



¿Cómo programa Mario en Python 3?



Este tutorial está obtenido de un tutorial en YouTube del canal Mario en Python.

En el código QR tendrás acceso a los 5 videotutoriales de que consta el curso.

Para descargar el material accede al siguiente enlace:

https://github.com/mundo-python/mario_arcade_python

PROGRAMA CON PYTHON UTILIZANDO LA LIBRERÍA ARCADE

PERE MANEL VERDUGO ZAMORA

Web: www.peremanelv.com
pereverdugo@gmail.com

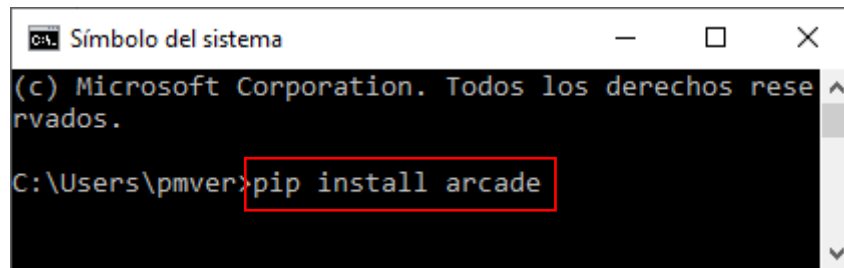
Contenido

1.- Introducción	2
2.- Como programar la ventana.....	3
3.- Con la parte de diseño	5
4.- Movimiento	9
5.- Física (Gravedad)	12

1.- Introducción

Vamos a realizar un simple proyecto de Mario.

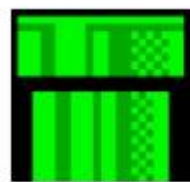
Para instalar la librería arcade, desde Cmd de Windows escribimos:



```
Símbolo del sistema
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\pmver>pip install arcade
```

Automáticamente va a descargar e instalar la librería.

En la portada del tutorial hay un link de descargar para que puedas agregar estos archivos a tu proyecto.



cylinder.png



ground.png



mario.png

Tendrá que ir en la misma carpeta donde vayamos a realizar el proyecto.



2.- Como programar la ventana.

Vamos a crear un nuevo archivo llamado mario.py.

```
1 import arcade
```

```
2
```

```
3 # Constantes
```

```
4 SCREEN_WIDTH = 1000
```

```
5 SCREEN_HEIGHT = 500
```

```
6 SCREEN_TITLE = "Mario Demo"
```

```
7
```

```
8 class Mygame(arcade.Window):
```

```
9     def __init__(self):
```

```
10         super().__init__(SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_TITLE)
```

```
11         arcade.set_background_color(arcade.csscolor.CORNFLOWER_BLUE)
```

```
12
```

```
13     def setup(self):
```

```
14         pass
```

```
15
```

```
16     def on_draw(self):
```

```
17         arcade.start_render()
```

```
18
```

```
19 def main():
```

```
20     window = Mygame()
```

```
21     window.setup()
```

```
22     arcade.run()
```

```
23
```

```
24 if __name__ == "__main__":
```

```
25     main()
```

Definimos tres constantes, para el ancho, alto y título de la ventana.

Creamos la clase Mygame con la super clase arcade.Window.

Cuando llamados a la super clase le pasamos los parámetros de ancho, alto y título de la ventana.

Definimos el color de la ventana.

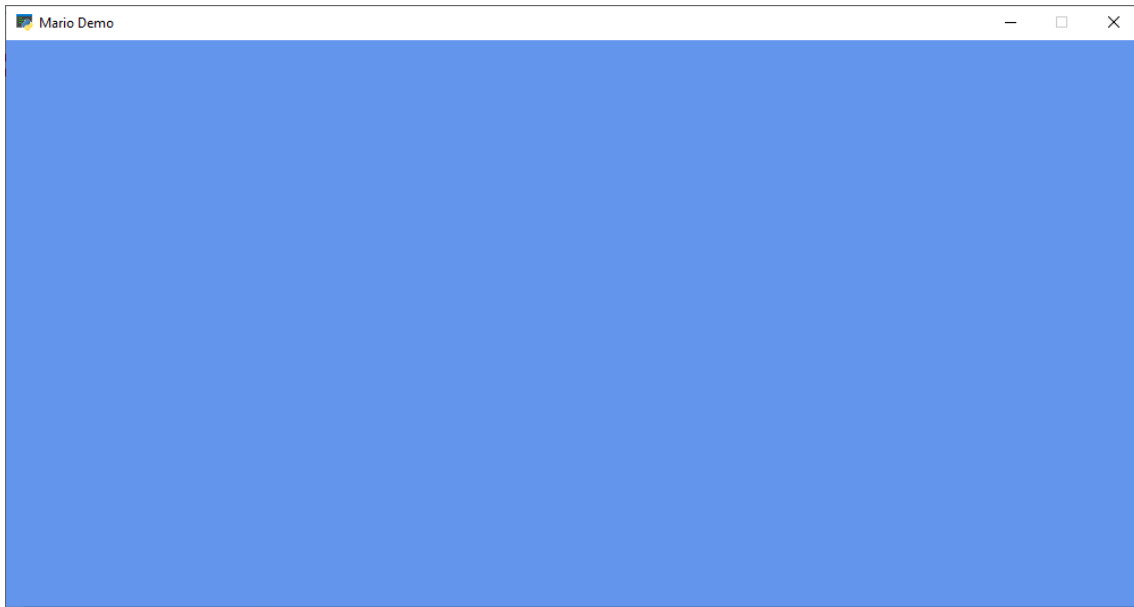
La función setup la utilizaremos para poner nuestros sprites e ir diseñando los niveles.

Con esta función vamos a inicializar el motor que va a ser el que empiece a dibujar o pintar.

Ejecutamos la función principal.

Definimos nuestra función principal, donde creamos la ventana, ejecutará nuestra función setup y además ejecutamos el motor de arcade.

Este será el resultado:



En el siguiente capítulo veremos como podemos agregar a nuestro Mario y los Sprite.

3.- Con la parte de diseño

Lo primero que vamos a hacer es agregar a nuestro personaje.

```
1 import arcade
2
3 # Constantes
4 SCREEN_WIDTH = 1000
5 SCREEN_HEIGHT = 500
6 SCREEN_TITLE = "Mario Demo"
7
8 # Constantes para escalar los sprites
9 CHARTER_SCALING = 0.17
10 GROUND_SCALING = 0.20
11 CYLINDER_SCALING = 0.20
12
13 class Mygame(arcade.Window):
14     def __init__(self):
15         super().__init__(SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_TITLE)
16         arcade.set_background_color(arcade.csscolor.CORNFLOWER_BLUE)
17
18         # Listas que contendran nuestros sprites
19         self.coin_list = None
20         self.wall_list = None
21         self.player_list = None
22
23         # Variable del sprite jugador
24         self.player_sprite = None
25
26     def setup(self):
27         self.player_list = arcade.SpriteList()
28         self.wall_list = arcade.SpriteList()
29         self.coin_list = arcade.SpriteList()
30
31         # Crear el jugador
32         image_source = "mario.png"
33         self.player_sprite = arcade.Sprite(image_source, CHARTER_SCALING)
34         self.player_sprite.center_x = 64
35         self.player_sprite.center_y = 93
36         self.player_list.append(self.player_sprite)
37
38     def on_draw(self):
39         arcade.start_render()
40         self.player_list.draw()
41
42 def main():
43     window = Mygame()
44     window.setup()
45     arcade.run()
46
47 if __name__ == "__main__":
48     main()
```

Vamos a declarar 3 constantes para escalar los Sprites.

Creamos 3 listas que van a contener nuestros Sprites.

Creamos una variable que contendrá el Sprite del jugador.

