Módulo 3 (Módulo Tortugas)

En este módulo se revisará a fondo el uso de la librería tortugas, está viene incluida en el paquete oficial de python y no se debe realizar ninguna instalación adicional.

Gráficas Turtle es una forma muy habitual de introducción a la programación para niñas y niños. Era parte original del lenguaje de programación Logo, desarrollado por Wally Feurzeig, Seymour Papert y Cynthia Solomon en 1967. Esto quiere decir que el módulo tortugas es una forma sencilla dee aprender a programar y darle instrucciones a un prototipo de *robot*

En caso de necesitar instalat algo adicional, se recomienda hacer uso de la terminal e instalar los paquete marcados

Paquete Base Turtle

```
pip install turtle
```

Paquete Base Tkinter

```
pip install tkinter
```

El siguiente es solo en el caso de estar usando google Colab

```
pip install ColabTurtle
```

1.- Comandos Básicos

1.1.-Avanzar y Girar

Para este primer caso, sólo se requiere importar la librería tortugas, esto se puede hacer de 2 formas, a continuación veremos la primera forma.

- Forma 1, importando toda la librería

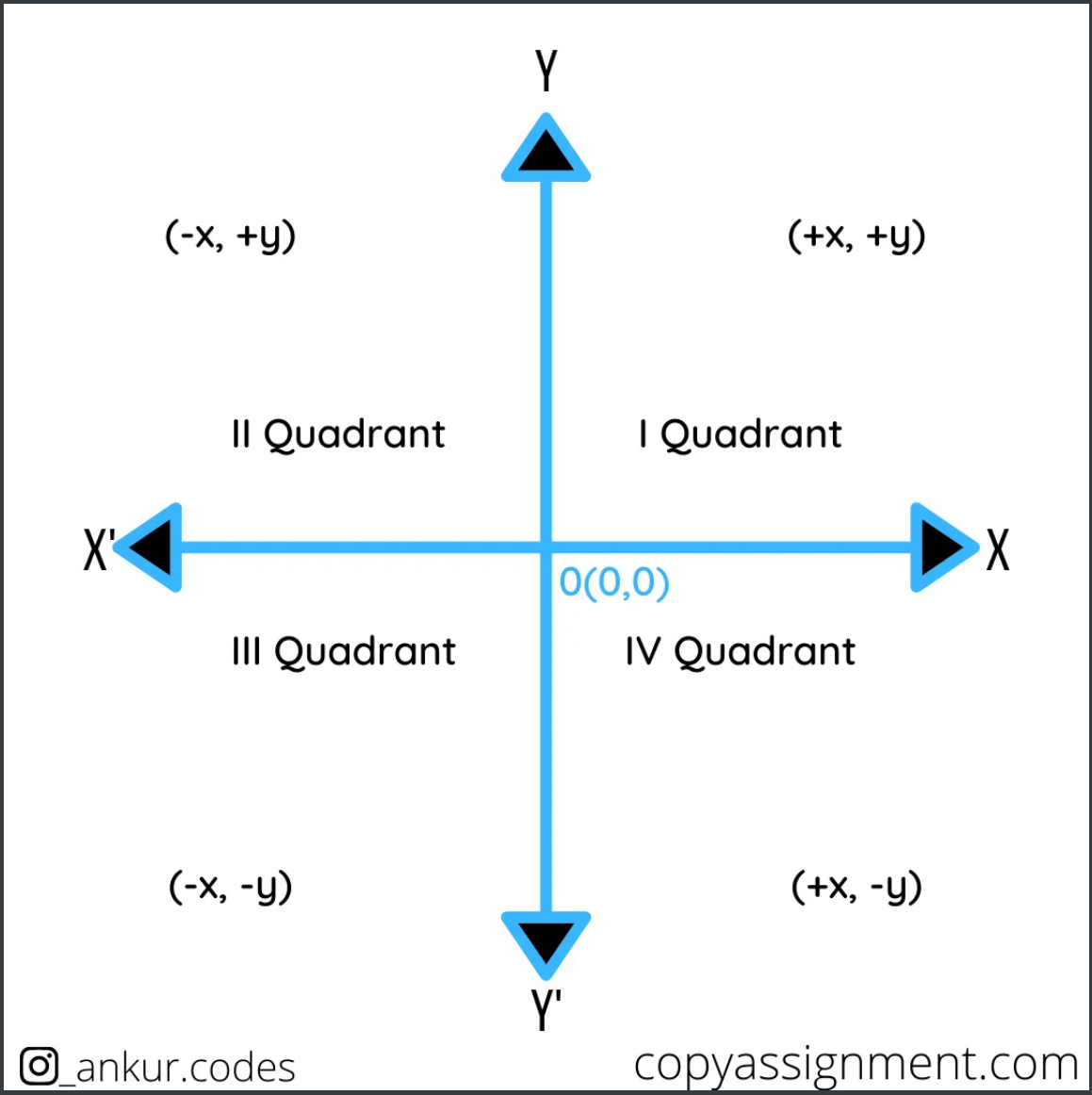
```
from turtle import *
```

- Forma 2, importando sólo lo necesario

```
import turtle
```

En algunos casos será necesario usar una forma en esepcifico para sacar mejor provecho a este módulo

Una vez que tenemos importado el módulo tortugas, lo siguiente que haremos será indicarle la dirección hacía donde esta se dirigirá, la tortuga, esto lo haremos con comandos y giros básicos. A continuación se detalla las coordenadas y orieentación que sigue la tortuga para hacer sus movimientos



Los comandos que se usarán para poder mover la tortuga de forma básica serán:

