# Taller Integrador - Python.

### Clase 8

### Sábado 16/05/2020 (Chivilcoy)

### Martes 19/05/2020 CABA.

# Gráficos con Turtle.

Hacer gráficos con este módulo es sencillo, pero permite, si le dedicamos el tiempo necesario, hacer verdaderas gráficas complejas para representar nuestros datos.

Lo primero que debemos hacer es importar el módulo, o dependiendo de la tarea que necesitemos hacer, importamos todo:



### Instrucciones.

FORWARD / BACK: ir hacia adelante / ir hacia atrás.

RIGHT/LEFT: girar a la derecha / girar a la izquierda.

La sintaxis es muy sencilla, le pasamos a cada función, la cantidad de pixeles que queremos que se mueva, como parámetro o los grados en caso de querer que gire.

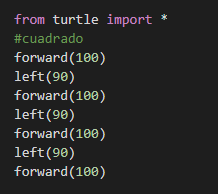
Ejercicios simples propuestos:

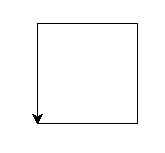
\*\*\* Escribir el código para generar un triángulo equilátero.

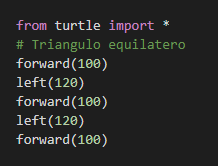
\*\*\* Escribir el código para generar un cuadrado.

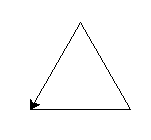
\*\*\* Escribir una función que reciba el lado del cuadrado y lo dibuje.

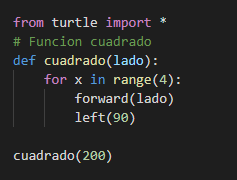
Soluciones sugeridas.

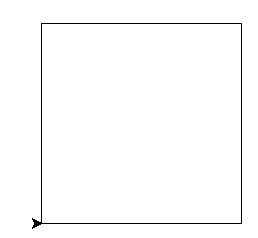










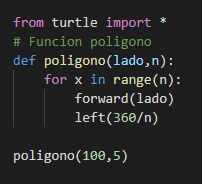


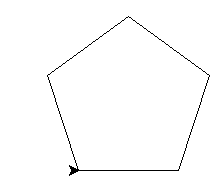
Haciendo un poco de memoria de lo visto en la secundaria, podríamos crear una función que nos dibuje cualquier polígono regular (es un polígono cuyos lados y ángulos interiores son iguales entre sí).

Esta función debería recibir la cantidad de lados que queremos dibujar y la longitud de dicho lado. Luego, sabiendo que la suma de los ángulos internos del polígono es 360 grados, podemos afirmar que cada ángulo interno de nuestro polígono será 360 grados dividido la cantidad de lados.

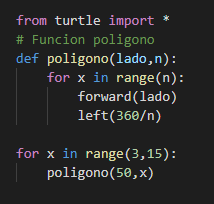
\*\*\* Crear una función que dibuje polígonos regulares.

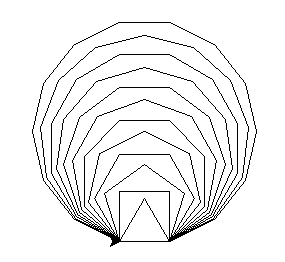
SOLUCIÓN PROPUESTA.





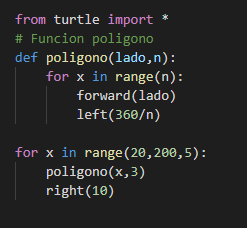
Podríamos dibujar varios polígonos con nuestra función, incrementando el número de lados:

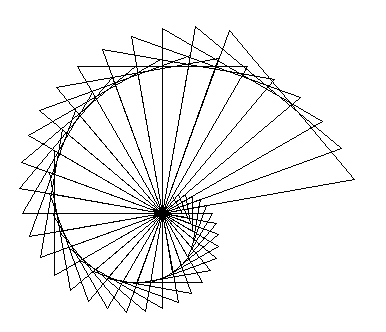




Otro ejemplo, hacemos una espiral con nuestra función polígono.

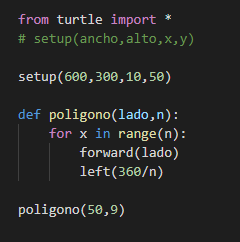
Le pasamos los datos para crear un triángulo, pero lo hacemos dentro de un bucle en el que luego de dibujar un triángulo, rotamos 10 grados a la derecha antes de dibujar el otro:

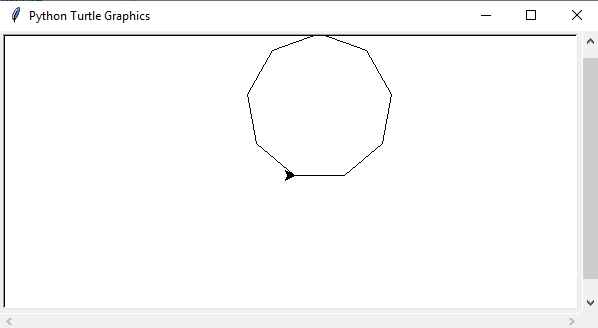




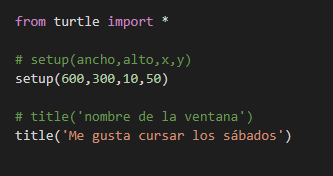
### Área / Ventana de dibujo.

Podemos definir el tamaño de nuestra ventana y su posición mediante SETUP.





Con la función TITLE le damos un título a nuestra ventana:



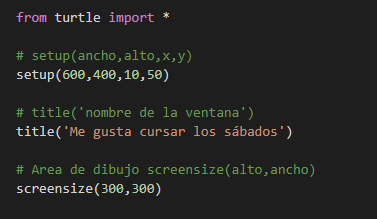


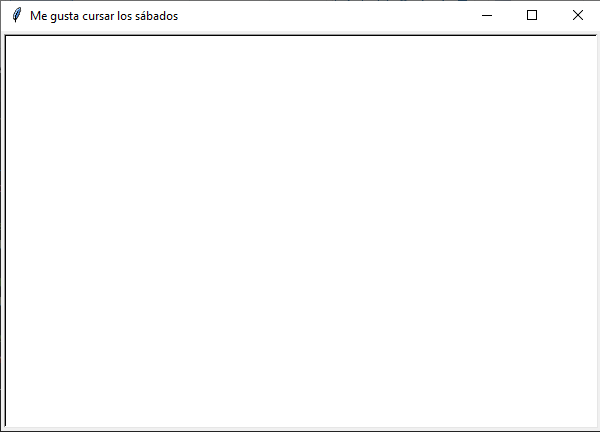
Si no se usa la función SETUP, la ventana se crea en el centro de la pantalla. El tamaño puede cambiarse a lo largo de la ejecución de nuestro programa, sin perder nada de lo ya hecho.

LA VENTANA CONTIENE AL AREA DE DIBUJO.

El área de dibujo inicialmente es de 400 x 300 píxeles, pero mediante la función SCREENSIZE se le puede dar el tamaño que sea necesario.

Si la ventana es más grande que el área de dibujo, NO se muestran las barras de desplazamiento, en cambio si el área de dibujo excede el tamaño de la ventana, SI se muestran las barras de desplazamiento que sean necesarias.



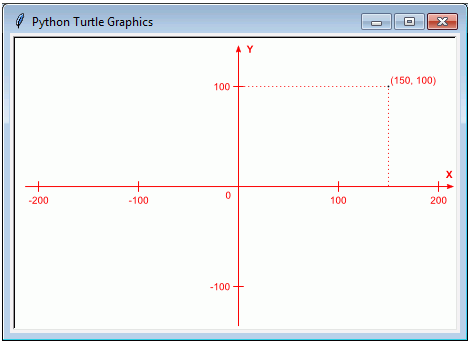


### Dibujando.

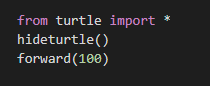
Dibujar gráficos de tortuga es similar a dibujar con un lápiz sobre papel. Las instrucciones de gráficos de tortuga permiten dibujar líneas y puntos, mover el lápiz de un sitio a otro, cambiar el color y grosor del trazo, colorear el interior de las figuras, etc.

Situar el cursor en un lugar del área de dibujo es equivalente a colocar la punta del lápiz en un punto de un papel y desplazar el cursor es equivalente a dibujar un segmento sobre el papel o desplazar el lápiz a otro punto del papel.

Las posiciones en el área de dibujo se localizan mediante coordenadas XY en el que cada píxel es una unidad y con origen en el centro de la ventana. La imagen siguiente muestra cuál sería la posición de los ejes de coordenadas y cuáles serían las coordenadas de un punto cualquiera.



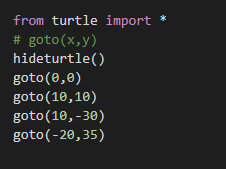
Las funciones SHOWTURTLE y HIDETURTLE nos posibilita mostrar o esconder el cursor.



Si no queremos que el cursor se vea, es mejor ocultarlo al inicio, lo que ocasiona que el programa se ejecute más rápido.

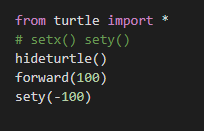
### Líneas.

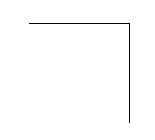
La función GOTO nos permite desplazar el cursor a un determinado punto del área de dibujo, a medida que lo vamos desplazando, entre una posición y la siguiente, se irá dibujando una línea.





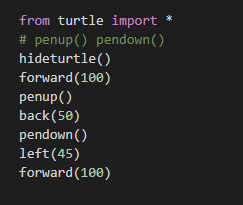
Las funciones SETX y SETY nos permiten desplazar el cursor a una determinada posición del área de dibujo modificando un solo componente de la coordenada (X o Y) manteniendo el otro fijo.

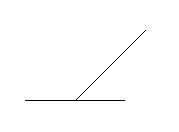




### LEVANTAR / BAJAR EL LÁPIZ.

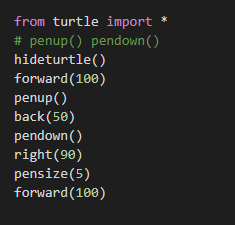
Podemos movernos de un lado al otro del área de dibujo sin dibujar la línea con la función PENUP (y con la función PENDOWN volvemos a bajar el lápiz).

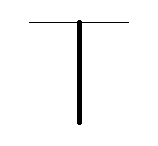




### GROSOR DEL TRAZO.

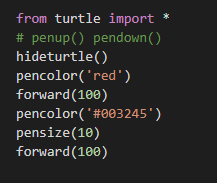
Podemos modificar el grueso de nuestro trazo con la función PENSIZE.





### COLOR DEL TRAZO.

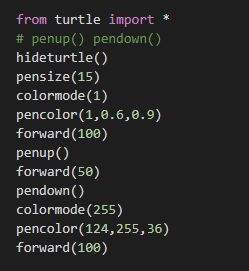
Existen tres formas de especificar el color del trazo. La primera es directamente, pasarle a la función PENCOLOR el nombre del color o su denominación hexadecimal.





Para las otras dos formas, debemos especificarle a Python el modo mediante la función COLORMODE, puede ser 255 o 1.

En ambos modos lo que estamos indicando es una combinación de los colores primarios rojo, verde y azul.

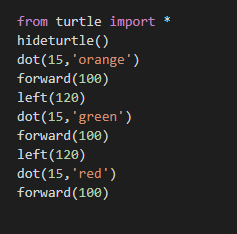


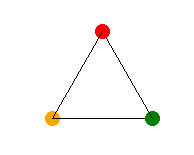


### Puntos.

La función DOT (grosor, color) permite dibujar puntos sobre el área de dibujo.

Veamos el siguiente ejemplo donde dibujamos un triángulo con sus vértices resaltados por un punto mas grueso y en colores.

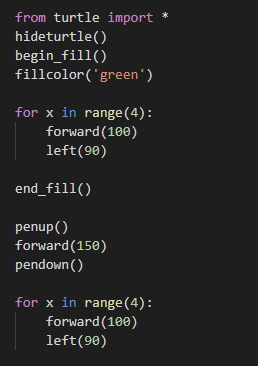


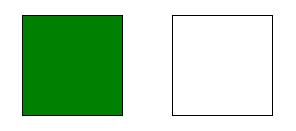


### Rellenar figuras.

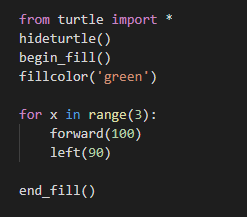
La función BEGIN\_FILL le indica al programa que, a partir de ahí, todas las figuras que se hagan, deben ir rellenas hasta que le indiquemos que deje de hacerlo mediante la función END\_FILL.

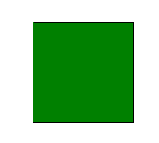
Para especificar el color de relleno, utilizaremos la función FILLCOLOR.