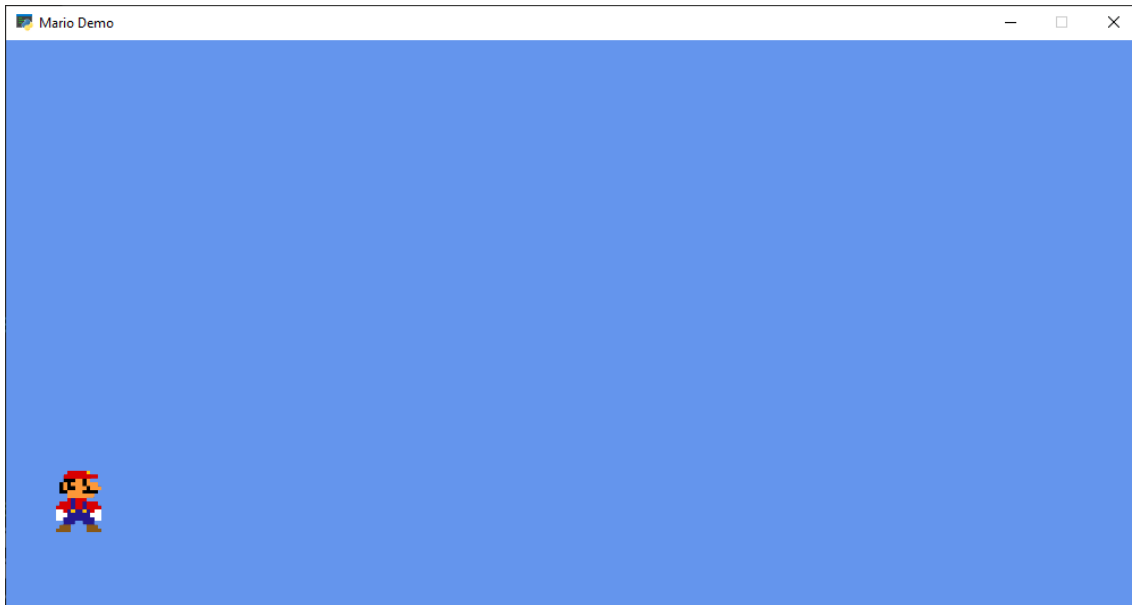
Inicializamos las listas.

El objeto arcade.SpriteList nos va permitir controlar las colisiones.

Agregamos a nuestro jugador.
1 Cargamos la imagen desde nuestra carpeta.
2 La escalamos al porcentaje CHARTER_SCALING.
3 Configuramos la coordenada x.
4 Configuramos la coordenada y.
5 Lo añadimos a la lista player_list.

Pinta el personaje.

Este será el resultado:



Ahora vamos a agregar el suelo.

```
26     def setup(self):
27         self.player_list = arcade.SpriteList()
28         self.wall_list = arcade.SpriteList()
29         self.coin_list = arcade.SpriteList()
30
31         # Crear el jugador
32         image_source = "mario.png"
33         self.player_sprite = arcade.Sprite(image_source, CHARTER_SCALING)
34         self.player_sprite.center_x = 64
35         self.player_sprite.center_y = 93
36         self.player_list.append(self.player_sprite)
37
38         # Crear el suelo
39         for x in range(0, 1250, 64):
40             wall = arcade.Sprite("ground.png", GROUND_SCALING)
41             wall.center_x = x
42             wall.center_y = 32
43             self.wall_list.append(wall)
```

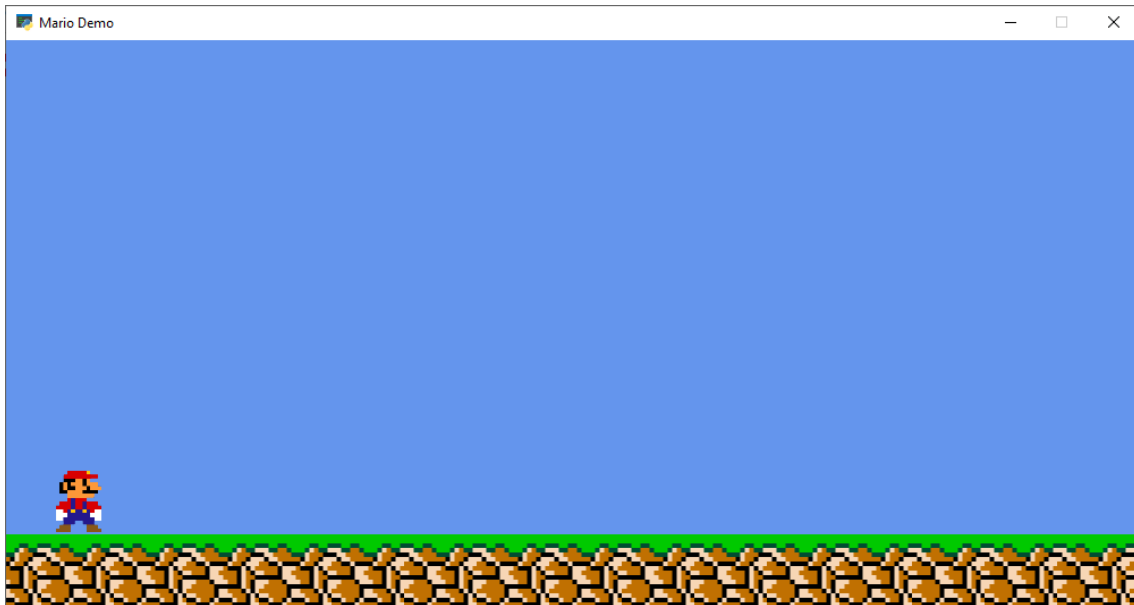
Hacemos un bucle for para pintar el suelo para ello x tendrá los valores 0 hasta 1250 con un incremento de 64, esto nos permite dibujar todo el suelo.

