Gráficos: el módulo turtle (1)

En esta lección se trata la creación de gráficos usando el módulo turtle.

El módulo turtle

En 1967 Wally Feurzeig y Seymour Papert crearon Logo, un lenguaje de programación con fines educativos. Ese lenguaje incluía las llamadas "gráficas tortuga".

La "tortuga" de Logo es un cursor al que se le pueden dar órdenes de movimiento (avance, retroceso o giro) y que puede ir dejando un rastro sobre la pantalla. Moviendo adecuadamente la tortuga se pueden conseguir dibujar todo tipo de figuras.

Python incluye un módulo llamado "turtle" que permite crear gráficos de tortuga.

En esta lección se explica cómo utilizar el módulo turtle para crear dibujos sencillos, pero sin utilizar la tortuga.

Para utilizar el módulo turtle sólo hace falta importarlo:

- Si se va a escribir código no orientado a objetos:

```
from turtle import *
```

- Si se va a escribir código orientado a objetos:

```
import turtle
```

La ventana de dibujo: setup(), title() y exitonclick()

El módulo turtle dibuja en una ventana distinta a la ventana de IDLE. Esta ventana de dibujo se crea al ejecutar un programa y se mantiene al acabar la ejecución del programa, pero se destruye al volver a ejecutar el programa.

La función **setup(ancho, alto, posicionX, posicionY)** permite definir el tamaño y la posición inicial de la ventana. Los cuatro argumentos de la función son (en píxeles):

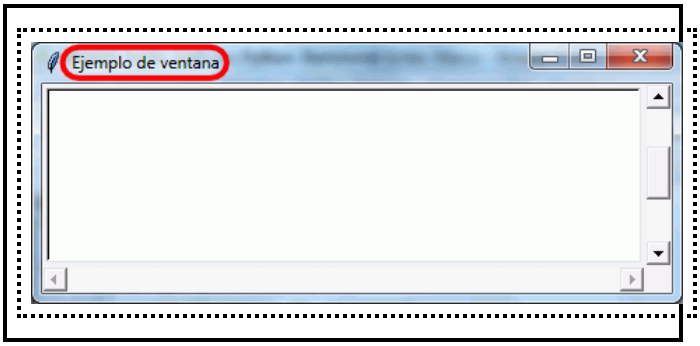
- ancho: ancho de la ventana.
- alto: alto de la ventana.
- posicionX: posición horizontal de la ventana. Los valores positivos se miden desde el borde izquierdo de la pantalla, los negativos desde el borde derecho de la pantalla.
- posicionY: posición vertical de la ventana. Los valores positivos se miden desde el borde superior de la pantalla, los negativos desde el borde inferior de la pantalla.

La función **exitonclick()** mantiene abierta la ventana hasta que se hace click en la ventana. Si el programa se ejecuta en IDLE, esta función no es necesaria, pero si el programa se ejecuta desde la línea de comandos la función es necesaria porque si no la ventana se cierra automáticamente una vez ejecutado el programa.

```
from turtle import *

setup(640, 480, 0, 0)

exitonclick()
```

La función **title()** permite definir el título de la ventana, que se muestra en el borde superior de la ventana.

```
from turtle import *

setup(450, 150, 0, 0)
title("Ejemplo de ventana")

exitonclick()
```

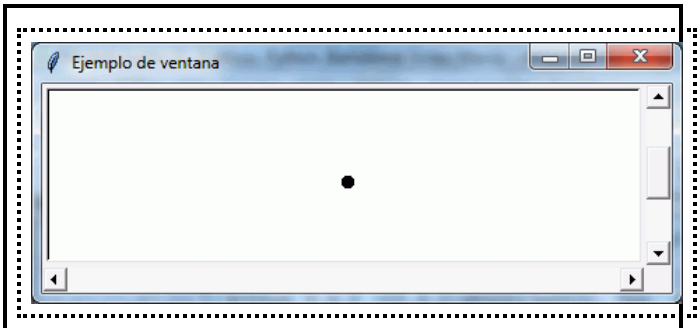


Si no se usa la función **setup()**, la ventana se crea en el centro de la pantalla (para que se cree la ventana tiene que aparecer alguna función del módulo turtle):

```
from turtle import *

title("Ejemplo de ventana")

exitonclick()
```

El tamaño de la ventana de dibujo se puede modificar a lo largo de un programa, sin que se pierda el dibujo realizado anteriormente.

```
from turtle import *

setup(250, 100, 0, 0)
title("Ejemplo de ventana")
hideturtle()
dot(10, 0, 0, 0)
setup(450, 150, 0, 0)

exitonclick()
```