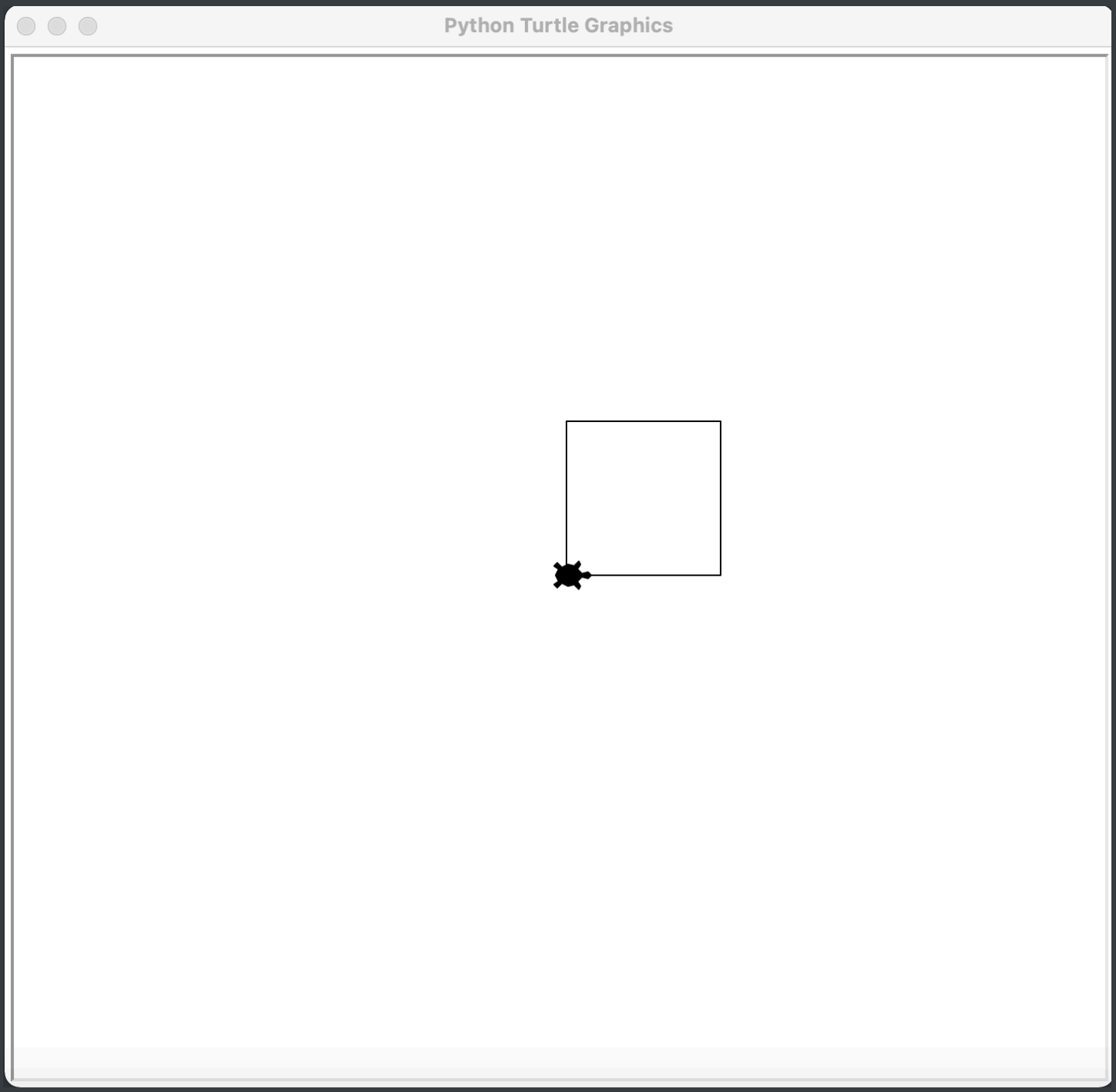
- **forward(numero_de_casillas_avanzar):** Numero de casillas a avanzar, esto nos permite que nuestra tortuga avance una cantidad de espacios determinados
  - Este comando también se puede abreviar a *fd*
- **left(numero_de_casillas_girar):** Numero de casillas a girar, estos giros siempre se dan en torno a 360º, cuando se indica la palabra *left*, se está diciendo que se gire hacia la izquierda
  - Este comando también se puede abreviar a *lt*

- **right(numero_de_casillas_girar):** Numero de casillas a girar, estos giros siempre se dan en torno a 360º, cuando se indica la palabra *right*, se está diciendo que se gire hacia la derecha
    - Este comando también se puede abreviar a *rt*
- **shape(forma_tortuga):** Adicionalmente podemos elegir entre 3 formas de nuestra tortuga
    - **turtle**: Nos permite ver una tortuga moviéndose
    - **circle**: Nos permite ver una círculo moviéndose
    - **arrow**: Nos permite ver una triángulo moviéndose

Código Final

A continación se describe el código final

```
#### Basico sin usar funciones
from turtle import *
shape("turtle")
forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
#input() ##Opcional si se requieree poner una pausa
```



1.2.-Ordenar Tu código

Por lo regular siempre será bueno que el código se divida en pequeñas funciones que hagan una tarea específica, de esta forma, podemos darle mayor congruencia a nuestro código y posteriormente a nuestras ideas

A continuación se describe la estructura básica de una función

```
def nombre_funcion(parametros):
  ##todo lo que haga mi función
  return
```

Partes de mi función:

```
def nombre_funcion
```

- En estas líneas dee código se usa la palabra reservada *def*, la cual sirve para indicarle a python que estamos por definir una función, la cual va acompañada por nuestro nombre de la función *nombre_funcion*, este nombre es definido por nosotros

```
def nombre_funcion(parametros):
```

- En algunos casos necesitaremos agregarle parámetros a nuestras funciones, es deecir que reciban datos, estos datos pueden ser de tipo *int*, *string*, *float* o bien una lista o tupla, estos van dentro de los paréntesis y sólo viven dentro de nuestra función

```
def nombre_funcion(parametros):
  ##todo lo que haga mi función
  return
```

- En estas últimas líneas podemos notar que se dejan 4 espacios entre el inicio de mi función y el inicio del código que pertenece a mi función, de esta forma, podemos decir que eel código alojado dentro de mi función, sólo y únicamente vive dentro de mi función. A estos 4 espacios se les conoce como **identación**

