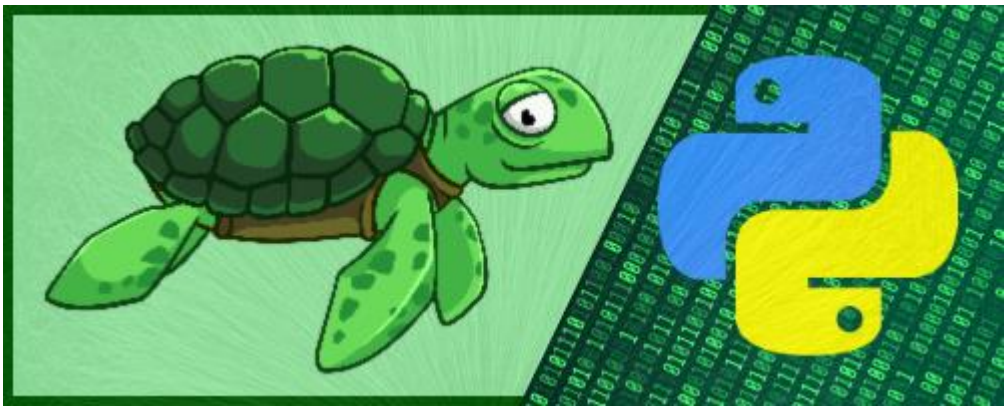


Basándome sobre el tutorial publicado en YouTube en el canal YouDevs, os quiero enseñar paso a paso como se hace un juego del snake.

En el código QR accederás al videotutorial por si necesitas completar la información.

Espero que este tutorial te pueda ser útil.

# PROGRAMAR JUEGOS EN PYTHON

## LIBRERÍA TURTLE

PERE MANEL VERDUGO ZAMORA
Web: www.peremanelv.com
pereverdugo@gamil.com

# Introducción

Para este proyecto lo primero que necesitar es instalarte Python y si es mejor instálate la última versión.

En el siguiente enlace podrás descargarte la última versión:

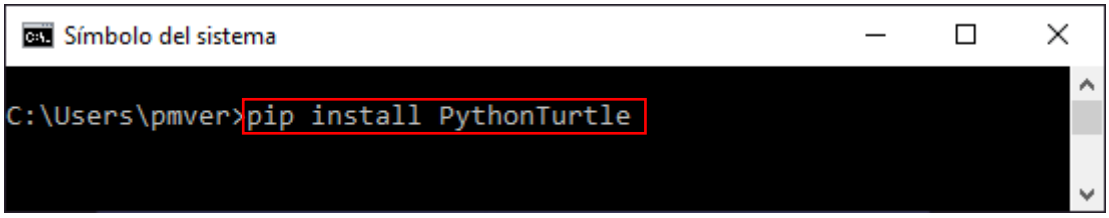
https://www.python.org/downloads/

En otros de mis tutoriales te explico paso a paso la instalación.

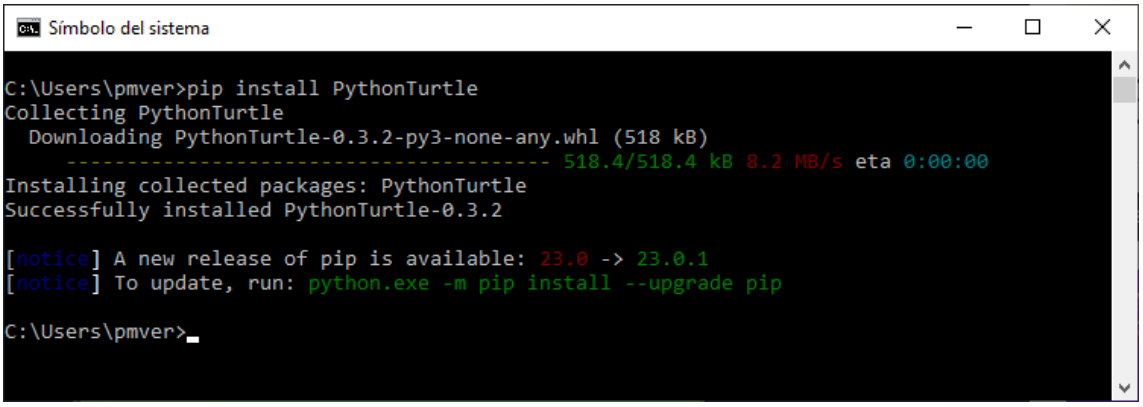
Como editor yo trabajo con Visual Estudio Code te adjunto el enlace por si quisieras trabajar con el:

https://code.visualstudio.com/

Para este proyecto necesitas la librería turtle, desde Cmd de Windows la podrás instalar introduciendo la siguiente línea:




Si la instalación se ha realizado con éxito, obtendrás la siguiente respuesta:




Vamos a empezar con el proyecto.

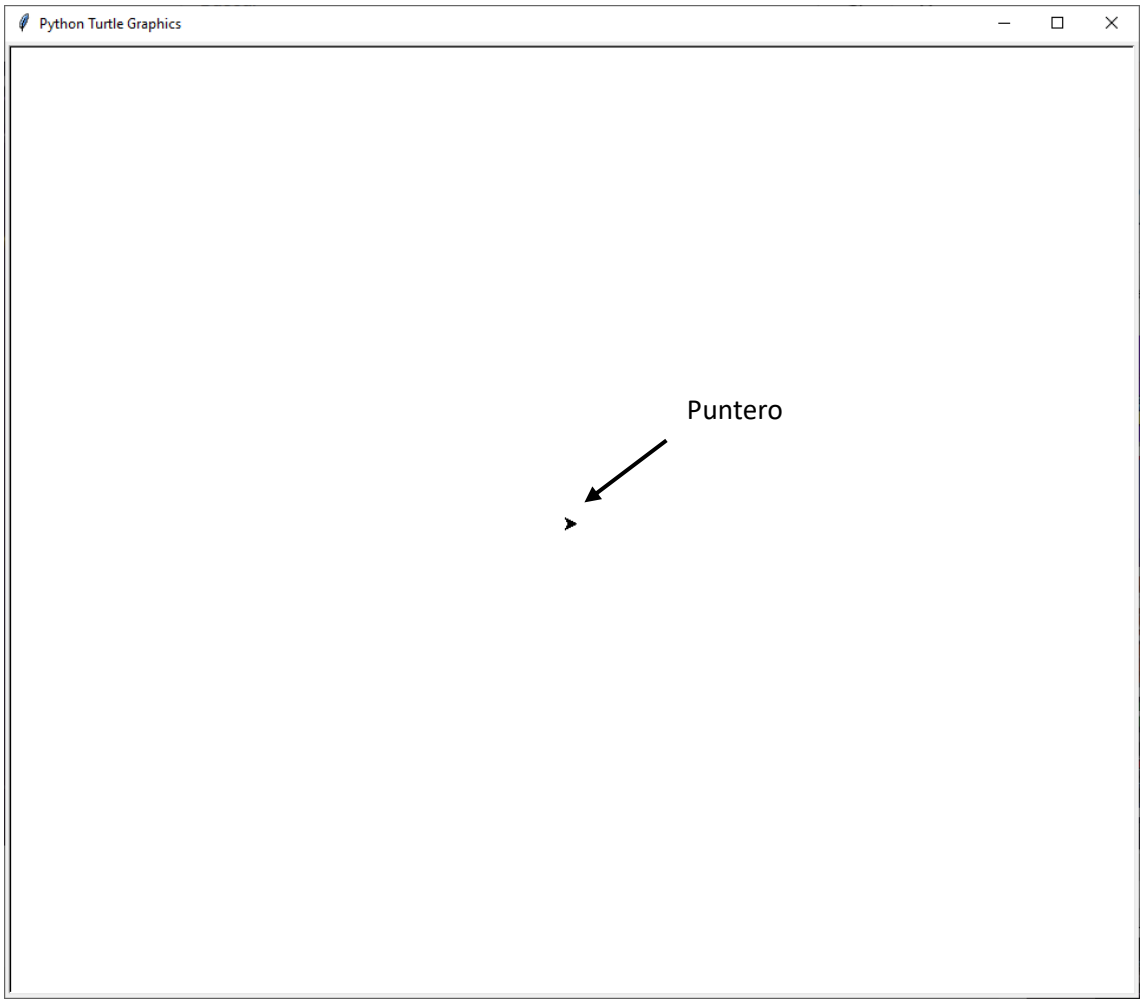
```
1   # Importamos la librería
2   import turtle
3
4   puntero = turtle.Turtle()
```

Se muestra una ventana, pero rápidamente se cierra, casi no da tiempo a verla.

En la línea 4 creamos un objeto llamado puntero perteneciente a la librería turtle.

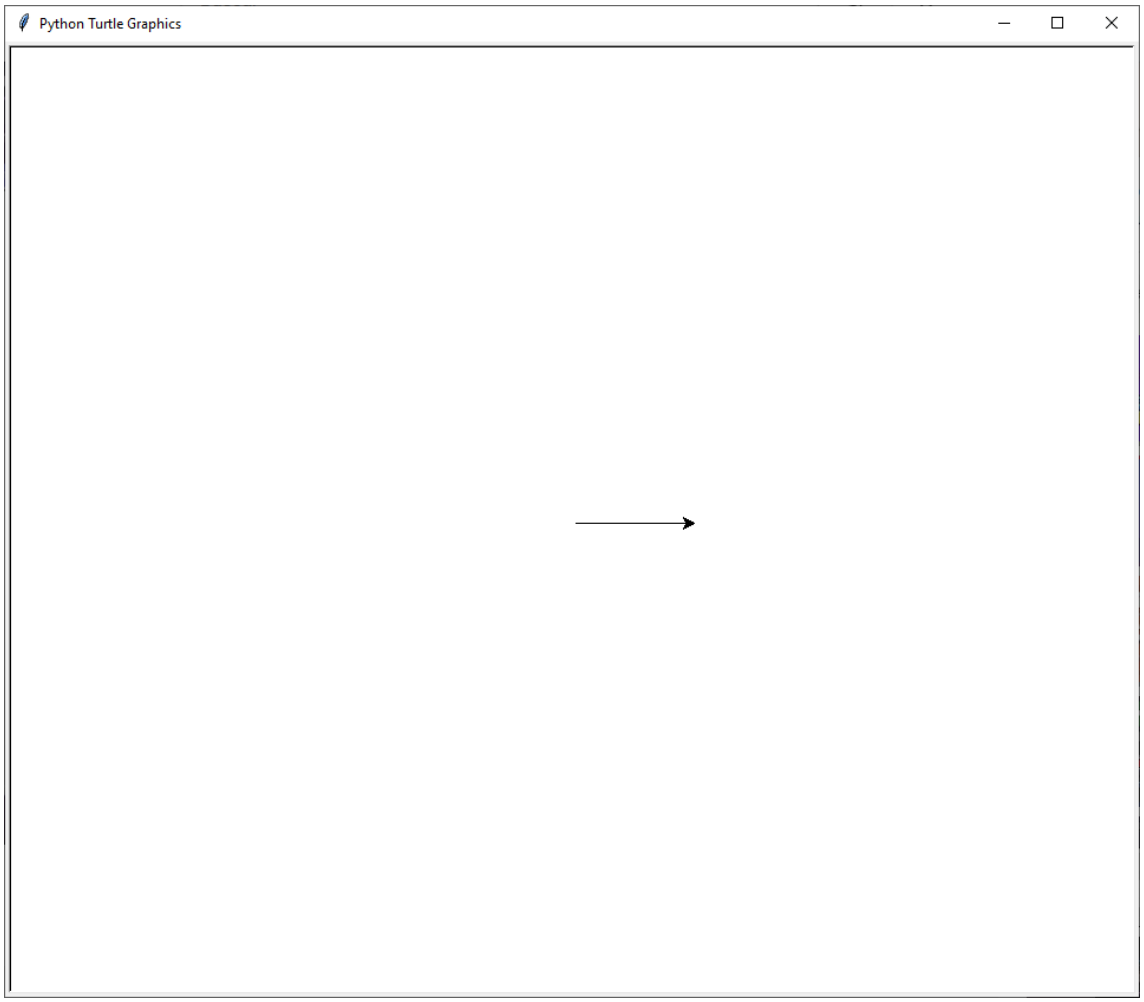
```
1   # Importamos la librería
2   import turtle
3
4   puntero = turtle.Turtle()
5
6   turtle.done()
```

Agregando la línea 6 y ejecutando este será el resultado:




```
1   # Importamos la librería
2   import turtle
3
4   puntero = turtle.Turtle()
5
6   puntero.forward(100)
7
8   turtle.done()
```

Puntero adelante 100 píxeles.

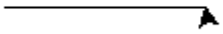



```python
# Importamos la librería
import turtle

puntero = turtle.Turtle()
puntero.speed(1)
puntero.forward(100)

turtle.done()
```

Velocidad.

```python
# Importamos la librería
import turtle

puntero = turtle.Turtle()
puntero.speed(1)
puntero.forward(100)
puntero.left(100)
turtle.done()
```

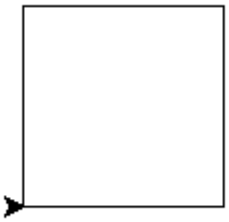
Girar a la izquierda 100



El puntero después de realizar el avance de 10 gira hacia la izquierda 100 grados.

Vamos a ver un ejemplo:

```
1   # Importamos la librería
2   import turtle
3
4   puntero = turtle.Turtle()
5   for i in range(4):
6       puntero.speed(1)
7       puntero.forward(100)
8       puntero.left(90)
9   turtle.done()
```



Vamos a empezar con el juego.

```
1   # Importamos la librería
2   import turtle
3   # Ventana
4   wn = turtle.Screen()
5   # Título de la ventana
6   wn.title("Juego Snake")
7   # Medidas de la ventana
8   wn.setup(width=600, height=600)
9   # Color de la ventana
10  wn.bgcolor('green')
11  turtle.done()
```

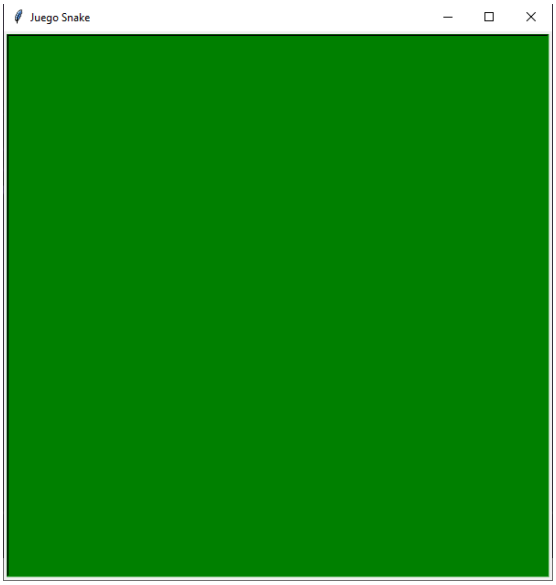
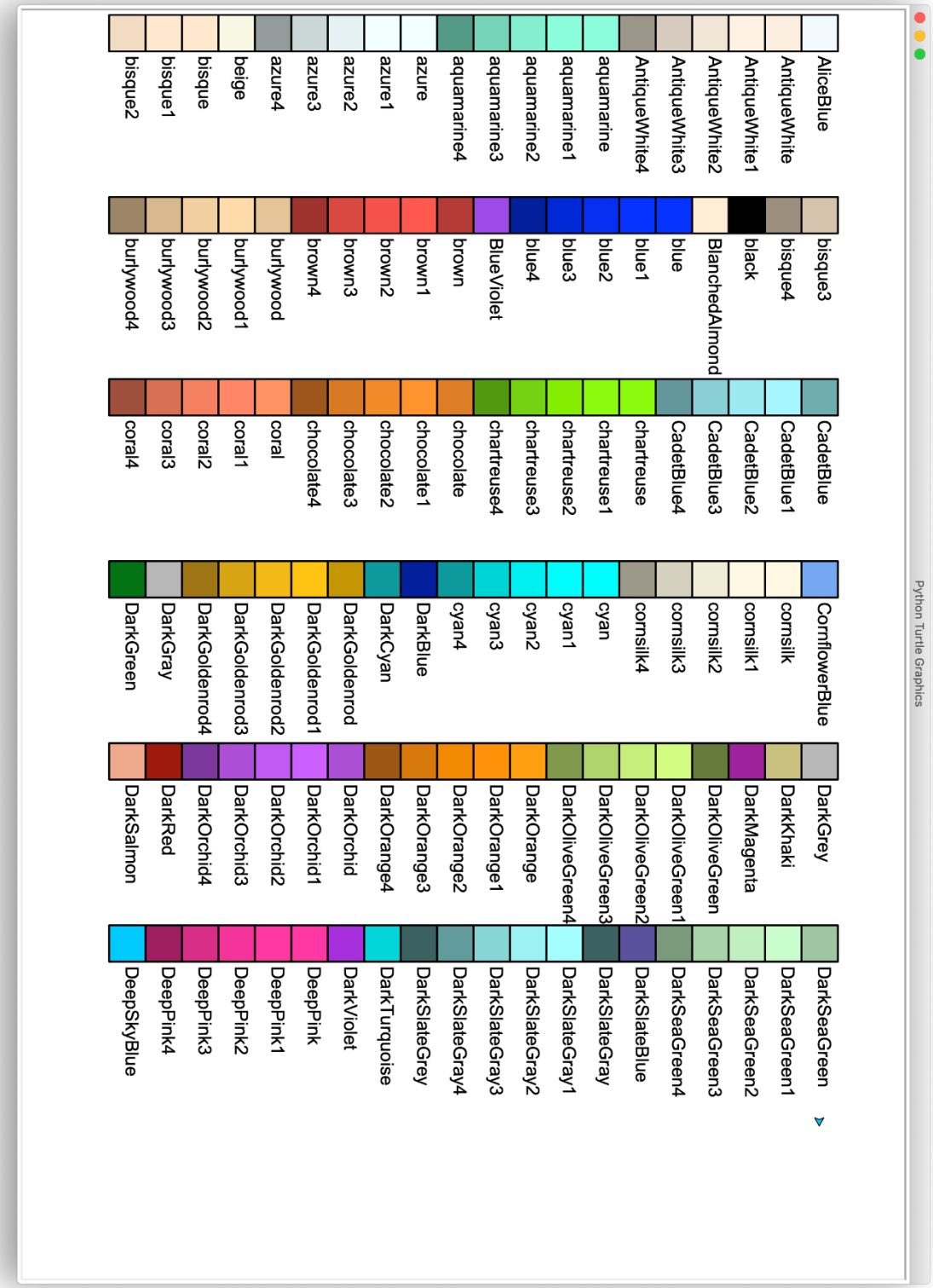
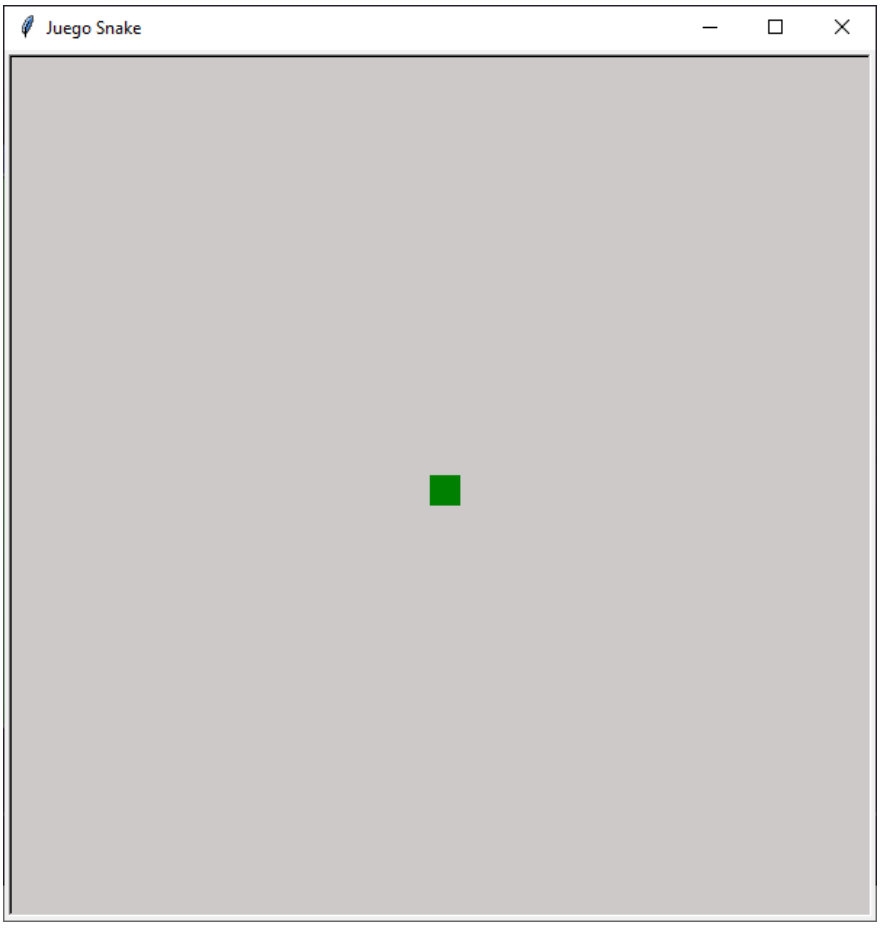