Cargamos la imagen ground.png que utilizamos para el suelo, le damos las coordenadas de x e y, por último lo agregamos a una lista (wall_list).

```
45     def on_draw(self):
46         arcade.start_render()
47         self.player_list.draw()
48         self.wall_list.draw()
```

Dibujamos la lista wall_list que contiene todos los Sprites del suelo con sus respectivas coordenadas.

Este será el resultado:



Ahora vamos a crear los cilindros.

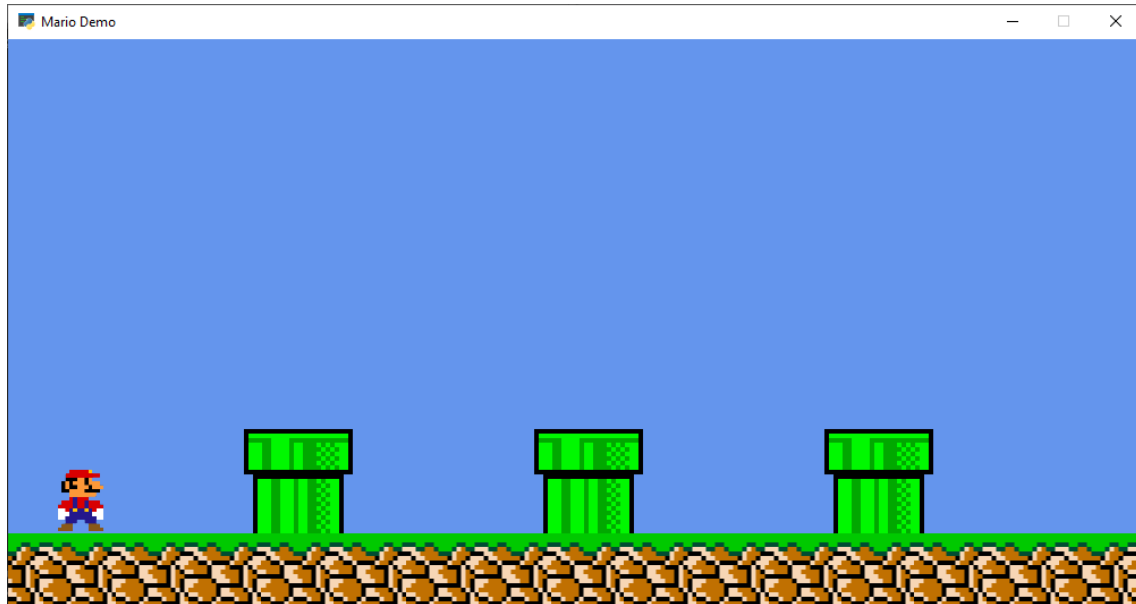
```
31     # Crear el jugador
32     image_source = "mario.png"
33     self.player_sprite = arcade.Sprite(image_source, CHARTER_SCAI
34     self.player_sprite.center_x = 64
35     self.player_sprite.center_y = 93
36     self.player_list.append(self.player_sprite)
37
38     # Crear el suelo
39     for x in range(0, 1250, 64):
40         wall = arcade.Sprite("ground.png", GROUND_SCALING)
41         wall.center_x = x
42         wall.center_y = 32
43         self.wall_list.append(wall)
44
45     # Crear los cilindros con una lista.
46     coordinate_list = [[512, 110], [256, 110], [768, 110]]
47
48     for coordinate in coordinate_list:
49         wall = arcade.Sprite("cylinder.png", CYLINDER_SCALING)
50         wall.position = coordinate
51         # Añadiendo a la lista
52         self.wall_list.append(wall)
```

Creamos una lista de 3 elementos que a su vez tiene otra lista de 2 elementos con las coordenadas de los cilindros.

