El futuro digital es de todos

MinTIC

Gráficos con Turtle y depuración

OPERADO POR:

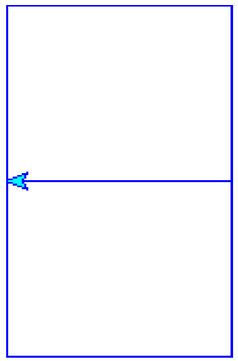
Universidad Tecnológica de Pereira

Mision TIC 2022

RUTA DE APRENDIZAJE 1

De igual manera es posible, que turtle haga dos figuras diferentes que se encuentre separadas, para eso debemos utilizar otra instrucción, que le indique que la tortuga debe moverse, pero no debe dibujar aun, si no se lo indicamos y solo le pedimos que dibuje otro cuadro tendríamos el siguiente resultado:

```
24
25  tortuga.forward(100)
26
27  tortuga.left(90)
28
29  tortuga.forward(100)
30
31  tortuga.left(90)
32
33  tortuga.forward(100)
34
35  tortuga.left(90)
36
37  tortuga.forward(100)
38
39
40
41  tortuga.forward(100)
42
43  tortuga.left(90)
44
45  tortuga.forward(100)
46
47  tortuga.left(90)
48
49  tortuga.forward(100)
50
51  tortuga.left(90)
52
53  tortuga.forward(100)
54
```

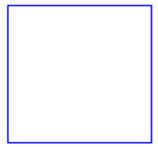


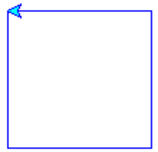
El código me indica que la tortuga debe dibujar dos cuadrados, sin embargo los dibuja juntos:

Vamos a indicarle, que necesitamos dibujar ambos cuadrados de forma separada usando la instrucción ***tortuga.penup()*** Lo cual le dice a la tortuga en otras palabras que levante su lápiz y no dibuje una línea en las siguientes instrucciones, ahora vamos a decirme que camine usando la instrucción foward: ***tortuga.forward(100)*** ahora vamos a decirle que vuelva usar el lápiz y comience a dibujar, usando la instrucción: ***tortuga.pendown()***

Así tendremos:

```
24
25  tortuga.forward(100)
26
27  tortuga.left(90)
28
29  tortuga.forward(100)
30
31  tortuga.left(90)
32
33  tortuga.forward(100)
34
35  tortuga.left(90)
36
37  tortuga.forward(100)
38
39
40  tortuga.penup()
41
42  tortuga.forward(100)
43
44  tortuga.pendown()
45
46
47
48  tortuga.forward(100)
49
50  tortuga.left(90)
51
52  tortuga.forward(100)
53
54  tortuga.left(90)
55
56  tortuga.forward(100)
57
58  tortuga.left(90)
59
60  tortuga.forward(100)
61
```





Depuración

La depuración es una de las herramientas más poderosas disponibles para cualquier desarrollador. En pocas palabras, se trata de ejecutar el código líneo a línea, parando en momentos claves para analizar el estado de los procedimientos y las variables. Esto nos permitirá identificar y corregir errores en nuestro código que de otra forma serían mucho más difíciles de encontrar.

La depuración de un código es muy importante para lograr encontrar errores en el código que Python no nos puede indicar.

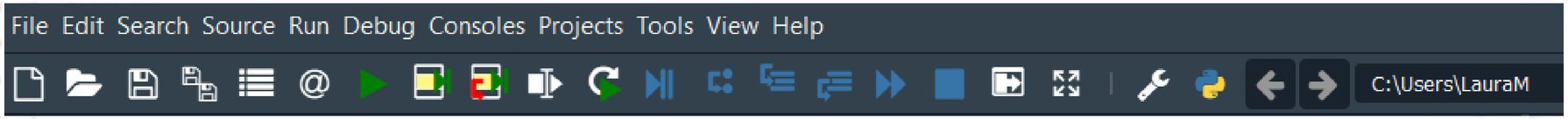
En general, los programadores gastan en promedio la mitad del tiempo de desarrollo trabajando en la etapa de depuración de código. ¡No está mal cometer errores al programar!

Por eso es necesario que nos apoyemos de las herramientas de Python para poder depurar o revisar el código que estamos implementando.

Depurando en Spyder

Para iniciar la depuración de nuestro programa desde spyder utilizaremos la barra de tareas, dando click a la pestaña "Depurar" / "Debug".

De igual forma, una vez iniciado el proceso, los botones azules nos permitirán movernos dentro del código, paso a paso. Los principales atajos que facilitarán esta tarea son: Ctrl + F10 para ejecutar la línea actual y Ctrl + F12 para ejecutar el código hasta su siguiente parada o finalización.



Explorador de variables y breakpoints

A continuación introduciremos dos elementos fundamentales al proceso de depuración: Los puntos de quiebre (breakpoints) y la ventana de variables (variable explorer).

Al depurar nuestro programa, este se detendrá en la línea dónde encuentre el primer **breakpoint** de forma que podamos iniciar con el proceso de ejecución paso a paso.

Para colocar un punto de quiebre o break point debemos dar click sobre la barra gris a la izquierda de nuestro código, al lado del número de línea. Lo que se visualizará cómo un punto rojo.

