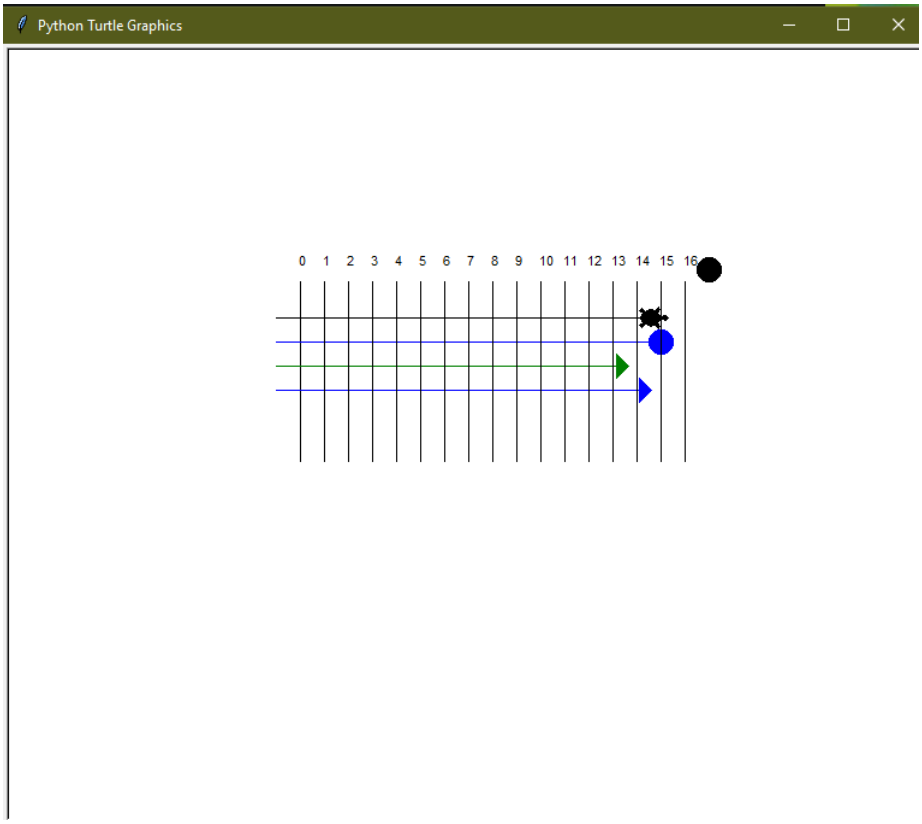
Proyecto 1 Módulo 4 *Tortugas Simultáneas*

En este proyecto se debe recrear y simular una carrera de tortugas. Estás deben ser simultáneas gracias al módulo.



Este se debe de recrear con ayuda del lenguaje de programación Python y la librería Turtle, el módulo *threading*, *queue* y el módulo propio proporcionado en clase. Para descargar el módulo propio hecho en clase, seguir el siguiente link

Especificaciones Generales

Formato de entrega, se debe entregar 2 archivos:

1. **Código del programa:** Guardar el código con el nombre *proyecto-carrera.py*
2. **Reporte del código:** Este debe conllevar los siguientes puntos:
    1. Nombre del alumno
    2. Fecha
    3. Nombre del proyecto
    4. Problemática que se soluciona
    5. Código realizado por el alumno

```
"""
@author: nombre de quien hace el programa
@date: Fecha en la que se realiza el programa
@description: Descripcion del programa
"""
Completar el código de la actividad que se realiza
```

6. Captura de pantalla del código ejecutandose
7. Captura de pantalla del validador (En este proyecto no hay validador, este punto se omite)

Especificaciones del código

A continuación se describe las características que debe contener el código y el dibujo.

1. Módulo creado en clase **modulo2.py**: Éste código es el msotrado en clase, de el se deben usar las funciones:
    1. generate_screen()
    2. generate_Tortugas()
    3. penup()
    4. pendown()
    5. goto()
    6. write()
    7. right()
    8. forward()
    9. backward()
    10. left()
2. Código respecto al archivo de acción **proyecto-carrera.py**, este código debe estar dividido en 3 secciones:
    1. **Creación de pantalla, tortugas y queue de reproducción**:
        1. En ésta sección se deben de definir 3 - 4 tortugas competidoras, éstan pueden tener el nombre que deseen, 1 tortuga trazadora de la pista y una pantalla.

```
##Pantalla
screen = generate_screen("")

##Tortugas Competidoras
frank = generate_Tortugas("black","turtle",1,0)
maturin = generate_Tortugas("blue","circle",1,0)
rafael = generate_Tortugas("green","arrow",1,0)
donatello = generate_Tortugas("blue","arrow",1,0)

###Tortuga trazadora
traza = generate_Tortugas("black","circle",1,0)
```

    2. **Funciones de acción:**
        1. **Función trazadora:** Ésta función es exclusiva de la tortuga trazadora

```
def trazadora():
    penup(traza, graphics)
    goto(traza, graphics, -140,140)
    for i in range(17):
        write(traza, graphics, str(i))
        right(traza, 90,graphics)
        forward(traza, 10, graphics)
        pendown(traza, graphics)
        forward(traza, 150, graphics)
        penup(traza, graphics)
        backward(traza, graphics,160)
        left(traza, 90, graphics)
        forward(traza,20, graphics)
```

2. **Función competidor:** Ésta función es para poder mover cada uno de los competidores

```
def competidor(tortuga, graphics,posx, posy):
    penup(tortuga, graphics)
    goto(tortuga, graphics, posx,posy)
    pendown(tortuga, graphics)
    for i in range(100):
        forward(tortuga, randint(1,5), graphics)
```