El área de dibujo: screensize()

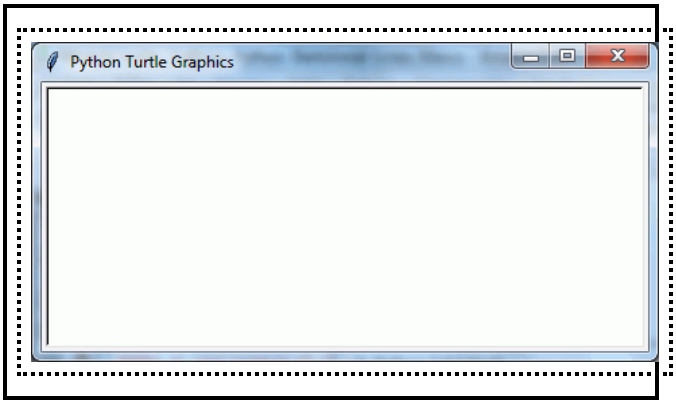
La ventana de dibujo contiene el área de dibujo, una superficie plana en la que se puede dibujar.

El área de dibujo tiene un tamaño inicial de 400 x 300 píxeles (en Windows), pero la función **screensize(ancho, alto)** permite definir el tamaño del área de dibujo, en píxeles.

El área de dibujo puede ser mayor o menor que la ventana de dibujo.

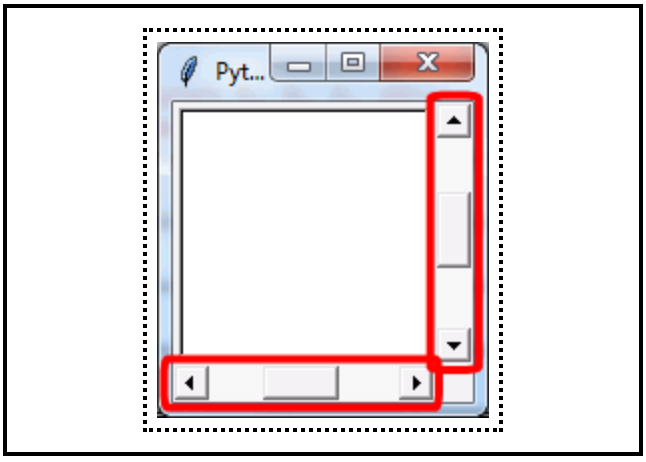
- Si la ventana es más grande que el área de dibujo, no se muestran barras de desplazamiento:

```
from turtle import *

setup(450, 150, 0, 0)
screensize(10, 10)

exitonclick()
```



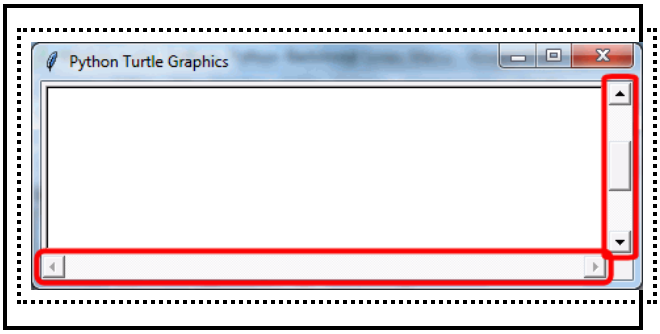
- Si la ventana es más pequeña que el área de dibujo, se muestran barras de desplazamiento en la dirección en que se requieran.
  - En el ejemplo siguiente, están activas las dos barras de desplazamiento:

```
from turtle import *

setup(150, 150, 0, 0)
screensize(300, 300)

exitonclick()
```



  - En el ejemplo siguiente, sólo se está activa la barra de desplazamiento vertical:

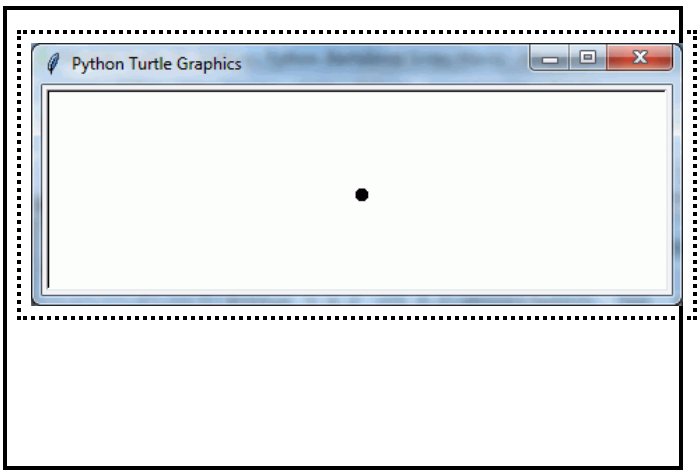
```
from turtle import *

setup(450, 150, 0, 0)
screensize(400, 300)

exitonclick()
```



En cualquier caso, el fondo del área de dibujo es del mismo color que el fondo de la ventana, por lo que no se distinguen los bordes del espacio de dibujo.