Con un ciclo for recorreremos toda la lista de coordenadas y vamos creando cilindros con sus respectivas coordenadas que al final lo añadimos a la lista `wall_list`.


```
54  def on_draw(self):  
55      arcade.start_render()  
56      self.player_list.draw()  
57      self.wall_list.draw()
```

← Dibujamos los cilindros.



4.- Movimiento

Estas has sido las modificaciones:

```
import arcade
```

```
# Constantes
```

```
SCREEN_WIDTH = 1000
```

```
SCREEN_HEIGHT = 500
```

```
SCREEN_TITLE = "Mario Demo"
```

```
# Constantes para escalar los sprites
```

```
CHARTER_SCALING = 0.17
```

```
GROUND_SCALING = 0.20
```

```
CYLINDER_SCALING = 0.20
```

```
# Velocidad del jugador
```

```
PLAYER_MOVEMENT_SPEED = 5
```

Declaramos una constante para la velocidad del jugador.

```
class Mygame(arcade.Window):
```

```
    def __init__(self):
```

```
        super().__init__(SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_TITLE)
```

```
        arcade.set_background_color(arcade.csscolor.CORNFLOWER_BLUE)
```

```
    # Listas que contendran nuestros sprites
```

```
    self.coin_list = None
```

```
    self.wall_list = None
```

```
    self.player_list = None
```

```
    # Variable del sprite jugador
```

```
    self.player_sprite = None
```

```
    def setup(self):
```

```
        self.player_list = arcade.SpriteList()
```

```
        self.wall_list = arcade.SpriteList()
```

```
        self.coin_list = arcade.SpriteList()
```

```
    # Crear el jugador
```

```
    image_source = "mario.png"
```

```
    self.player_sprite = arcade.Sprite(image_source, CHARTER_SCALING)
```

```
    self.player_sprite.center_x = 64
```

```
    self.player_sprite.center_y = 93
```

```
    self.player_list.append(self.player_sprite)
```

```
    # Crear el suelo
```

```
    for x in range(0, 1250, 64):
```

```
        wall = arcade.Sprite("ground.png", GROUND_SCALING)
```

```
        wall.center_x = x
```

```
        wall.center_y = 32
```

```

self.wall_list.append(wall)

# Crear los cilindros con una lista.
coordinate_list = [[512, 110], [256, 110], [768, 110]]

for coordinate in coordinate_list:
    wall = arcade.Sprite("cylinder.png", CYLINDER_SCALING)
    wall.position = coordinate
    # Añadiendo a la lista
    self.wall_list.append(wall)

```

```

# Motor de física
self.physics_engine =
arcade.PhysicsEngineSimple(self.player_sprite, self.wall_list)

```

```

def on_draw(self):
    arcade.start_render()
    self.player_list.draw()
    self.wall_list.draw()

```

Declarar nuestro motor de física, que es el que nos va ayudar a detectar las colisiones.

```

def on_key_press(self, key, modifiers):
    if key == arcade.key.UP:
        self.player_sprite.change_y = PLAYER_MOVEMENT_SPEED
    if key == arcade.key.DOWN:
        self.player_sprite.change_y = -PLAYER_MOVEMENT_SPEED
    if key == arcade.key.LEFT:
        self.player_sprite.change_x = -PLAYER_MOVEMENT_SPEED
    if key == arcade.key.RIGHT:
        self.player_sprite.change_x = PLAYER_MOVEMENT_SPEED

```

```

def on_key_release(self, key, modifiers):
    if key == arcade.key.UP:
        self.player_sprite.change_y = 0
    if key == arcade.key.DOWN:
        self.player_sprite.change_y = 0
    if key == arcade.key.LEFT:
        self.player_sprite.change_x = 0
    if key == arcade.key.RIGHT:
        self.player_sprite.change_x = 0

```

Cuando presionamos la tecla:
Con esta función vamos a detectar todas las teclas que presionemos, como parámetros le pasamos la tecla y el modificador.
Según la tecla que presionemos, arriba, abajo, derecha o izquierda modificaremos su coordenada.

```

def on_update(self, delta_time):
    self.physics_engine.update()

```

Cuando soltamos la tecla:
Hace prácticamente que la función anterior pero le pasamos valores de 0 para que el jugador no se mueva.

```

def main():
    window = Mygame()
    window.setup()
    arcade.run()

```

Declaramos una función que es la que va a ir actualizando todo el juego.

```

if __name__ == "__main__":

```

main()



Lo único que no tenemos es gravedad, pero si que detecta las colisiones.