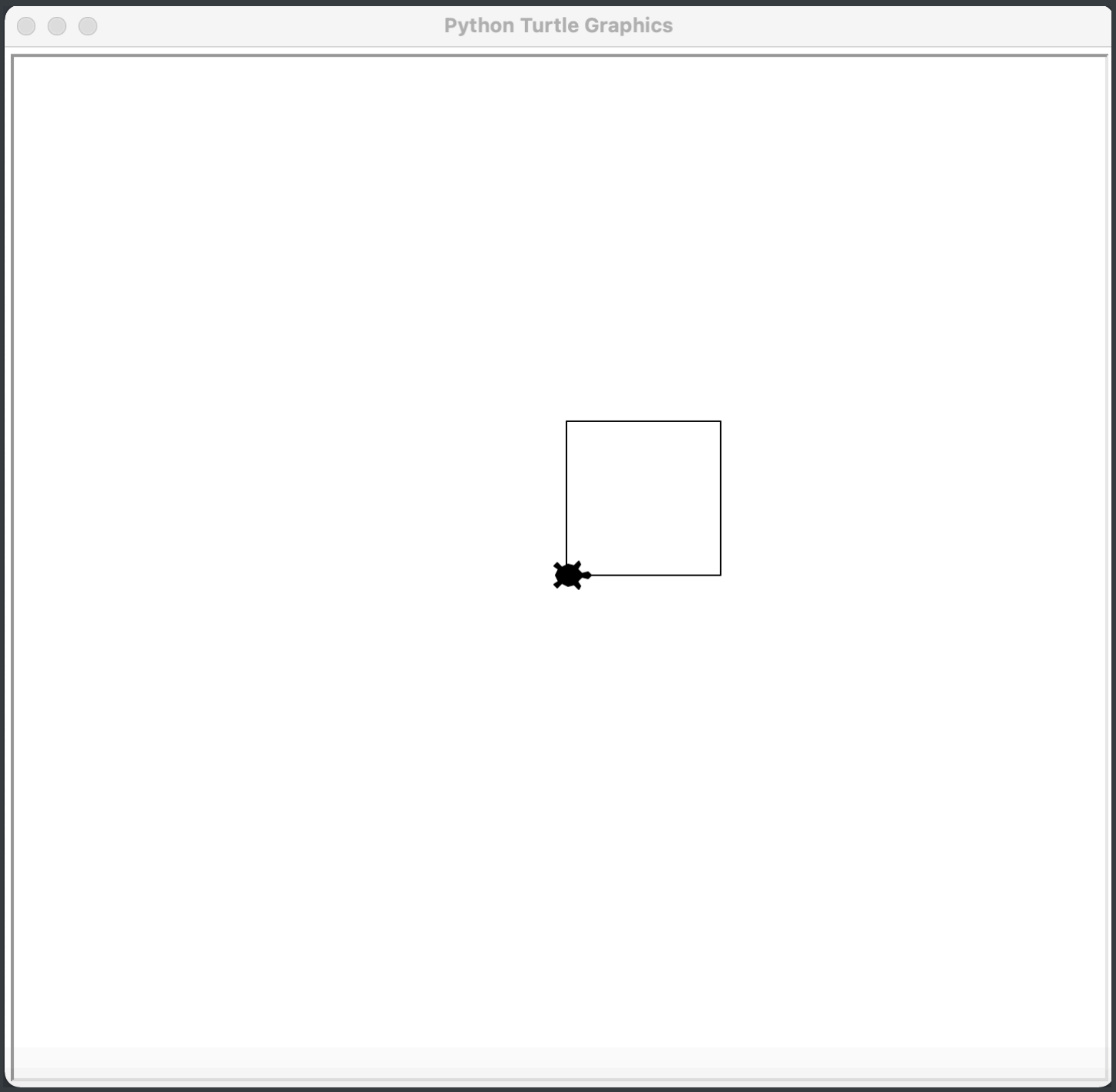
Creando un Cuadrado

Para esta parte se creará un cuadrado utilizando lo anteriormente visto

```
def cuadrado():
    shape("turtle")
    forward(100)
    left(90)
    forward(100)
    left(90)
    forward(100)
    left(90)
    forward(100)
    left(90)
```

¡¡¡Importante!!!: Cada vez que se cree una función no basta con sólo crearla, también hace falta mandarla a llamar, esto lo conseguimos escribiendo el nombre de la función fuera de la misma, dándole a entender a python que la mandaremos a llamar

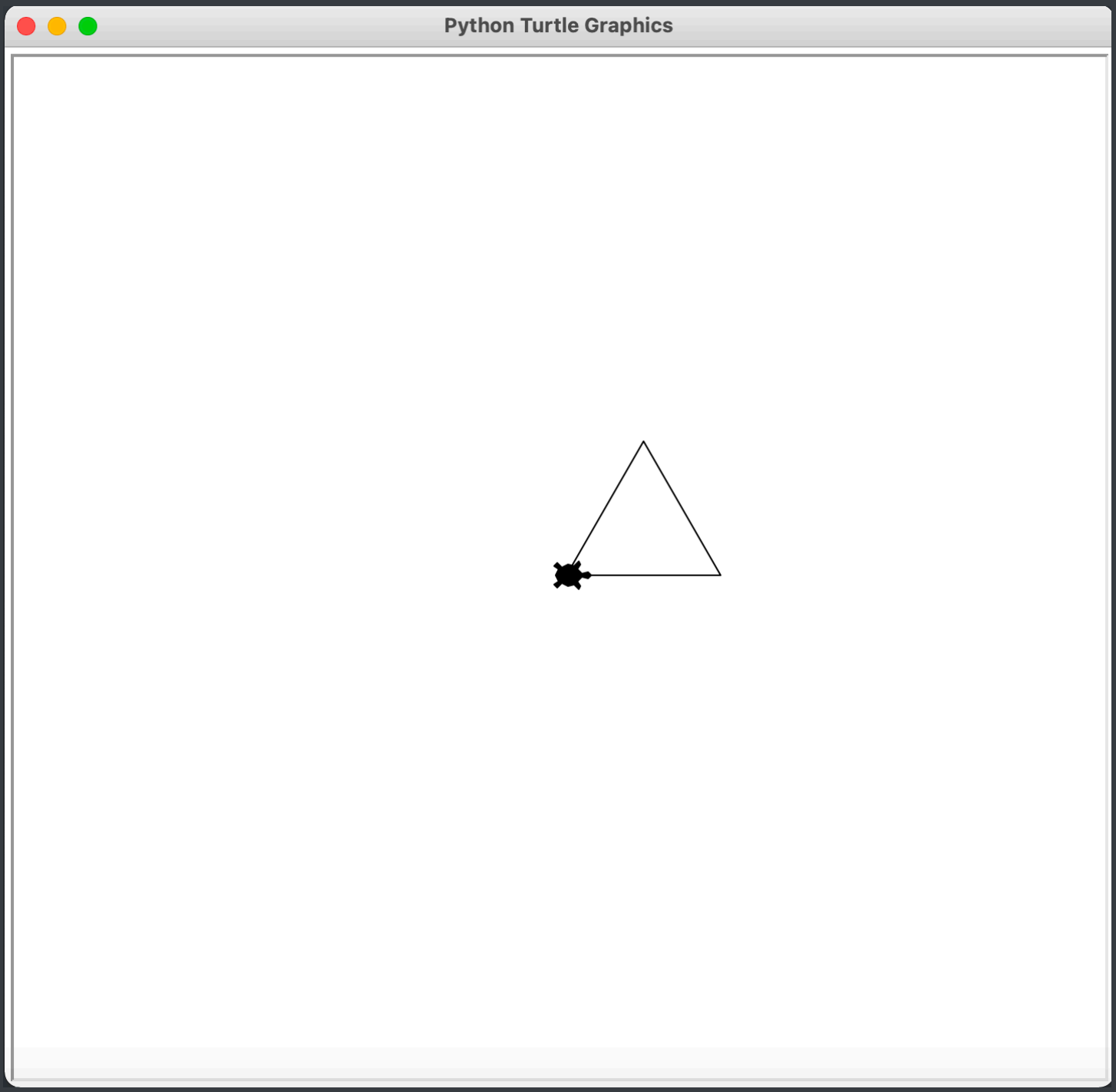
```
def cuadrado():
    shape("turtle")
    forward(100)
    left(90)
    forward(100)
    left(90)
    forward(100)
    left(90)
    forward(100)
    left(90)
cuadrado()
```