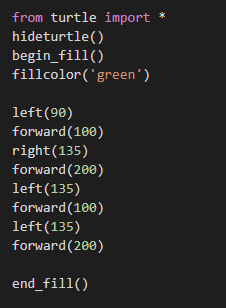
Si no dibujamos una figura completa, se rellena como si la hubiésemos terminado.

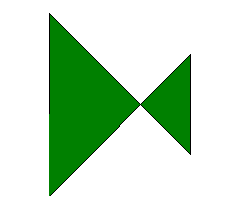




Si no especificamos un color de relleno, se hará por defecto con color negro.

Si dibujamos con líneas que se cruzan, Python interpretará cada figura que hagamos sobre el área de dibujo.

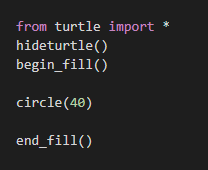


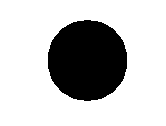


\*\*\* Probar con dibujos similares en los que no terminamos de dibujar la línea o levantamos el lápiz haciendo líneas incompletas.

### Círculos.

Con la función CIRCLE indicándole el radio, podemos dibujar un círculo.

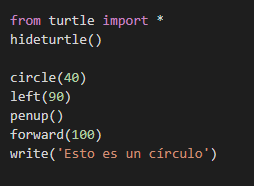


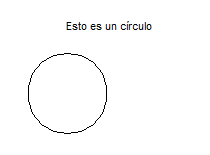


### Texto.

La función WRITE nos permite poner texto dentro de nuestros dibujos. Se le pueden pasar hasta cuatro parámetros. Estos parámetros son:

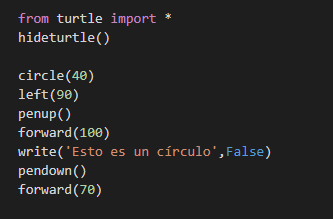
* TEXTO: cadena de texto a mostrar. Comienza a escribirse donde este el cursor, salvo que le indiquemos otra cosa.

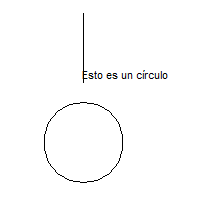




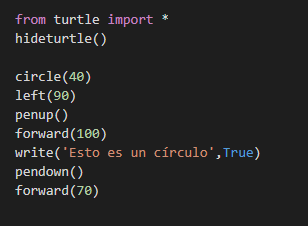
* DESPLAZAMIENTO: este parámetro puede tomar los valores TRUE o FALSE y lo que le indicamos a Python es si desplaza el cursor a la esquina inferior derecha del texto o no antes de seguir dibujando.

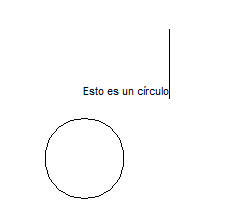
Probamos FALSE:



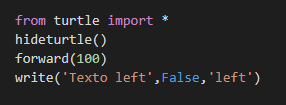


Probamos True:



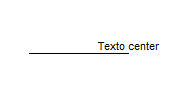


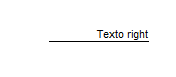
* ALINEACIÓN: indicamos cómo será la alineación del texto respecto a la posición actual del cursor. Puede ser LEFT, CENTER o RIGHT.



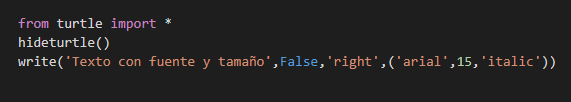


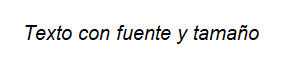
Probamos ahora que sucede si usamos CENTER o RIGHT:





* FUENTE: es una tupla de hasta tres elementos que indica tipo de letra, tamaño y tipo.



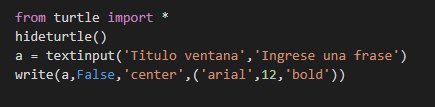


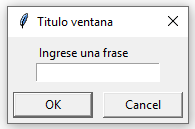
### Ingreso de datos por teclado.

Podemos pedirle al usuario que ingrese valores alfanuméricos por teclado mediante la función TEXTINPUT.

Los parámetros son TITULO (es el título de la ventana emergente) y MENSAJE (texto que se muestra en la ventana para el usuario).

El valor se deberá asignar a una variable para poder utilizarla.





De similar manera, podemos pedirle al usuario valores numéricos mediante la función NUMINPUT con algunos parámetros adicionales (título, mensaje, predeterminado, mínimo, máximo).