El tamaño del área de dibujo se puede modificar a lo largo de un programa, sin que se pierda el dibujo realizado anteriormente.

```
from turtle import *

setup(250, 100, 0, 0)
screensize(100, 100)
hideturtle()
dot(10, 0, 0, 0)
screensize(200, 100)

exitonclick()
```



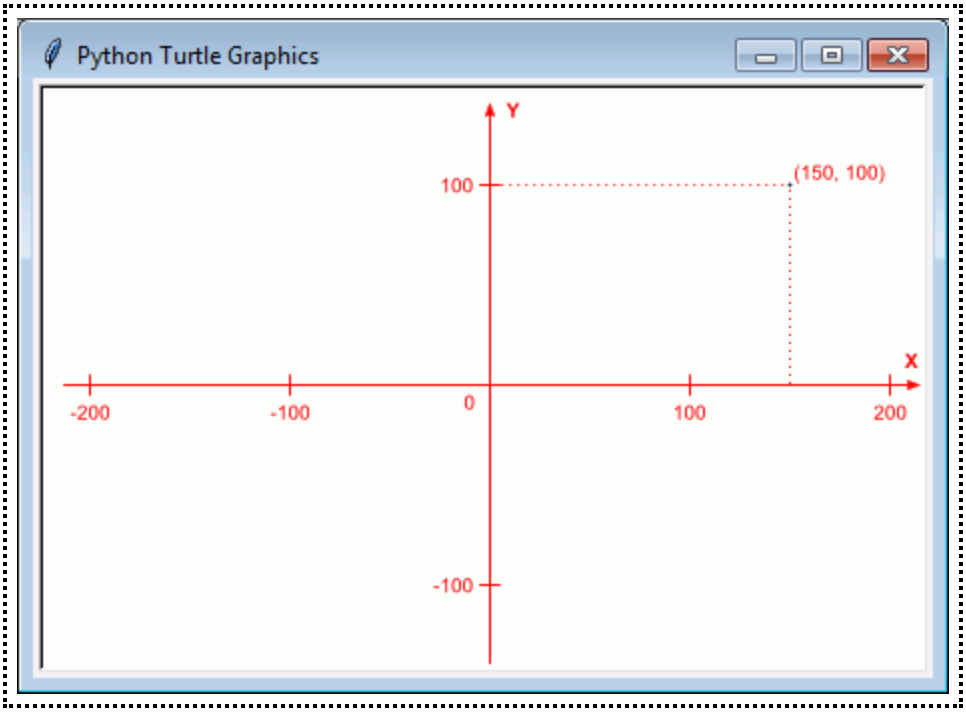
Podemos dibujar fuera del área de dibujo, pero las barras de desplazamiento corresponden al tamaño del área de dibujo, por lo que utilizando las barras de desplazamiento no podemos ir más allá del tamaño definido. Para ver más allá del área de dibujo podemos hacer más grande la ventana, pero si el dibujo está más allá de la pantalla, no habría manera de llegar a él (salvo hacer en el programa más grande el área de dibujo, ya que el área de dibujo puede ser mayor que la pantalla).

Dibujar en la ventana

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.

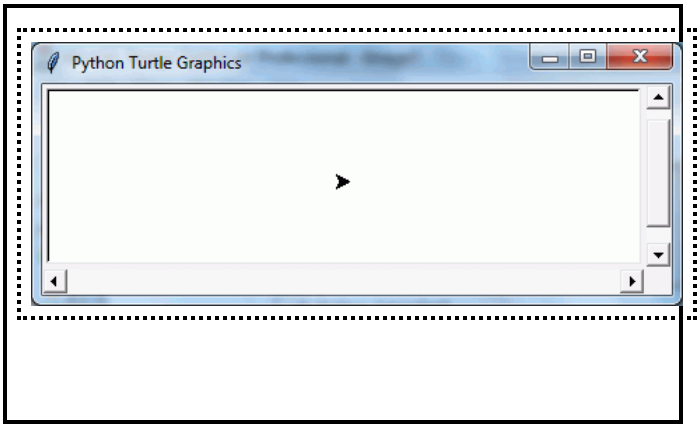
Las posiciones en el área de dibujo se localizan mediante coordenadas XY en el que cada píxel es una unidad y con origen en el centro de la ventana. La imagen siguiente muestra cuál sería la posición de los ejes de coordenadas y cuáles serían las coordenadas de un punto cualquiera.



La tortuga: showturtle(), hideturtle()

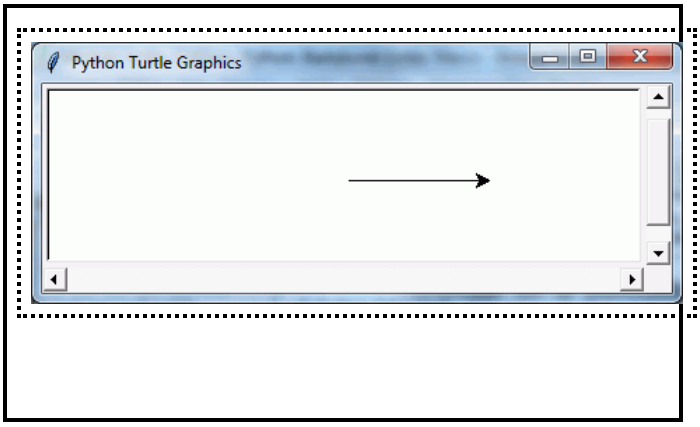
La función **showturtle()** muestra el cursor (la tortuga).

```
from turtle import *

setup(450, 150, 0, 0)
screensize(300, 150)

showturtle()

exitonclick()
```



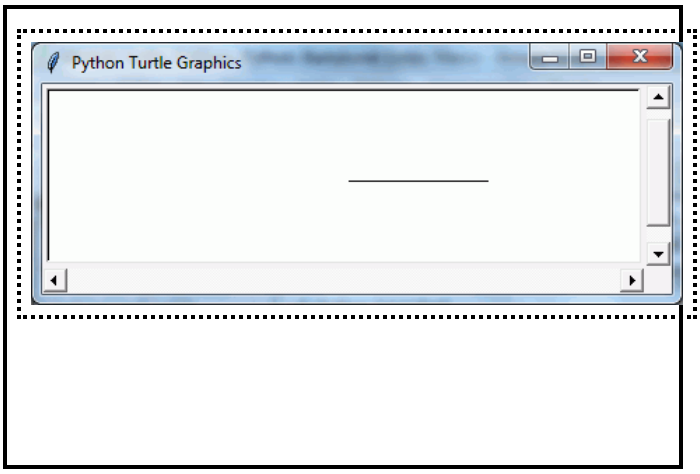
Cuando se hace un dibujo, la tortuga se muestra en la posición final del dibujo (la función **goto()** se explica más adelante):

```
from turtle import *

setup(450, 150, 0, 0)
screensize(300, 150)

goto(100, 0)

exitonclick()
```



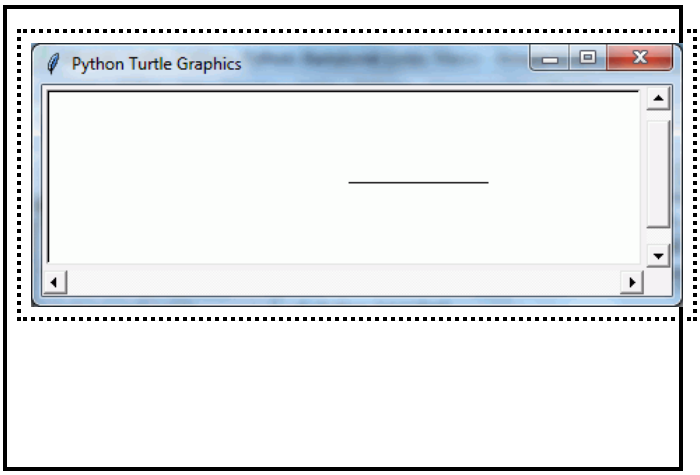
La función **hideturtle()** oculta el cursor (la tortuga).

```
from turtle import *

setup(450, 150, 0, 0)
screensize(300, 150)

goto(100, 0)
hideturtle()

exitonclick()
```



Si no se quiere mostrar la tortuga, es mejor ocultarla al principio del programa, ya que el programa se ejecutará más rápido.

```
from turtle import *

setup(450, 150, 0, 0)
screensize(300, 150)

hideturtle()
goto(100, 0)

exitonclick()
```



Dibujar líneas

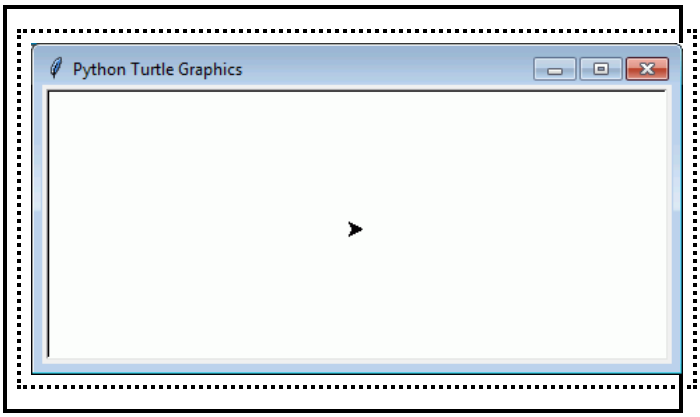
Mover el lápiz: goto(), setx() y sety()

Los píxeles del área de dibujo se pueden localizar mediante un sistema de coordenadas XY centrado en el centro del área de dibujo. Al crear la ventana, el cursor se sitúa en el centro de la ventana, de coordenadas (0, 0).

La función **goto(x, y)** permite desplazar el cursor a una posición determinada del área de dibujo. En el ejemplo siguiente el cursor se muestra en la posición (0, 0) (el centro de la ventana):

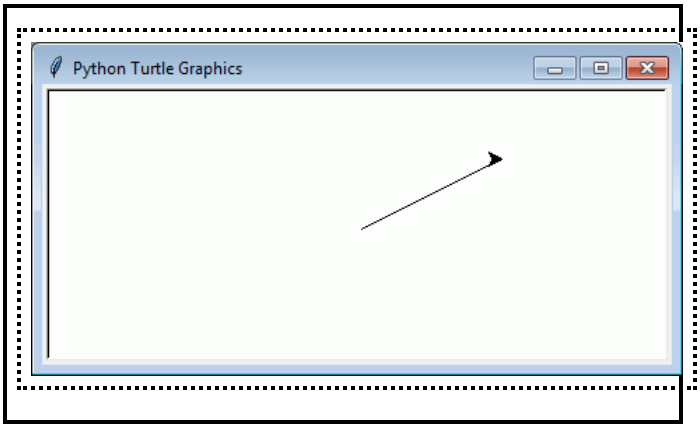
```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
```

```
goto(0, 0)

exitonclick()
```



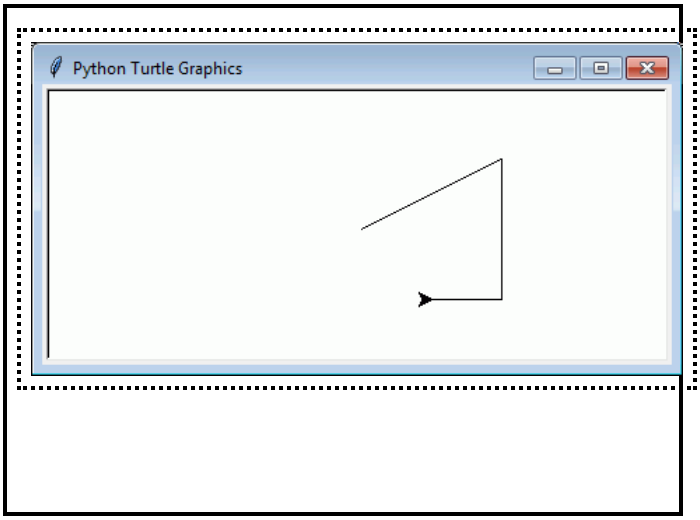
Si el primer goto() desplaza a un punto distinto, se dibuja la línea entre el punto inicial (0, 0) y el indicado:

```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)

goto(100, 50)

exitonclick()
```



Si se escriben varias instrucciones goto(), se van dibujando los segmentos uno a continuación del otro.

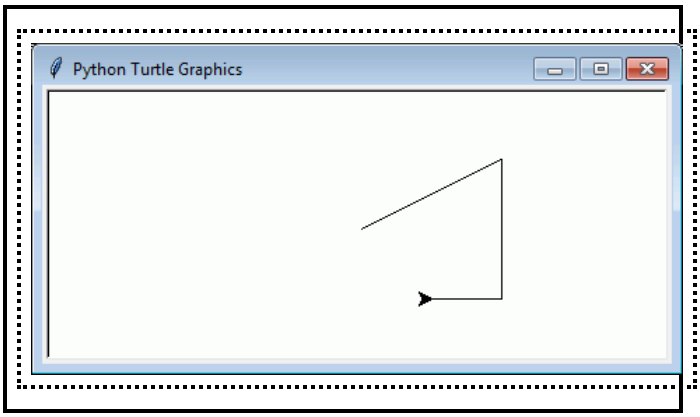
```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)

goto(100, 50)
goto(100, -50)
goto(50, -50)

exitonclick()
```



Las funciones **setx()** y **sety()** permiten desplazar el cursor a una posición determinada del área de dibujo. La función **setx()** modifica la abcisa X manteniendo la ordenada Y, mientras que la función **sety()** modifica la ordenada Y manteniendo la abcisa X.

```
from turtle import *
```

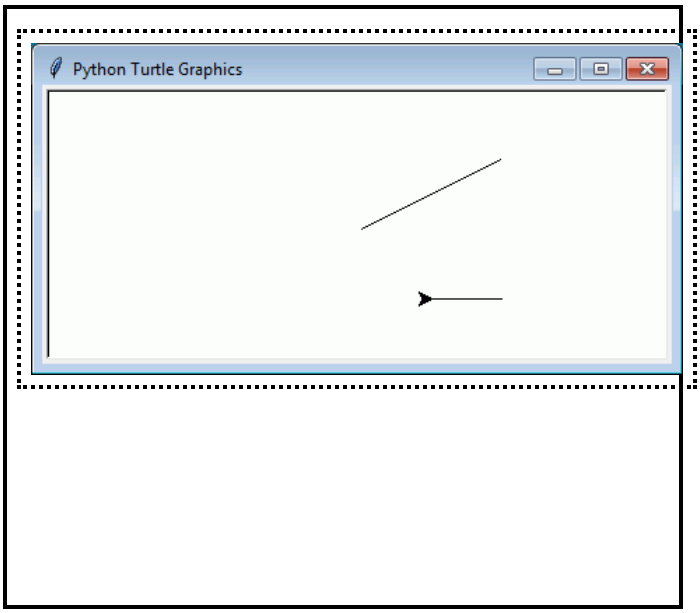
```
setup(450, 200, 0, 0)
screensize(300, 150)

goto(100, 50)
sety(-50)
setx(50)

exitonclick()
```



Levantar y bajar el lápiz: penup() y pendown()

Las funciones **pendown()** y **penup()** son equivalentes a bajar y levantar el lápiz del papel. Una vez levantado el lápiz del papel, al desplazar el lápiz ya no se dibujan segmentos. Al volver a bajar el lápiz, al desplazar el lápiz vuelven a dibujarse los fragmentos.

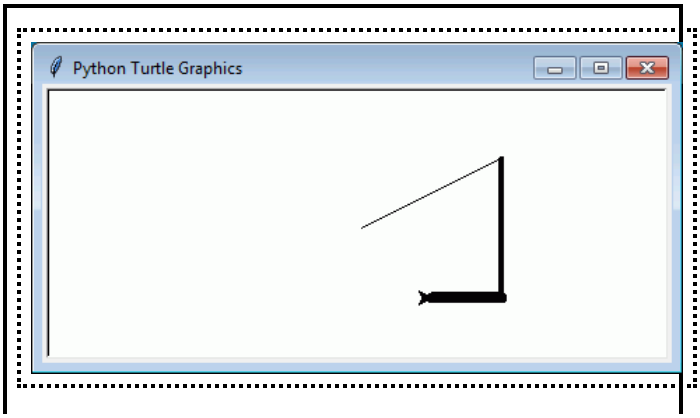
```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)

goto(100, 50)
penup()
goto(100, -50)
pendown()
goto(50, -50)

exitonclick()
```



Grosor del trazo: pensize()

La función **pensize(grosor)** permite modificar el grosor del trazo.

```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)

goto(100, 50)
pensize(4)
goto(100, -50)
```

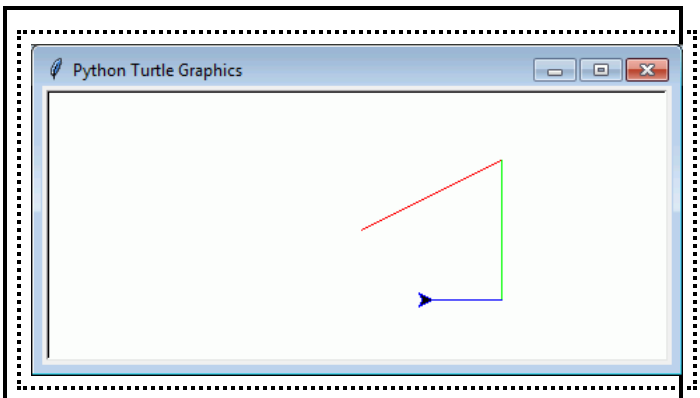


```
pensize(8)
goto(50, -50)

exitonclick()
```

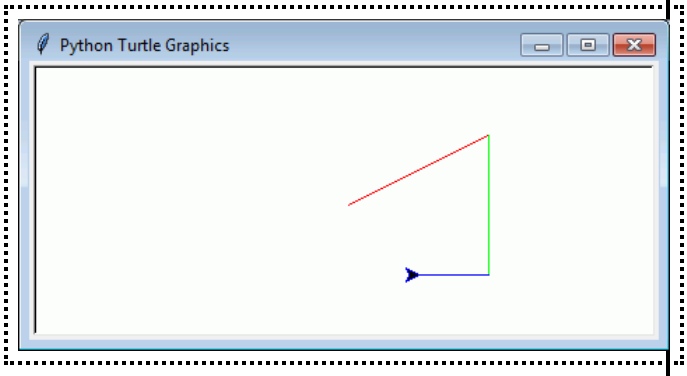
Color del trazo: pencolor() y colormode()

La función **pencolor(rojo, azul, verde)** permite modificar el color del trazo. El color se da como combinación de rojo, azul y verde. Los valores de color se pueden dar como valores enteros entre 0 y 255 o como valores decimales entre 0 y 1. Para elegir entre un modo y otro de indicar los colores hay que utilizar la función **colormode(1)** o **colormode(255)**.

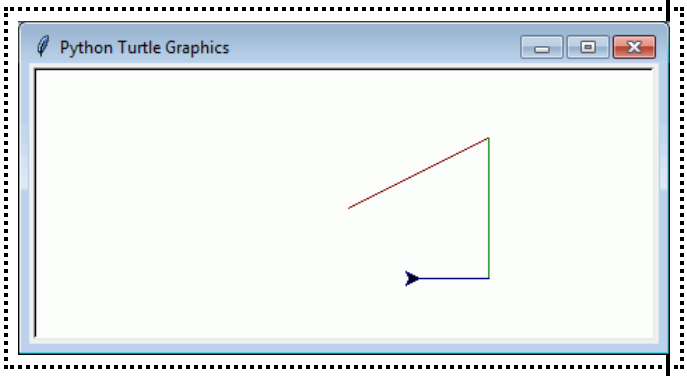
```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
colormode(255)

pencolor(255, 0, 0)
goto(100, 50)
pencolor(0, 255, 0)
goto(100, -50)
pencolor(0, 0, 255)
goto(50, -50)

exitonclick()
```



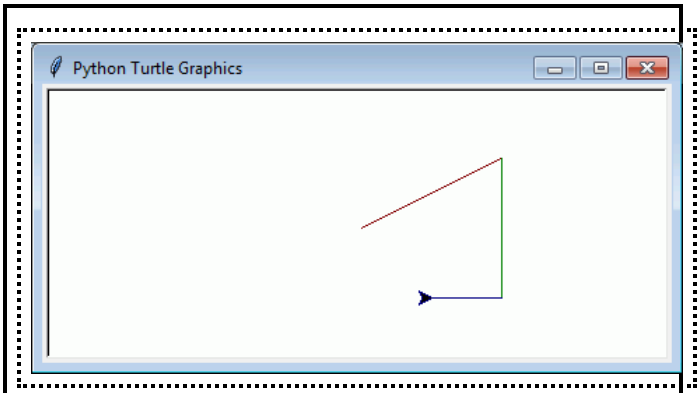

```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
colormode(1)

pencolor(1, 0, 0)
goto(100, 50)
pencolor(0, 1, 0)
goto(100, -50)
pencolor(0, 0, 1)
goto(50, -50)

exitonclick()
```



```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
colormode(255)

pencolor(128, 0, 0)
goto(100, 50)
pencolor(0, 128, 0)
goto(100, -50)
pencolor(0, 0, 128)
goto(50, -50)

exitonclick()
```



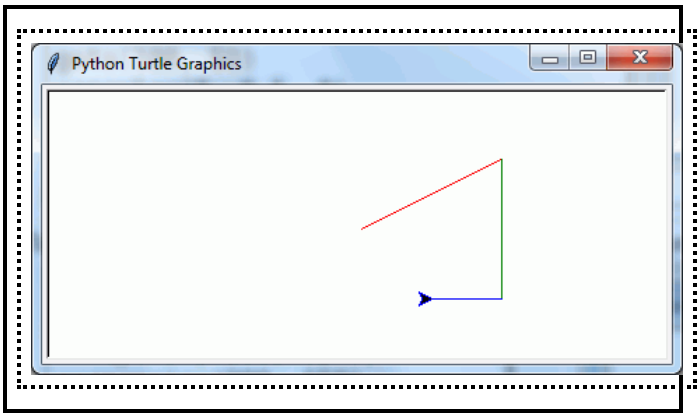

```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
colormode(1)

pencolor(0.5, 0, 0)
goto(100, 50)
pencolor(0, 0.5, 0)
goto(100, -50)
pencolor(0, 0, 0.5)
goto(50, -50)

exitonclick()
```