Creando un Triángulo

Para este caso se necesita un giro diferente y un numero diferente de pasos a dar, ya que la figura tiene 3 lados, se darán solo 3 *forward*

```
def triangulo():
    shape("turtle")
    forward(100)
    left(120)
    forward(100)
    left(120)
    forward(100)
    left(120)
triangulo()
```



1.3.-Configurar Tu tortuga

Tamaño de la figura

El tamaño de la figura puede ser modificado de la siguiente forma

```
shapesize(2) ###Tamaño de la figura
```

Esto nos permite agrandar o hacer mas pequeña nuestra tortuga, el numero que va dentro del parentesís, puede ser un número entero, sin embargo se recomienda no hacer el numero tan grande, ya que podría ser contraproducente

Color de la pluma (línea)

Para este caso sólo se colorea el borde de la pluma, es decir del trazo, no se colorea el interior

```
pencolor("green") ###Color de la tortuga
```

El color que va entre los parentesís debe ser un color en inglés, sin embargo este puede ser cualquiera de los colores válidos que se muestran a continuación: https://trinket.io/docs/colors

Grosor de la pluma (línea)

En este caso se verá el grosor de la pluma y por consiguiente el grosor del trazo

```
pensize(2) ##Grosor de la tortuga
```

Esto nos permite agrandar o hacer mas pequeño el trazo de la tortuga, el numero que va dentro del parentesís, puede ser un número entero, sin embargo se recomienda no hacer el numero tan grande, ya que podría ser contraproducente

Relleno del trazo

Para este caso particular se usan 3 líneas de código, cada una con una utilidad diferente

```
color("green", "blue") ###Primer argumento linea de color, segundo
argumento Color de relleno
```

Si no se define un segundo color, se tomará de default que el relleno será del mismo

Marcar el comienzo de relleno: Se debe poner estas líneas cuando se marca el inicio de la figura o trazo que se va a rellenar

```
begin_fill()###Marcar inicio de relleno
```

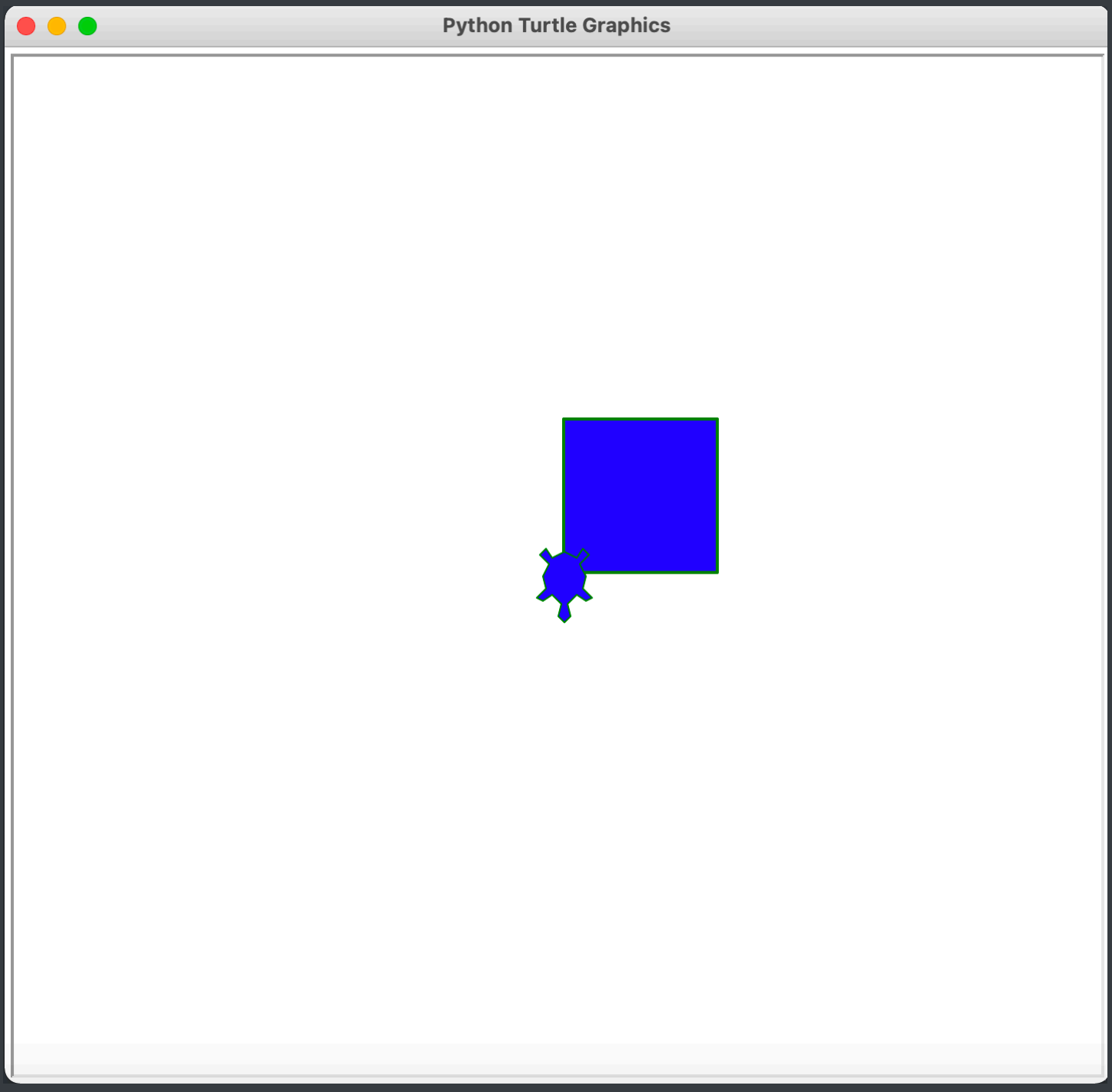
Marcar el fin de relleno: Se debe poner estas líneas cuando se marca el Fin de la figura o trazo que se va a rellenar, este en automático coloreara del color definido en las líneas anteriores

```
end_fill() ###fin de relleno
```

