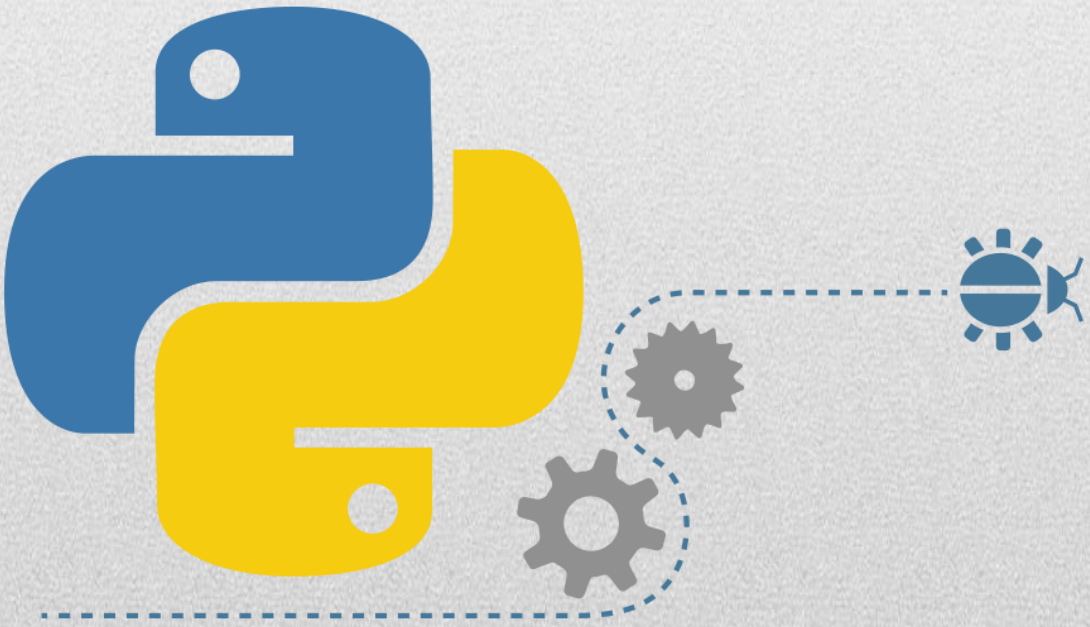
Python Proyecto



Miguel Torres Medina y Jorge León Sánchez 2º Bach

Índice

1-Introducción

En este trabajo usando turtle graphics, una herramienta del lenguaje de programación Python, hemos hecho un código para representar diferentes funciones gráficamente.

```
In [8]: import turtle
        import random
        import time

        turtle.bgcolor("#000000")
        turtle.pensize(3)
        turtle.speed(150)

        time.sleep(4)
        def cuadrado():  # definimos una función llamaa cuadrado
            for i in range(0,4):
                turtle.forward(30)
                turtle.left(90)

        def estrella():   # definimos una función llamada estrella
            for i in range(0,24):
                color= random.choice(["#FE2E2E","#BFFF00","#0080FF","#FF8000","#81DAF5","#F4FA58"])
                turtle.color(color)
                cuadrado()     # hacemos una llamada a esta función
                turtle.left(15)
```

2-Código y su explicación

En esta parte comenzamos importando diferentes variables para ser utilizadas posteriormente.

Después seleccionamos un color de fondo, un grosor de línea y una velocidad de dibujo para representar las funciones que vamos a escribir. Y ordenamos que antes de empezar a dibujar espere durante 4 segundos.

En lo siguiente definimos un cuadrado y su tamaño.

Luego definimos una estrella diciendo que se forma por cuadrados que giran 15 grados entre sí cuyo color es aleatorio entre 6 colores.

```
def rosa():
    for i in range(1,13):
        turtle.forward(80)
        turtle.left(30)
        estrella()

rosa()

turtle.clearscreen()


import turtle
import random

myPen = turtle.Turtle()
myPen.ht()
myPen.speed(250)
color = random.choice (["#FF0000", "#FFFF00"])
myPen.pencolor(color)
turtle.bgcolor ("#000000")
points = [[-175,-125],[0,175],[175,-125]]
```

Ahora definimos el dibujo de la rosa, que está formada por las estrellas definidas anteriormente.

Después indicamos que devuelva la pantalla al inicio como si no hubiera código antes.

Entonces volvemos a importar variables, y definimos la velocidad de dibujo y que el color de la línea será uno entre dos posibles, también ponemos un color de fondo.

Y finalmente unos puntos que conformaran nuestro triángulo inicial.

```
def getMid(p1,p2):
    return ( (p1[0]+p2[0]) / 2, (p1[1] + p2[1]) / 2)

def triangle(points,depth):

    myPen.up()
    myPen.goto(points[0][0],points[0][1])
    myPen.down()
    myPen.goto(points[1][0],points[1][1])
    myPen.goto(points[2][0],points[2][1])
    myPen.goto(points[0][0],points[0][1])

    if depth>0:
        triangle([points[0],
                        getMid(points[0], points[1]),
                        getMid(points[0], points[2])],
                  depth-1)
        triangle([points[1],
                        getMid(points[0], points[1]),
                        getMid(points[1], points[2])],
                  depth-1)
        triangle([points[2],
                         getMid(points[2], points[1]),
                         getMid(points[0], points[2])],
                  depth-1)


triangle(points,6)
turtle.exitonclick()
```

Ahora definimos como calcular el punto medio de cada lado del triángulo.

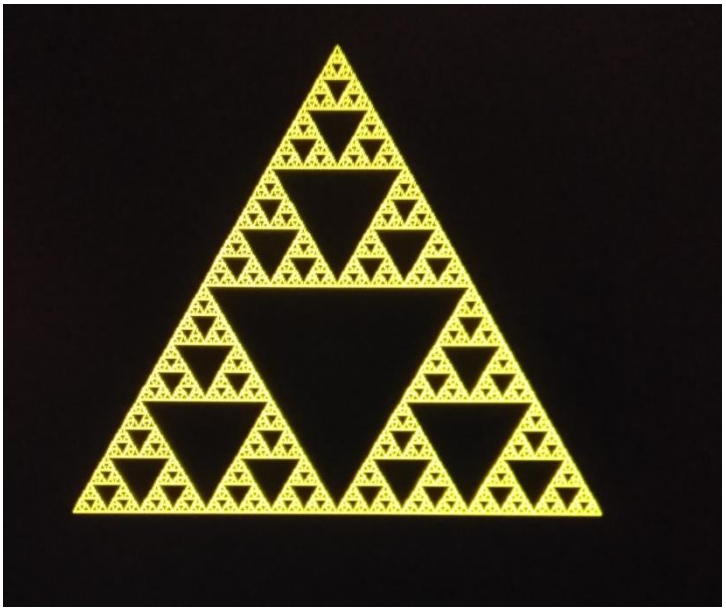
Entonces decimos que el punto inicial se mueva sin que se vea y luego decimos que dibuje el triángulo inicial.

Después decimos que vaya calculando el punto medio de cada lado dividiendo el triángulo en otros triángulos dentro de este.

La última orden es el número de veces que lo hace.

Finalmente decimos que al hacer click se cierre.

3-Fotografías demostración.



4-Conclusión.

No ha sido un trabajo de especial complejidad, pero entre que algunos códigos más complicados no los cargaba el ordenador y el tiempo del que disponíamos no hemos podido hacer un trabajo muy complejo.