5.- Física (Gravedad)

Vamos a ver las modificaciones:

```
import arcade
```

```
# Constantes
```

```
SCREEN_WIDTH = 1000
```

```
SCREEN_HEIGHT = 500
```

```
SCREEN_TITLE = "Mario Demo"
```

```
# Constantes para escalar los sprites
```

```
CHARTER_SCALING = 0.17
```

```
GROUND_SCALING = 0.20
```

```
CYLINDER_SCALING = 0.20
```

```
# Velocidad del jugador
```

```
PLAYER_MOVEMENT_SPEED = 5
```

```
GRAVITY = 1
```

```
PLAYER_JUMP_SPEED = 20
```

Definimos dos constantes GRAVITY para guardar la gravedad, y PLAYER_JUMP_SPEED la fuerza del salto.

```
class Mygame(arcade.Window):
```

```
    def __init__(self):
```

```
        super().__init__(SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_TITLE)
```

```
        arcade.set_background_color(arcade.csscolor.CORNFLOWER_BLUE)
```

```
    # Listas que contendran nuestros sprites
```

```
    self.coin_list = None
```

```
    self.wall_list = None
```

```
    self.player_list = None
```

```
    # Variable del sprite jugador
```

```
    self.player_sprite = None
```

```
    def setup(self):
```

```
        self.player_list = arcade.SpriteList()
```

```
        self.wall_list = arcade.SpriteList()
```

```
        self.coin_list = arcade.SpriteList()
```

```
    # Crear el jugador
```

```
    image_source = "mario.png"
```

```
    self.player_sprite = arcade.Sprite(image_source, CHARTER_SCALING)
```

```
    self.player_sprite.center_x = 64
```

```
    self.player_sprite.center_y = 93
```

```
    self.player_list.append(self.player_sprite)
```

```
    # Crear el suelo
```

```
    for x in range(0, 1250, 64):
```

```
        wall = arcade.Sprite("ground.png", GROUND_SCALING)
```

```
wall.center_x = x
wall.center_y = 32
self.wall_list.append(wall)
```

Crear los cilindros con una lista.

```
coordinate_list = [[512, 110], [256, 110], [768, 110]]
```

```
for coordinate in coordinate_list:
    wall = arcade.Sprite("cylinder.png", CYLINDER_SCALING)
    wall.position = coordinate
    # Añadiendo a la lista
    self.wall_list.append(wall)
```

```
# Motor de física
self.physics_engine =
arcade.PhysicsEnginePlatformer(self.player_sprite, self.wall_list,
GRAVITY)
```

```
def on_draw(self):
    arcade.start_render()
    self.player_list.draw()
    self.wall_list.draw()
```

Hemos cambiado PhysicsEngineSimple por PhysicsEnginePlatformer y hemos agregado un parámetro de más GRAVITY = 1.

```
def on_key_press(self, key, modifiers):
    if key == arcade.key.UP:
        if self.physics_engine.can_jump():
            self.player_sprite.change_y = PLAYER_JUMP_SPEED
    elif key == arcade.key.LEFT:
        self.player_sprite.change_x = -PLAYER_MOVEMENT_SPEED
    elif key == arcade.key.RIGHT:
        self.player_sprite.change_x = PLAYER_MOVEMENT_SPEED
```

```
def on_key_release(self, key, modifiers):
    if key == arcade.key.LEFT:
        self.player_sprite.change_x = 0
    elif key == arcade.key.RIGHT:
        self.player_sprite.change_x = 0
```

La flecha hacia abajo ya no hará falta ya que la gravedad se encargará de ello.

Ahora cuando presionemos el botón de arriba y nuestro jugador puede saltar una velocidad PLAYER_JUMP_SPEED = 20.

```
def on_update(self, delta_time):
    self.physics_engine.update()
```

```
def main():
    window = Mygame()
    window.setup()
    arcade.run()
```

Ahora solo necesitamos en soltar el botón, el botón derecho y el botón izquierdo, el resto de botones no hace falta.

```
if __name__ == "__main__":
    main()
```

Ya puedes probar el programa.