Tabla de colores:

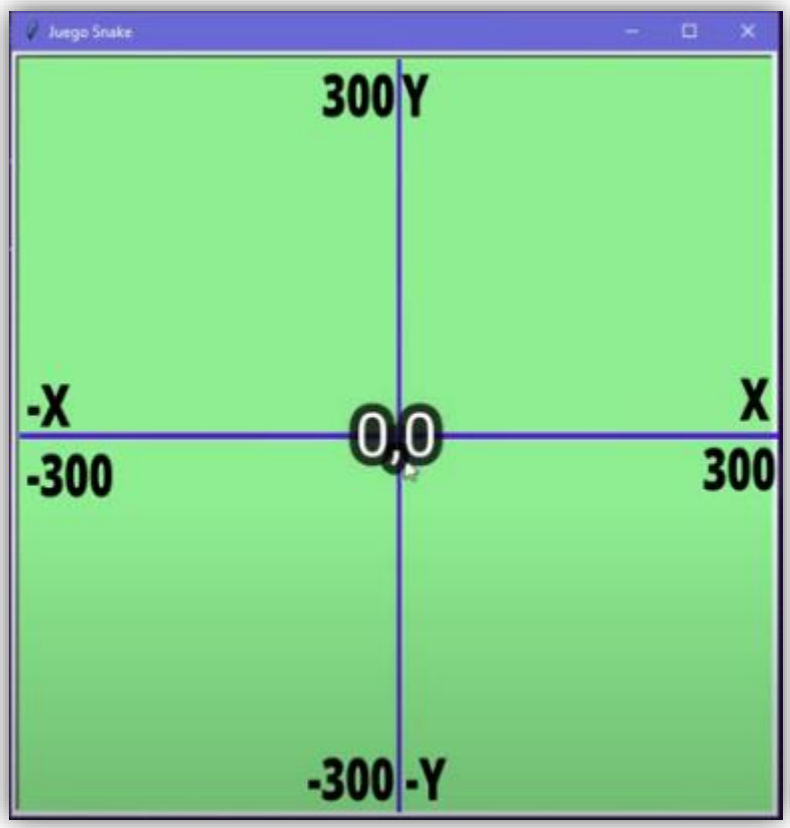


```
1   # Importamos la librería
2   import turtle
3   # Ventana
4   wn = turtle.Screen()
5   # Título de la ventana
6   wn.title("Juego Snake")
7   # Medidas de la ventana
8   wn.setup(width=600, height=600)
9   # Color de la ventana
10  wn.bgcolor('snow3')
11
12  head = turtle.Turtle()
13  head.speed(0)
14  head.shape('square')
15  head.color('green')
16  head.goto(0,0)
17  head.direction = "up"
18
19  turtle.done()
```

Creamos un objeto de tipo Turle()
A una velocidad de 0
Que sea de tipo cuadrado.
De color verde.
Coordenada (0,0) es justo al centro.
La dirección mirando hacia arriba.



Así son las coordenadas:




```
1   # Importamos la librería
2   import turtle
3
4   head = turtle.Turtle()
5   head.penup()
6   head.goto(-300, 0)
7   for i in range(1, 30):
8       head.pendown()
9       head.forward(10)
10      head.penup()
11      head.forward(10)
12
13  turtle.done()
```

Se crea un objeto de tipo Turtle().
Sube el lápiz (no pinta).
Posiciónate a las coordenadas -300, 0.
Un bucle que se repite 30 veces.
Baja el lápiz (pinta)
Hacia adelante 10 pixeles.
Sube el lápiz (No pinta).
Hacia adelante 10 pixeles.

**Adjunto todo el código:**

```
import turtle
import time
import random
```

Importamos las librerías turtle, time y random.

```
delay = 0.1
body_segments = []
score = 0
high_score= 0
```

Definimos las siguientes variable:

delay tiempo de espera, score los puntos y high_cores lo puntos totales.

```
wn = turtle.Screen()
wn.title("Juego Snake")
wn.setup(width=600, height= 600)
wn.bgcolor("light green")
```

Para crear la ventana, con el título “Juego Snake”, dimensiones de 600 x 600. Color de la ventana “light green”.

```
head = turtle.Turtle()
head.speed(0)
head.shape('square')
head.color('green')
head.goto(0,0)
head.penup()
head.direction = "stop"
```

Creamos un objeto llamado head (Cabeza del gusano) que será de la clase Turlle, será un cuadrado de color verde, se posicionará en el centro de la ventana. Cuando se mueva no pintará y dirección es igual a “stop”.

```
# food config
food =turtle.Turtle()
food.speed(0)
food.shape("circle")
food.penup()
food.goto(0,100)
food.direction = "stop"
```

Creamos un objeto llamado (alimento) de tipo Turtle(), velocidad igual a 0 en forma de circulo, al moverse no pintará, lo situaremos en las coordenadas 0,100 y su dirección será igual a “stop”.

```
# Text Score
text = turtle.Turtle()
text.speed(0)
text.color('white')
text.penup()
text.hideturtle()
text.goto(0, 260)
text.write(f'Score 0          High Score: 0', align="center",
font=("Impact", 24))
```

Definimos un objeto llamado Text de tipo Turtle, con una velocidad a 0, color blanco, no pinta al moverse, estará oculta, se posicionará en las coordenadas 0, 260 y escribirá el siguiente texto.

```
def mov():
    if head.direction == "up":
        y = head.ycor()
        head.sety(y + 10)
```

Función mov():
Si head apunta hacia arriba la variable ‘y’ asume el valor de su coordenada y, A la coordenada ‘y’ de head a y se le incrementa 10.

```
    if head.direction == "down":
        y = head.ycor()
        head.sety(y - 10)
    if head.direction == "right":
        y= head.xcor()
        head.setx(y + 10)
    if head.direction == "left":
        y = head.xcor()
        head.setx(y - 10)
```

Lo mismo hacemos cuando apunta hacia abajo pero en este caso a la coordenada y de head le restamos 10.