3. Creación y acción de hilos de ejecución: En esta sección se deben de crear los hilos de ejecución correspondientes a cada tortuga

```
###Hilos de Ejecución
threading.Thread(target=trazadora).start()
threading.Thread(target=competidor, args=(frank, graphics,-160,
100)).start()
threading.Thread(target=competidor, args=(maturin, graphics,-160,
80)).start()
threading.Thread(target=competidor, args=(rafael, graphics,-160,
60)).start()
threading.Thread(target=competidor, args=(donatello, graphics,-160,
40)).start()
process_queue(graphics, screen)

screen.exitonclick()
```

4. **EXTRAS:** Se recomienda importar los siguientes módulos.

```
###
from random import *
import queue
import threading
from turtle import pendown, penup
```

Proyecto 2 Módulo 4 *Tortugas Simultáneas*

Esta sección es meramente opcional, se recomienda usar el archivo *modulo2.py*, para poder crear un dibujo de creación libre, este puede ser el que sea, mándalas, figuras de fantasía, etc. Sin embargo se deve de considerar los siguientes puntos:

1. **Uso del modulo2:** Se debe usar el modulo 2 creado en clase, se adjunta el link
2. **Creación de por lo menos 3 hilos:** Para este proyecto se deben de usar por lo menos 3 hilos de ejecución para que sean tortugas simultáneas
3. **Creatividad:** Se calificará con puntos extra la creatividad.

Ejemplo de Código, en el se usan 10 hilos:

```
import queue
import threading
###
#from modulo2 import forward,right, left, figuree, process_queue,
generate_screen,generate_Tortugas
from modulo2 import *
##torttuga 1
frank = generate_Tortugas("green", "turtle", 1, 0)
##tortuga 2
maturin = generate_Tortugas("yellow", "arrow", 1, 0)

##torttuga 3
frank0 = generate_Tortugas("blue", "turtle", 1, 0)
##tortuga 4
maturin0 = generate_Tortugas("red", "arrow", 1, 0)

##torttuga 5
frank1 = generate_Tortugas("blue violet", "turtle", 1, 0)
##tortuga 6
maturin1 = generate_Tortugas("linen", "arrow", 1, 0)

##torttuga 7
frank2 = generate_Tortugas("deep pink", "turtle", 1, 0)
##tortuga 8
maturin2 = generate_Tortugas("dark cyan", "arrow", 1, 0)
##torttuga 9
frank3 = generate_Tortugas("medium slate blue", "turtle", 1, 0)
##tortuga 10
maturin3 = generate_Tortugas("white", "arrow", 1, 0)

###Queue de Reproduccion
graphics = queue.Queue(1)  # size = number of hardware threads you have - 1
########################################################
##MAturin Circulo interno
def func1():
    for i in range(30):
        figure(maturin, graphics,3,70)
        right(maturin, 65, graphics)
##Frank circulo externo
def func2():
    for i in range(60):
        figure(frank, graphics,8,100)
```

```
        right(frank, 91,graphics)
#####################################################
##MAturin Circulo interno
def func3():
    penup(maturin0,graphics)
    goto(maturin0,graphics,-300, 180)
    pendown(maturin0,graphics)
    for i in range(30):
        figure(maturin0, graphics,3,70)
        right(maturin0, 65, graphics)
##Frank circulo externo
def func4():
    penup(frank0,graphics)
    goto(frank0,graphics,-300, 180)
    pendown(frank0,graphics)
    for i in range(60):
        figure(frank0, graphics,8,100)
        right(frank0, 91,graphics)

#####################################################
##MAturin Circulo interno
def func5():
    penup(maturin1,graphics)
    goto(maturin1,graphics,300, 180)
    pendown(maturin1,graphics)
    for i in range(30):
        figure(maturin1, graphics,3,70)
        right(maturin1, 65, graphics)
##Frank circulo externo
def func6():
    penup(frank1,graphics)
    goto(frank1,graphics,300, 180)
    pendown(frank1,graphics)
    for i in range(60):
        figure(frank1, graphics,8,100)
        right(frank1, 91,graphics)

#####################################################
##MAturin Circulo interno
def func7():
    penup(maturin2,graphics)
    goto(maturin2,graphics,300, -180)
    pendown(maturin2,graphics)
    for i in range(30):
        figure(maturin2, graphics,3,70)
        right(maturin2, 65, graphics)
##Frank circulo externo
def func8():
    penup(frank2,graphics)
    goto(frank2,graphics,300, -180)
    pendown(frank2,graphics)
    for i in range(60):
        figure(frank2, graphics,8,100)
        right(frank2, 91,graphics)
#####################################################
##MAturin Circulo interno
def func9():
    penup(maturin3,graphics)
```

```
    goto(maturin3,graphics,-300, -180)
    pendown(maturin3,graphics)
    for i in range(30):
        figure(maturin3, graphics,3,70)
        right(maturin3, 65, graphics)
##Frank circulo externo
def func10():
    penup(frank3,graphics)
    goto(frank3,graphics,-300, -180)
    pendown(frank3,graphics)
    for i in range(60):
        figure(frank3, graphics,8,100)
        right(frank3, 91,graphics)
###Hilos de Ejecución
threading.Thread(target=func1).start()
threading.Thread(target=func2).start()
###
threading.Thread(target=func3).start()
threading.Thread(target=func4).start()
###
threading.Thread(target=func5).start()
threading.Thread(target=func6).start()
###
threading.Thread(target=func7).start()
threading.Thread(target=func8).start()
###
threading.Thread(target=func9).start()
threading.Thread(target=func10).start()

screen = generate_screen("black")

process_queue(graphics, screen)

screen.exitonclick()
```

Ejemplo de salida: