44              # Si letra está en la palabra,
45              # quitar la letra del conjunto de letras
46              # pendiente por adivinar.
47              # Si no está en la palabra, quitar una vida.
48              if letra_usuario in letras_por_adivinar:
49                  letras_por_adivinar.remove(letra_usuario)
50                  print('')
51              else:
52                  vidas = vidas - 1
53                  print(f"\nTu letra, {letra_usuario} no está en la palabra.")
```

Línea 41 Si letra_usuario está en el abecedario – letras adivinadas.

Línea 42 esta se agrega a la lista letras_adivinadas.

Línea 48 Si letra_usuario se encuentra en letras_por_adivinar.

Línea 49 de la lista letras_por_adivinar eliminamos la letra_usuario.

Línea 50 imprime una línea en blanco.

Línea 51 si se cumple con lo anterior.

Línea 52 a la variable vidas se le resta 1.

Línea 53 muestra un menaje de que dicha letra {letra_usuario} no está en la palabra.

```
54          # Si la letra escogida por el usuario ya fue ingresada.
55          elif letra_usuario in letras_adivinadas:
56              print("\nYa escogiste esa letra. Por favor escoge una letra nueva.")
57          else:
58              print("\nEsta letra no es válida.")
59      # El juego llega a esta línea cuando se adivinan todas las
60      # letras de la palabra o cuando se agotan las vidas del
61      # jugador.
62      if vidas == 0:
63          print(vidas_diccionario_visual[vidas])
64          print(f"¡Ahorcado! Perdiste. Lo lamento mucho. La palabra era: {palabra}")
65      else:
66          print(f"¡Excelente! ¡Adivinaste la palabra {palabra}!")
67
68  ahorcado()
```

Línea 55 Si no si letra_usuario está en letras_adivinadas.
Línea 56 esta letra ya las elegido.
Línea 57 si no.
Línea 58 un mensaje 'Esta letra no es válida'.
Línea 62 si vidas es igual a 0.
Línea 63 muestra el dibujo del ahorcado para vidas igual a 0.
Línea 64 muestra un mensaje de perdiste y te dice cuál era la palabra.
Línea 65 si no.
Línea 66 muestra mensaje adivinaste la palabra y muestra dicha palabra.
Línea 68 llamamos a la función ahorcado()

Este será el resultado:

```
=====================================
 ¡Bienvenido(a) al juego del Ahorcado!
=====================================
Te quedan 7 vidas y has usado estas letras:

Palabra: - - - - -
Escoge una letra:

Esta letra no es válida.
Te quedan 7 vidas y has usado estas letras:

Palabra: - - - - -
PS D:\python> python Ahorcado.py
=====================================
 ¡Bienvenido(a) al juego del Ahorcado!
=====================================
Te quedan 7 vidas y has usado estas letras:

Palabra: - - - - -
Escoge una letra: a

Tu letra, A no está en la palabra.
Te quedan 6 vidas y has usado estas letras: A
```

```
|
|
|
|
|
```

Palabra: - - - - -
Escoge una letra: e

Te quedan 6 vidas y has usado estas letras: A E

```
|
|
|
|
|
```

Palabra: - - - - E
Escoge una letra: i

Tu letra, I no está en la palabra.
Te quedan 5 vidas y has usado estas letras: A I E

```
 _____
| /
|/
|
|
|
```

Palabra: - - - - E
Escoge una letra: o

Te quedan 5 vidas y has usado estas letras: O A I E

```
 _____
| /
|/
|
|
|
```

Palabra: - O - - E
Escoge una letra: b

Te quedan 5 vidas y has usado estas letras: O A E B I

```
 _____
```

```
| /
|/
|
|
|
```

Palabra: - O B - E
Escoge una letra: s

Tu letra, S no está en la palabra.
Te quedan 4 vidas y has usado estas letras: O S A E B I

```
 _________
| /       |
|/
|
|
|
```

Palabra: - O B - E
Escoge una letra: c

Tu letra, C no está en la palabra.
Te quedan 3 vidas y has usado estas letras: O S C A E B I

```
 _________
| /       |
|/      ( )
|
|
|
```

Palabra: - O B - E
Escoge una letra: r

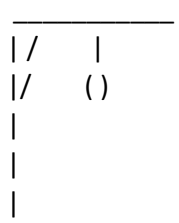
Te quedan 3 vidas y has usado estas letras: O S C A E B R I

```
 _________
| /       |
|/      ( )
|
|
|
```

Palabra: - O B R E
Escoge una letra: c

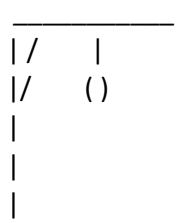
Ya escogiste esa letra. Por favor escoge una letra nueva.

Te quedan 3 vidas y has usado estas letras: O S C A E B R I



Palabra: - O B R E
Escoge una letra: s

Ya escogiste esa letra. Por favor escoge una letra nueva.
Te quedan 3 vidas y has usado estas letras: O S C A E B R I



Palabra: - O B R E
Escoge una letra: p

¡Excelente! ¡Adivinaste la palabra POBRE!

Búsqueda Binaria

Vamos a implementar el algoritmo de búsqueda binaria.

Búsqueda ingenua (Naive Search)

```
1   import random
2   import time
3
4   def búsqueda_ingenua(lista, objetivo):
5       for i in range(len(lista)):
6           # range(len(lista))  -> 0, 1, 2, 3, ..., len(lista)-1
7           if lista[i] == objetivo:
8               return i
9       return -1
10
11  mi_lista = [1, 3, 5, 10, 12]
12
13  print(búsqueda_ingenua(mi_lista, 10))
```

En la línea 4 creamos una función llamada búsqueda_ingenua(lista, objetivo): con dos parámetros lista y objetivo.

Línea 5 hacemos un ciclo for que empezará desde 0 hasta el número de elementos.

Línea 7 si lista[i] i irá empezando por 1 y en cada ciclo incrementará 1 hasta llegar al número de elementos de la lista, y compara si es igual al objetivo, un determinado elemento de la lista.

Línea 8 si es así que retorne i (valor del incide).

Línea 9 si una vez terminado todos los bucles no ha encontrado dicho elemento que retorne el valor -1, este valor no va a coincidir con ningún número de índice de la lista, porque empieza por 0 y va incrementando en 1.

Línea 11 creamos la lista llamada mi_lista con una serie de valores.

Línea 13 mostramos en pantalla el resultado de llamar a la función búsqueda_ingenua(mi_lista, objetivo) mi_lista es el nombre de la lista y objetivo el elemento que queremos consultar.

Si ejecutamos este será el resultado:

```
PS D:\python> python busqueda_binaria.py
3
```

El elemento número 10 se encuentra en la posición 3, recuerda que comienza por la posición 0.

```
11  mi_lista = [1, 3, 5, 10, 12]
12
13  print(búsqueda_ingenua(mi_lista, 15))
```

Si buscamos un valor que no está en la lista este será el resultado:

```
PS D:\python> python busqueda_binaria.py
-1
```

Retorna el valor -1.

Búsqueda binaria.

```
10
11  lista = [1, 3, 7, 9, 12, 18, 23, 26, 45]
12
13  def búsqueda_binaria(lista,objetivo,limite_inferior = None, limite_superior = None):
14      if limite_inferior is None:
15          limite_inferior = 0 # Inicio de la lista
16      if limite_superior is None:
17          limite_superior = len(lista)-1 # Final de la lista
18      if limite_superior < limite_inferior:
19          return -1
20
21      punto_medio = (limite_inferior + limite_superior) // 2
22
23      if lista[punto_medio] == objetivo:
24          return punto_medio
25      elif objetivo < lista[punto_medio]:
26          return búsqueda_binaria(lista, objetivo, limite_inferior, punto_medio-1)
27      else:
28          return búsqueda_binaria(lista, objetivo, punto_medio + 1, limite_superior)
29
30  print(búsqueda_binaria(lista, 18))
```

Línea 11 tenemos la lista que debemos consultar.

Línea 13 definimos la función búsqueda_binaria(lista, objetivo, limite_inferior = None, limite_superior = None): admite 4 parámetros pero podemos omitir los dos últimos ya que si no los introducimos la función determinará el valor.
Línea 14 si límite_inferior es igual a None
Línea 15 límite_inferior = 0
Línea 16 si límite_superior es igual a None
Línea 17 limite_superior es igual a el número de elementos de la lista menos 1.
Línea 18 si limite_superior es menor a limite_inferior
Línea 19 retorna -1
Línea 21 calcula el punto medio de la lista, sumando limite_inferior más limite_superior división entero 2.
Línea 23 si lista[punto_medio] es igual a objetivo
Línea 24 retorna punto_medio.
Línea 25 si no si objetivo es menor de lista[punto_medio]
Línea 26 se llama a la función así misma pero cambiando el valor de los parámetros, lista y objetivo siguen siendo los mismos pero limitamos la búsqueda desde limite_inferior hasta punto_medio -1.
Línea 27 si no.
Línea 28 se llama a la función así misma pero cambiando el valor de los parámetros, lista y objetivo siguen siendo los mismos pero limitamos la búsqueda desde punto_medio + 1 hasta limite_superior.

Línea 30 fuera de la función queremos mostrar el contenido de la función búsqueda_binaria(lista, valor 18).

Este será el resultado:

```
PS D:\python> python busqueda_binaria.py
5
```

Hemos agregado el siguiente código:

```
30 if __name__=='__main__':
31     mi_lista = [1, 3, 5, 10, 12]
32     print(búsqueda_binaria(mi_lista, 12))
```

Línea 30 esta condición simplemente hace una consulta para verificar si el archivo que se está ejecutando es el archivo principal o raíz del programa. En Python el método __name__ es el nombre de tu archivo ejemplo: búsqueda_binaria.py ella toma __name__ el nombre de tu archivo.py y lo compara con el __main__ que siempre será para Python el archivo principal, el archivo de mayor jerarquía.

Línea 31 creamos una lista con valores.

Línea 32 Mostramos el resultado de llamar a la función búsqueda_binaria(mi_lista, 12))

Este será el resultado:

```
PS D:\python> python busqueda_binaria.py
4
```

```
30  if __name__=='__main__':
31
32      # Crear una lista ordenada con 10000 números aleatorios.
33      tamaño = 10000
34      conjunto_inicial = set()
35      while len(conjunto_inicial) < tamaño:
36          conjunto_inicial.add(random.randint(-3*tamaño, 3*tamaño))
37
38      lista_ordenada = sorted(list(conjunto_inicial))
39
40      # Medir el tiempo de búsqueda ingenua.
41      inicio = time.time()
42      for objetivo in lista_ordenada:
43          búsqueda_ingenua(lista_ordenada, objetivo)
44      fin = time.time()
45      print(f"Tiempo de búsqueda ingenua: {fin - inicio} segundos.")
46
47      # Medir el tiempo de búsqueda binaria.
48      inicio = time.time()
49      for objetivo in lista_ordenada:
50          búsqueda_binaria(lista_ordenada, objetivo)
51      fin = time.time()
52      print(f"Tiempo de búsqueda binaria: {fin - inicio} segundos.")
```

Línea 30 programa principal.
Línea 33 definimos la variable tamaño con el valor de 10.000.
Línea 34 conjunto_inicial está vacío.
Línea 35 mientras el tamaño del conjunto_inicial sea menor de 10.000.
Línea 36 Al conjunto_incial se le añade un valor aleatorio desde -30.000 hasta 30.000, los valores repetidos no se añaden.
Línea 38 a la variable lista_ordenada le pasamos los datos de conjunto_inicial ordenada.
Línea 41 la variable inicio toma el valor del tiempo.
Línea 42 un ciclo for donde la variable objetivo empezará desde 0 hasta el número de elementos de la lista_ordenada.
Línea 43 llamamos a la función búsqueda_ingenua(lista_ordenada, objetivo)
Línea 44 la variable fin toma el valor del tiempo.
Línea 45 muestra en pantalla el tiempo empleado en segundos.
Línea 48 la variable inicio toma el valor del tiempo.
Línea 49 un ciclo for donde la variable objetivo empezará desde 0 hasta el número de elementos de la lista_ordenada.
Línea 50 llamamos a la función búsqueda_binaria(lista_ordenada, objetivo).
Línea 51 la variable fin toma el valor del tiempo.
Línea 52 muestra en pantalla el tiempo empleado en segundos.

Este será el resultado:

```
PS D:\python> python busqueda_binaria.py
Tiempo de búsqueda ingenua: 2.3616511821746826 segundos.
Tiempo de búsqueda binaria: 0.03094625473022461 segundos.
```

Contenido