Lo mismo hacemos cuando apunta a la derecha en ese caso a la coordenada x se le incrementa 10

Lo mismo hacemos cuando apunta hacia la izquierda en ese caso la coordenada x se le resta 10

```
def dirUp():
    head.direction = "up"

def dirDown():
    head.direction = "down"

def dirRight():
    head.direction = "right"

def dirLeft():
    head.direction = "left"
```

Estas 4 funciones hacen que la dirección de head apunte hacia arriba, abajo, derecha o izquierda, cuando se les llamen.

```
# Conectar teclado:
wn.listen()
wn.onkeypress(dirUp, "Up")
wn.onkeypress(dirDown, "Down")
wn.onkeypress(dirRight, "Right")
wn.onkeypress(dirLeft, "Left")
```

Que esté en alerta a la tecla que presionamos.

Según la flechas de dirección del teclado que presionemos, estas llamarán a su respectiva función, según la tecla que se detalla.

La tecla tiene que empezar con la primera letra en mayúsculas.

```
while True:
```

Bucle principal

```
    wn.update()
```

Actualiza la ventana

```
    # Colisiones con la ventana
    if head.xcor() > 280 or head.xcor() < -280 or head.ycor() > 280 or
head.ycor() < -280:
        time.sleep(1)
        head.goto(0,0)
        head.direction = "stop"

        # Esconder segmentos:
        for segment in body_segments:
            segment.goto(1000, 1000)

        body_segments.clear()
        score = 0
        text.clear()
        text.write(f'Score {score}          High Score: {high_score}',
align="center", font=("Impact", 24))
```

Si head toca los bordes de la ventana, que se detenga un segundo, se sitúe al centro de la ventana, y como dirección sea igual a “stop”.
Envía los segmentos de la cola fuera de la ventana y a continuación los elimina.
La variable score (puntos) pasa a valer 0.
Borra el texto de la ventana para actualizarlos con los nuevos datos.

```
    if head.distance(food) < 20:
        x = random.randint(-280, 280)
        y = random.randint(-280, 280)
        food.goto(x,y)

        new_segment = turtle.Turtle()
        new_segment.speed(0)
        new_segment.shape('square')
        new_segment.color('yellow')
        new_segment.penup()
        body_segments.append(new_segment)

        score += 1
        if score > high_score:
            high_score = score

        text.clear()
        text.write(f'Score {score}
align="center", font=("Impact", 24))


    totalSeg = len(body_segments)

    for i in range(totalSeg-1, 0, -1):
        x = body_segments[i-1].xcor()
        y = body_segments[i-1].ycor()
        body_segments[i].goto(x,y)

    if totalSeg > 0:
        x = head.xcor()
        y = head.ycor()
        body_segments[0].goto(x,y)
```

mov() ← Llamamos a la función mov()

```
    # Colisiones con el propio cuerpo
    for segment in body_segments:
        if segment.distance(head) < 10:
            time.sleep(1)
            head.goto(0,0)
            head.direction = "stop"

            # Esconder segmentos
            for segment in body_segments:
                segment.goto(1000, 1000)
```

Si la distancia de head (cabeza) con respecto a food (alimento) es menor a 20 el alimento se tiene que desplazar a una coordenada aleatoria, para que head lo pueda ir siguiendo.

Creamos un objeto llamado new_segment de tipo Turtle(), con velocidad 0, de forma cuadrada, color amarillo, que no pinte y por último lo agregamos a la lista body_segments, la cola crece.

La variable score (puntos) se incrementa en 1. Si score es mayor a high_score la variable high_score asuma el valor de score una forma de controla la puntuación total.

```
High Score: {high_score}',
```

Borra el texto de la ventana y lo vuelve a mostrar actualizado.
La variable totalSeg asume el número de segmentos que tiene, partiendo de este valor realizamos un ciclo for donde i asume el número de segmentos menos uno hasta 0 con un incremento de -1.
La variable x asume la coordenada x de cada segmento, lo mismo la y que asume la coordenada y de todos los segmentos.
Todos los segmentos se desplazarán en las coordenadas x, y.

En un ciclo for la variable segment recorrerá por todos los elementos de body_segments. Si el segment tiene una distancia menor a 10 pixeles, hay una espera de 1 segundo, se coloca al centro de la ventana y como dirección igual a stop, para que no se mueva.