Una vez nuestro código se encuentra en proceso de depuración, podemos ver el valor de las variables en tiempo real en la pestaña "Variable explorer" que se encuentra encima de la consola de spyder. Esto nos dará una intuición de lo que realmente está pasando en nuestro programa.

Primero haremos la prueba con el código que implementamos en **Spyder**, Vamos a revisa r su paso a paso usando la herramienta de depuración.

Ahora vamos a hacer click al lado del numero que esta relacionado con la línea de código, en este ejemplo seria la línea numero 10, y nos debe aparecer un punto rojo. Con esto podemos iniciar el proceso.

```
9
10  def area(parametro1:float, parametro2:float)->float:
11
12
13      area= (parametro1 * parametro2)/(2)
14
```

Solo debemos hacer click sobre la opción debug file, y con eso Python comienza el proceso y nos mue stra la información en consola.

```python
def area(parametro1:float, parametro2:float)->float:


    area= (parametro1 * parametro2)/(2)

    return area

lado1= 10.2
lado2= 30.0


respuesta=area(lado1,lado2)
print("El area del triagngulo es: " +str( respuesta));
```

```
Help | Variable explorer | Plots

Console 1/A
El area del triagngulo es: 153.0

In [2]: debugfile('C:/Users/LauraM/Downloads/codigo.py', wdir='
> c:\users\lauram\downloads\codigo.py(2)<module>()
      1 # -*- coding: utf-8 -*-
----> 2 """
      3 Created on Thu Jul  1 16:48:47 2021
      4
      5 @author: LauraM


ipdb> continue
> c:\users\lauram\downloads\codigo.py(10)<module>()
      8 # Función para calcular el área de un triangulo
      9
4--> 10 def area(parametro1:float, parametro2:float)->float:
     11
     12


ipdb>

IPython console | History
```

En la consola vamos a tener información relacionada con el proceso paso a paso, por lo tanto, nos indica donde comenzó el proceso y donde se detuvo, para este ejemplo se detiene en la fila 10, que fue la que definimos.

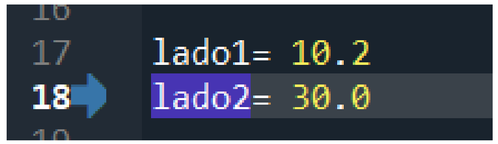
Si utilizamos las otras herramientas podemos avanzar en el código, paso a paso.

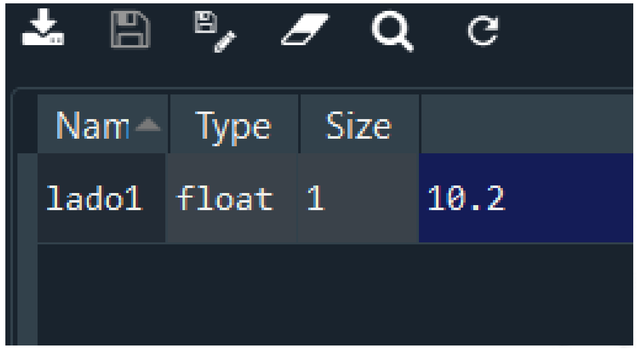
Así podemos seguir avanzando para descubrir que val ores entrega Python y como podemos solucionarlo.

Si hacemos click en la siguiente opción para debug, el código avanza hasta el momento donde aparecen los valores que necesita la función, recordemos que el no va a entrar a la función hasta que la llamemos, por eso vemos un a flecha azul indicando donde va el proceso.

```
9
10  def area(parametro1:float, parametro2:float)->float:
11
12
13      area= (parametro1 * parametro2)/(2)
14
15      return area
16
17  lado1= 10.2
18  lado2= 30.0
19
20
21  respuesta=area(lado1,lado2)
22  print("El area del triagngulo es: " +str( respuesta));
23
```

Si hacemos click nuevamente en el botón de **run current line**, el código avanza una línea mas, en este caso la siguiente línea ejecutable.





Ahora aparece en la parte superior aparecerá la variable lado1, con sus características según lo que lee el programa.

Si hacemos click nuevamente en el botón de run current line, el código avanza una línea mas, en este caso la siguiente línea ejecutable.

Ahora aparece el proceso en la línea 21, y tenemos la infor mación de ambas variables, con esto podemos revisar si e xiste algún error en los cálculos y es un error que no pode mos encontrar con la sintaxis.

```
20
21  respuesta=area(lado1,lado2)
22  print("El area del triagngulo es: " +str( respuesta));
23
```

Nam	Type	Size	
lado1	float	1	10.2
lado2	float	1	30.0

Si hacemos click nuevamente en el botón de run current line, el código avanza una línea mas, en este caso la siguiente línea ejecutable.

```
21    respuesta=area(lado1,lado2)
22    print("El area del triagngulo es: " +str( respuesta));
```

Ahora aparece el proceso en la línea 21, y tene mos la información de ambas variables, con es to podemos revisar si existe algún error en los cálculos y es un error que no podemos encont rar con la sintaxis.

Name	Type	Size	
lado1	float	1	10.2
lado2	float	1	30.0
respuesta	float	1	153.0

Al hacer click nuevamente en la opción **run current line** logramos llegar al final del código, y podemos ver el resultado en consola que entrega el print.

```
El area del triagngulo es: 153.0
--Return--

ipdb>
```

El futuro digital es de todos

MinTIC

GRACIAS

OPERADO POR:

Universidad Tecnológica de Pereira

MisiónTIC2022