Código Completo

```
##Creando una tortuga personalizada
from turtle import *
shape("turtle") ##forma de la tortuga
shapesize(2) ###Tamaño de la fugura
pencolor("green") ###Color de la tortuga
pensize(2) ##Grosor de la tortuga
def cuadrado():
    ##Emepezar a rellenar
    frank.color("green", "blue") ###Primer argumento linea de color,
segundo argumento Color de relleno
    frank.begin_fill()###Marcar inicio de relleno
    ###Creando un cuadrado
    frank.fd(100)
    frank.lt(90)
    frank.fd(100)
    frank.lt(90)
    frank.fd(100)
    frank.lt(90)
    frank.fd(100)
```

```
    frank.end_fill() ###fin de relleno
    input()
```



2.- Generando una Tortuga Propia

Para este tramo crearemos un objeto de tipo tortuga, es decir haremos un clon de la estructura original, esto nos da la capacidad de modificar, crear y clonar varias tortugas para un mismo programa, esto se traduce que tendríamos varios pinceles funcionando.

```
##Creando una tortuga personalizada
import turtle
##Creando una tortuha con nombre
frank = turtle.Turtle()
frank.shape("turtle") ##forma de la tortuga
frank.shapesize(2) ###Tamaño de la fugura
frank.pencolor("green") ###Color de la tortuga
frank.pensize(2) ##Grosor de la tortuga
```

En el tramo anterior se describe la configuración de la tortuga, de manera que ésta pueda ser personalizada al gusto que se decida

Código Completo

A continuación, se presenta el código completo

```
##Creando una tortuga personalizada
import turtle
##Creando una tortuha con nombre
frank = turtle.Turtle()
frank.shape("turtle") ##forma de la tortuga
frank.shapesize(2) ###Tamaño de la fugura
frank.pencolor("green") ###Color de la tortuga
frank.pensize(2) ##Grosor de la tortuga
def cuadrado():
    ##Emepezar a rellenar
```

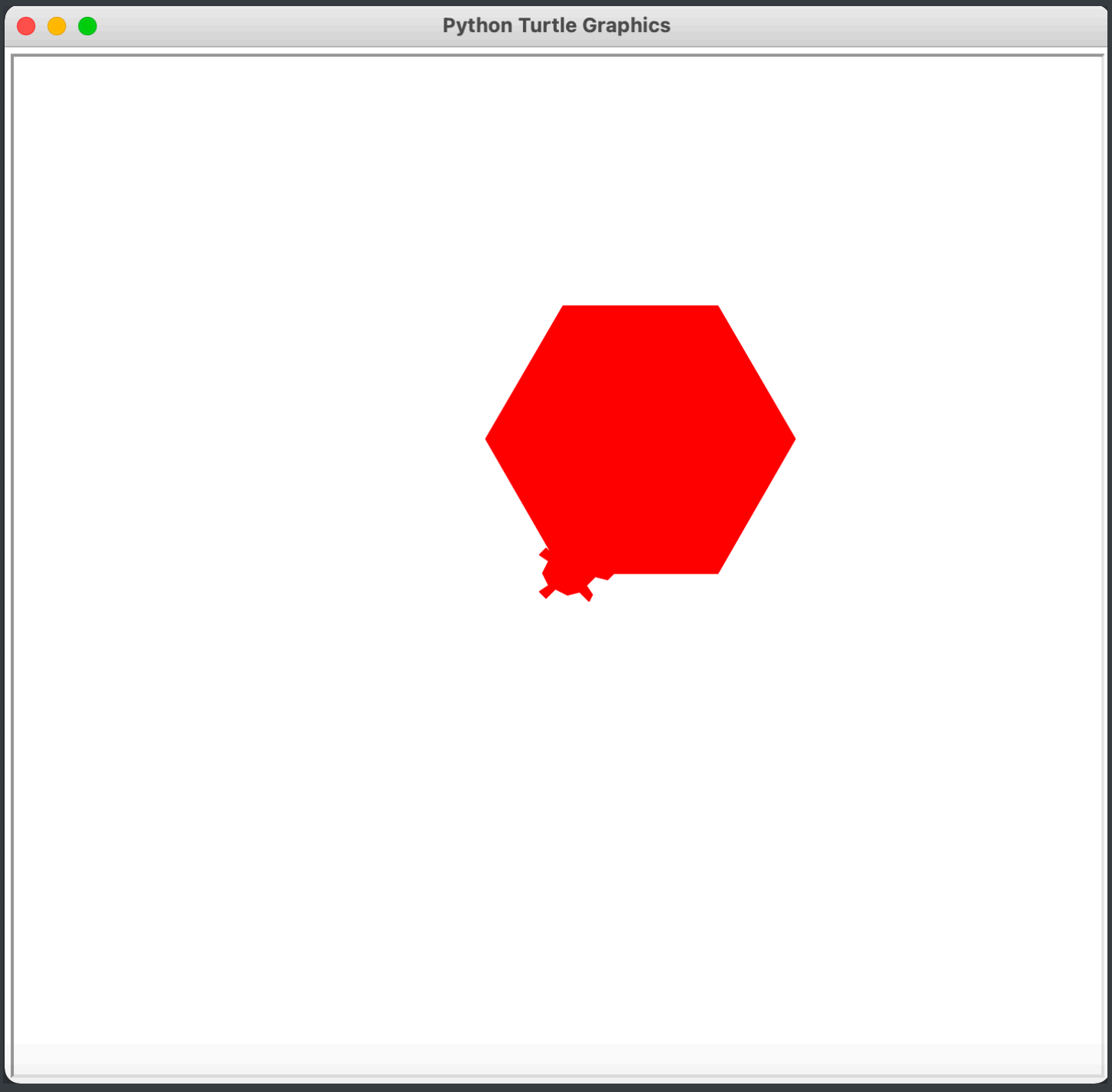
```
    frank.color("green", "blue") ###Primer argumento linea de color,
segundo argumento Color de relleno
    frank.begin_fill()###Marcar inicio de relleno
    ###Creando un cuadrado
    frank.fd(100)
    frank.lt(90)
    frank.fd(100)
    frank.lt(90)
    frank.fd(100)
    frank.lt(90)
    frank.fd(100)
    frank.end_fill() ###fin de relleno
    input()

def triangulo():
    ##Emepezar a rellenar
    frank.color("red") ###Primer argumento linea de color, segundo
argumento Color de relleno
    frank.begin_fill()###Marcar inicio de relleno
    ###Creando un cuadrado
    frank.fd(100)
    frank.lt(120)
    frank.fd(100)
    frank.lt(120)
    frank.fd(100)
    frank.end_fill() ###fin de relleno
    input()

def pentagono():
    ##Emepezar a rellenar
    frank.color("red") ###Primer argumento linea de color, segundo
argumento Color de relleno
    frank.begin_fill()###Marcar inicio de relleno
```

```
    ###Creando un Pentagono
    frank.fd(100)
    frank.lt(360/5)
    frank.fd(100)
    frank.lt(360/5)
    frank.fd(100)
    frank.lt(360/5)
    frank.fd(100)
    frank.lt(360/5)
    frank.fd(100)
    frank.lt(360/5)
    frank.end_fill() ###fin de relleno
    input()
#pentagono()
def hexagono():
    ##Emepezar a rellenar
    frank.color("red") ###Primer argumento linea de color, segundo
argumento Color de relleno
    frank.begin_fill()###Marcar inicio de relleno
    ###Creando un Hexágono
    frank.fd(100)
    frank.lt(360/6)
    frank.fd(100)
    frank.lt(360/6)
    frank.fd(100)
    frank.lt(360/6)
    frank.fd(100)
    frank.lt(360/6)
    frank.fd(100)
    frank.lt(360/6)
    frank.end_fill() ###fin de relleno
```

```
    input()
hexagono()
```



3.- Usando Ciclos de Repetición

Hemos estado viendo las formas en las cuales se puede crear diferentes prograas que hagan diferentes actividades. Sin embargo si hemos sido observadores, notaremos que algunas líneas se repiten demasiado y lelgan a ser engorrosos, ¿Ésto se puede corregir? La respuesta corta es sí, para ello se usa un concepto llamado **Ciclo de Repetición**,

3.1.- ¿Qué son los ciclos de repetición?

Los ciclos de repetición son una forma de poder repetir varias líneas de código, reduciéndolo en pocas líneas. Para ello usamos 2 ciclos de repetición *principalemnte*, éstos son

- **For:** Éste será el principal ciclo de repetición el cual se verá en éstos apuntes. Este contiene una variable, la cual se va repitiendo las veces en las que nosotros le digamos que lo haga. De ésta forma, nosotros podemos repetir algunas líneas que nosotros querramos.
  - **Código:** El código se demuestra a continuación:

    ```
    for i in range(numero_veces):
      ###veces en las que se repita algún código
    ```

    En este ejemplo, podemos ver 2 cosas principales, la variable **i**, la variable **numero_veces.**
  - **Variable i:** Esta variable no tiene un nombre en específico, se elige la letra *i*, por ser la letra inicial de la palabta *iterations*, esta variable toma el valor de las veces que le indicamos en el ciclo de repetición.
  - **Variable numero_veces:** Esta variable es el máximo que nosotros podemos definir. **NOTA: Al numero de veces máximo se le resta 1, ya que python comienza desde el número 0**

  Ejemplo: En el siguiente código se muestra un ciclo de repetición que va desde el 0 hasta el número 5-1, de esta forma, cuando imprimimos el valor de *i*, este cambiará el número de veces deefinidas en *range*

```
for i in range(5):
  print(i)
```

```
>>> for i in range(5):
...     print(i)
...
0
1
2
3
4
>>>
```

- **While:** Para este ciclo de repetición necesitamos una variable adicional, esta se hace de manera comprobatoria, es decir, se necesita una variable para preguntar si aun sigue el ciclo, imagina lo siguiente *¿Como se cuando estoy dentro de una alberca?*, La respuesta es simple,*mientras estés dentro de la alberca*
  - **Ejemplo:** En el siguiente código se usa una variable llamada *v1*, esta variable se comprobará si es menor a 0 y se iniciará con un valor 5, para poder comprobar que se rompa la condición, se debe de ir restando.

```
v1 = 5
while(v1>0):
  print(v1)
  v1 = v1 - 1
```

```
>>> v1 = 5
>>> while(v1>0):
...   print(v1)
...   v1 = v1 - 1
...
5
4
3
2
1
>>>
```

3.2.- ¿Cómo uso los ciclos de repetición aquí?

Tomaremos como ejemplo el siguiente código

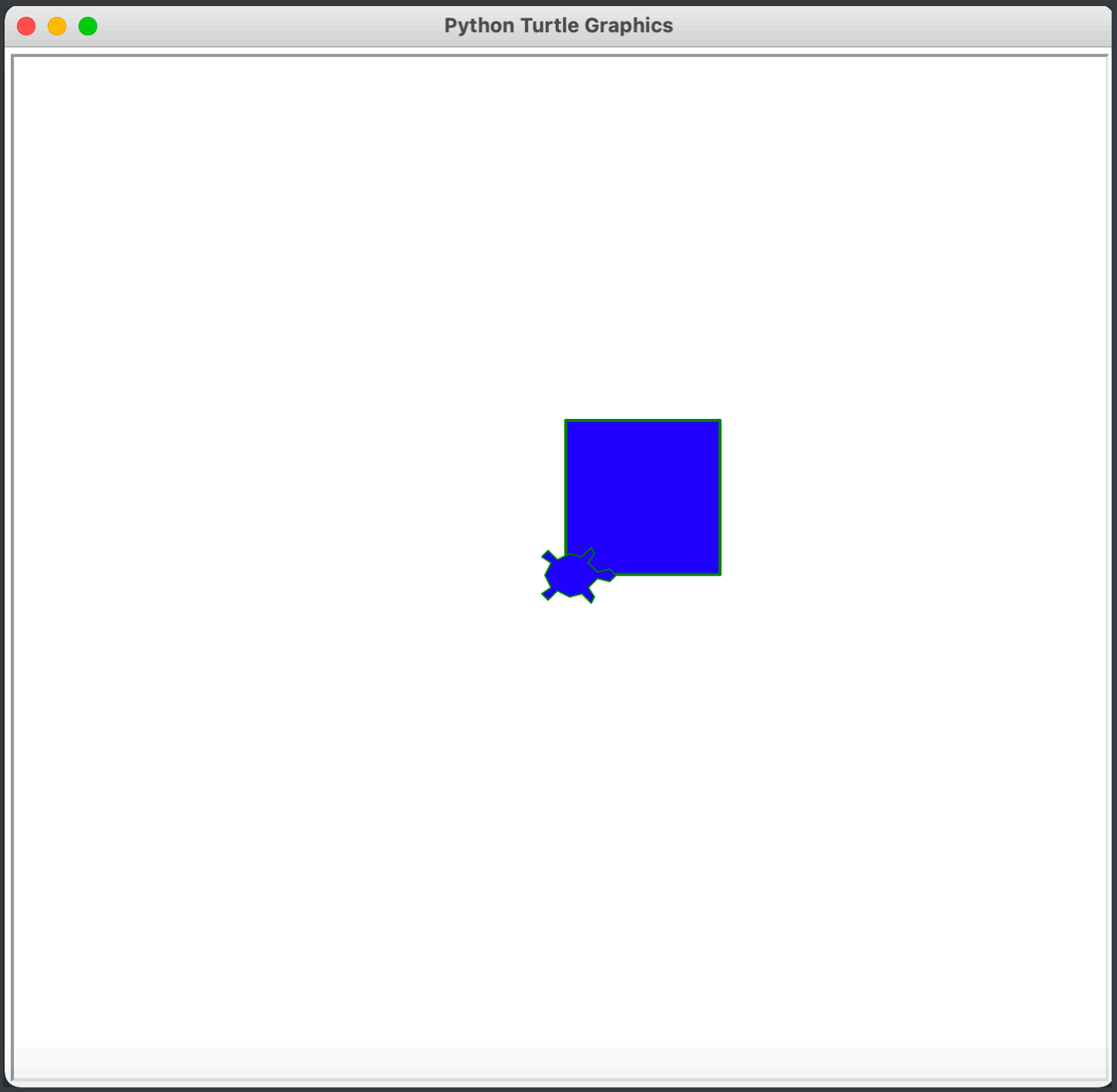
```
##Creando una tortuga personalizada
import turtle
##Creando una tortuha con nombre
frank = turtle.Turtle()
frank.shape("turtle") ##forma de la tortuga
def cuadrado():
    ##Emepezar a rellenar
    frank.color("green", "blue") ###Primer argumento linea de color,
segundo argumento Color de relleno
    frank.begin_fill()###Marcar inicio de relleno
    ###Creando un cuadrado
    frank.fd(100)
    frank.lt(90)
    frank.fd(100)
    frank.lt(90)
    frank.fd(100)
    frank.lt(90)
    frank.fd(100)
```

```
    frank.end_fill() ###fin de relleno
    input()
```

En este código se presenta el código para la figura cuadrado, sin embargo, podemos observar que las líneas *frank.lt(90)* y *frank.fd(100)*, se repiten 4 veces, las cuales son el número de lados que tiene el cuadrado, de esta forma podemos simplificar el código usando un ciclo de repetición, poniendo como máximo el número 4.

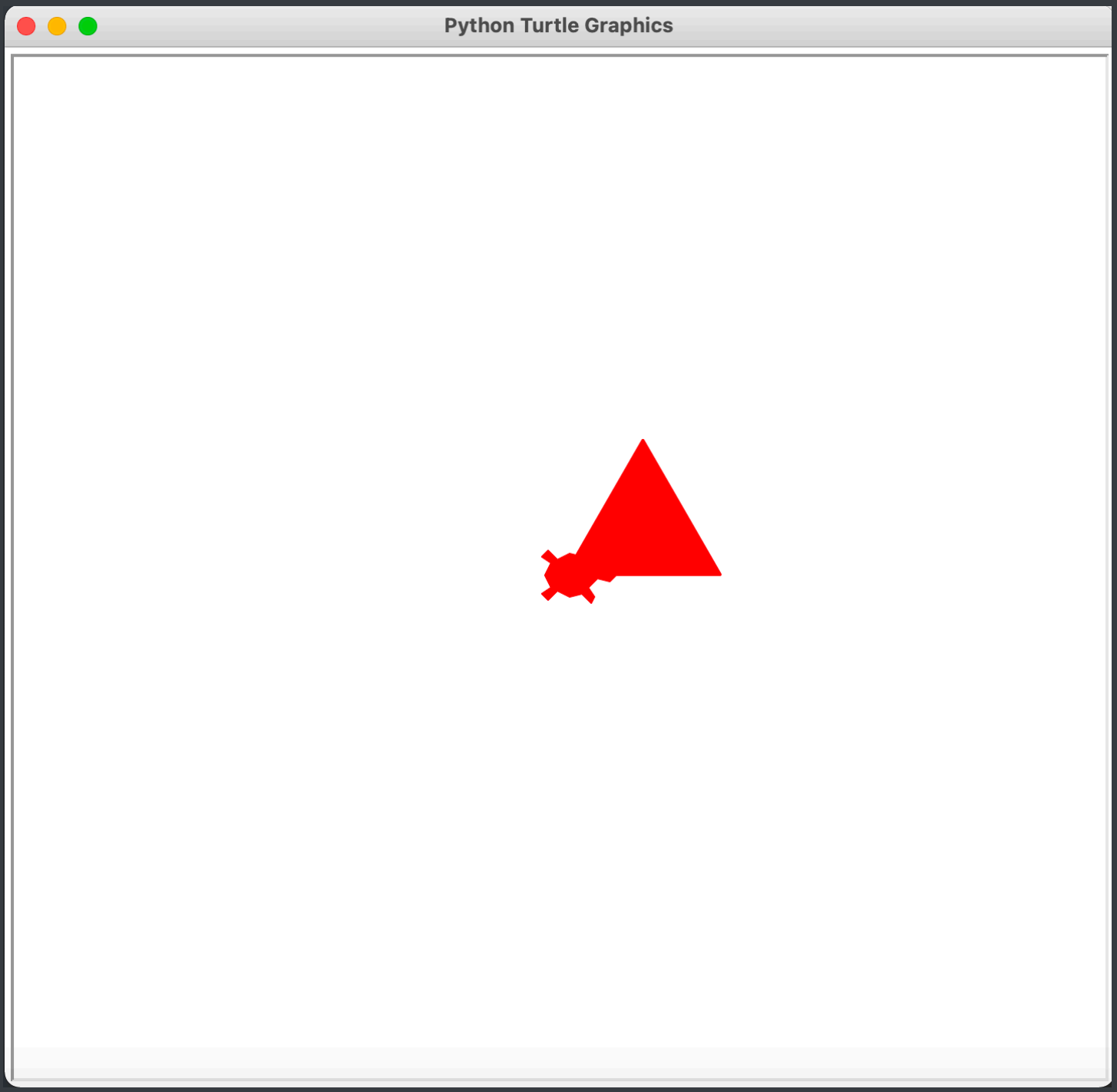
Cuadrado

```
##Creando una tortuga personalizada
import turtle
##Creando una tortuha con nombre
frank = turtle.Turtle()
frank.shape("turtle") ##forma de la tortuga
def cuadrado():
    ##Emepezar a rellenar
    frank.color("green", "blue") ###Primer argumento linea de color,
segundo argumento Color de relleno
    frank.begin_fill()###Marcar inicio de relleno
    ###Creando un cuadrado
    for i in range(4):
      frank.fd(100)
      frank.lt(90)
    frank.end_fill() ###fin de relleno
    input()
```



Triángulo

```
def triangulo():
    ##Emepezar a rellenar
    frank.color("red") ###Primer argumento linea de color, segundo
argumento Color de relleno
    frank.begin_fill()###Marcar inicio de relleno
    ###Creando un Triángulo
    frank.fd(100)
    frank.lt(120)
    frank.fd(100)
    frank.lt(120)
    frank.fd(100)
    frank.end_fill() ###fin de relleno
    input()
```



Pentágono

```
def pentagono():
    ##Emepezar a rellenar
    frank.color("red") ###Primer argumento linea de color, segundo
argumento Color de relleno
    frank.begin_fill()###Marcar inicio de relleno
    ###Creando un Pentagono
    for i in range(5):
      frank.fd(100)
      frank.lt(360/5)
    frank.end_fill() ###fin de relleno
    input()
```

4.- Interctuando con varias Tortugas

Para esta última parte checaremos varias formas de interactuar con varios factores, la mayoría de éstos códigos se incluyen en la sección **fantasia**, ubicado en el repositorio: LINK

En esta ocasión, cómo la vez pasada, usaremos una instancia para crear diferentes tortugas, esto nos servirá para pdoer crear varias tortugas y que se puedan mover entre ellas.

Tortuga 1

```
##Tortuga 1
frank = turtle.Turtle()
colors =['orange', 'red', 'pink', 'yellow', 'blue', 'green']
frank.pen(speed=25)#setting the speed of the pen to 25
frank.shape('turtle')#setting the shape of the turtle as "classic
```

Tortuga 2

```
##Tortuga 2
frank2 = turtle.Turtle()
colors2 =['dark turquoise', 'turquoise', 'medium spring green', 'yellow',
'blue', 'green']
frank2.pen(speed=25)#setting the speed of the pen to 25
frank2.shape('turtle')#setting the shape of the turtle as "classic
```

Tortuga 3

```
##Tortuga 3
frank3 = turtle.Turtle()
colors3 =['orange red', 'light salmon', 'bisque', 'yellow', 'blue',
'green']
frank3.pen(speed=25)#setting the speed of the pen to 25
frank3.shape('turtle')#setting the shape of the turtle as "classic
```

De esta forma definimos 3 tortugas con diferentes características y nombres.

Definiremos una función, la cual nos permitirá hacer una **recursión**, es decir, mandaremos a llamar la función dentro de la misma función.

```
def spiral_shape(name_turtle,p,c, colors):
    if p > 0:
        name_turtle.forward(p)
        name_turtle.right(c)
        name_turtle.pencolor(colors[p%6])
        spiral_shape(name_turtle,p-5,c, colors)
```

A continuación se muestra el código completo y el resultado

```
import turtle
##COLORES: https://trinket.io/docs/colors
```

```
###Escenario
screen = turtle.Screen()
screen.bgcolor('black')
##Tortuga 1
frank = turtle.Turtle()
colors =['orange', 'red', 'pink', 'yellow', 'blue', 'green']
frank.pen(speed=25)#setting the speed of the pen to 25
frank.shape('turtle')#setting the shape of the turtle as "classic

##Tortuga 2
frank2 = turtle.Turtle()
colors2 =['dark turquoise', 'turquoise', 'medium spring green', 'yellow',
'blue', 'green']
frank2.pen(speed=25)#setting the speed of the pen to 25
frank2.shape('turtle')#setting the shape of the turtle as "classic

##Tortuga 3
frank3 = turtle.Turtle()
colors3 =['orange red', 'light salmon', 'bisque', 'yellow', 'blue',
'green']
frank3.pen(speed=25)#setting the speed of the pen to 25
frank3.shape('turtle')#setting the shape of the turtle as "classic

def spiral_shape(name_turtle,p,c, colors):
    if p > 0:
        name_turtle.forward(p)
        name_turtle.right(c)
        name_turtle.pencolor(colors[p%6])
        spiral_shape(name_turtle,p-5,c, colors)
###Tortuga 1
frank.setpos(-300, 0)#setting the position as given
spiral_shape(frank,400, 121, colors)#calling the function with the given
arguments
```

```
###Tortuga 2
frank2.setpos(-600,0)
spiral_shape(frank2,400, 121, colors2)#calling the function with the given
arguments
###Tortuga 3
frank3.setpos(0,100)
spiral_shape(frank3,400, 121, colors3)#calling the function with the given
arguments
screen.exitonclick()
```