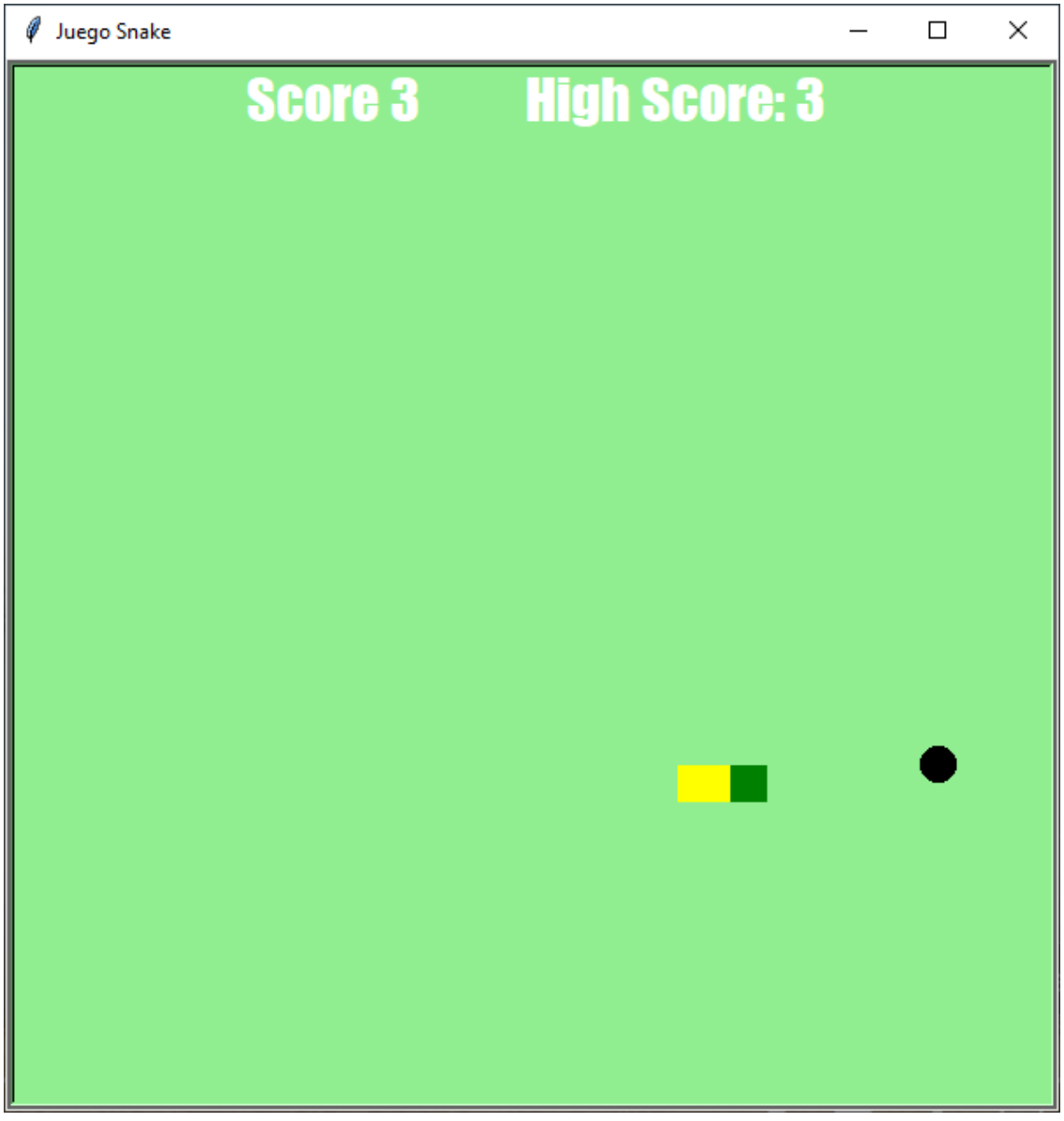
Los segmentos se van fuera de la ventana para eliminarlos.

```
            body_segments.clear()
            score = 0
            text.write(f'Score {score}          High Score:
{high_score}', align="center", font=("Impact", 24))

    time.sleep(delay)


turtle.done()
```

La variable score pasa a vale 0 y de nuevo se actualizan los datos en la pantalla.

Una espera de 0.1 segundos.

Y por último volvemos a iniciar el bucle.




Para más información sobre los comandos Turtle te adjunto el siguiente enlace:

https://docs.python.org/3/library/turtle.html#turtle.update