También se pueden utilizar los nombres de colores de Tk, que incluyen entre otros los nombres de colores SVG, en cuyo caso no hace falta utilizar la función **colormode()**.

```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)

pencolor("red")
```

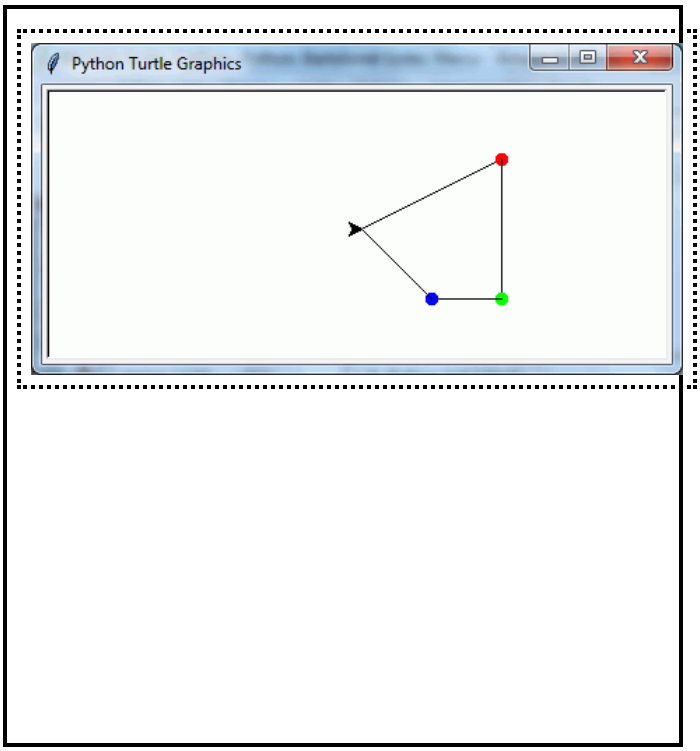
```
goto(100, 50)
pencolor("green")
goto(100, -50)
pencolor("blue")
goto(50, -50)

exitonclick()
```



Dibujar puntos: dot()

La función **dot(grosor, color)** permite dibujar un punto del grosor y color indicado en el punto en el que se encuentra el cursor. El grosor se indica en píxeles y el color se expresa como en la función **pencolor()** vista en el apartado anterior (entre paréntesis, como tupla, o sin paréntesis).

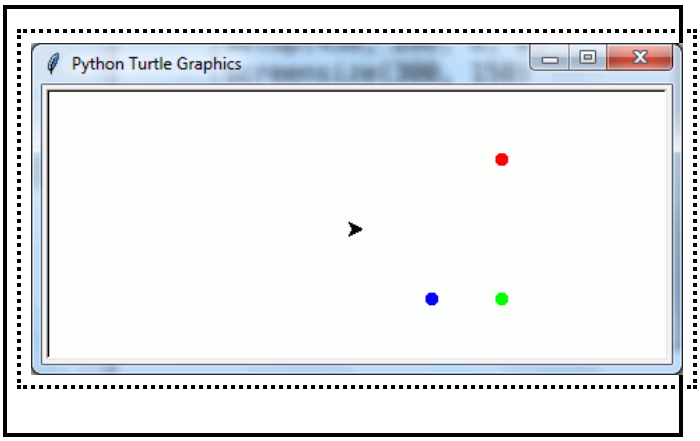
```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
colormode(255)

goto(100, 50)
dot(10, 255, 0, 0)
goto(100, -50)
dot(10, 0, 255, 0)
goto(50, -50)
dot(10, 0, 0, 255)
goto(0,0)

exitonclick()
```



Si queremos dibujar únicamente puntos, basta con levantar el lápiz (no es necesario bajarlo para dibujar puntos).

```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
colormode(255)

penup()
```

```
goto(100, 50)
dot(10, 255, 0, 0)
goto(100, -50)
dot(10, 0, 255, 0)
goto(50, -50)
dot(10, 0, 0, 255)
goto(0,0)

exitonclick()
```



Rellenar figuras

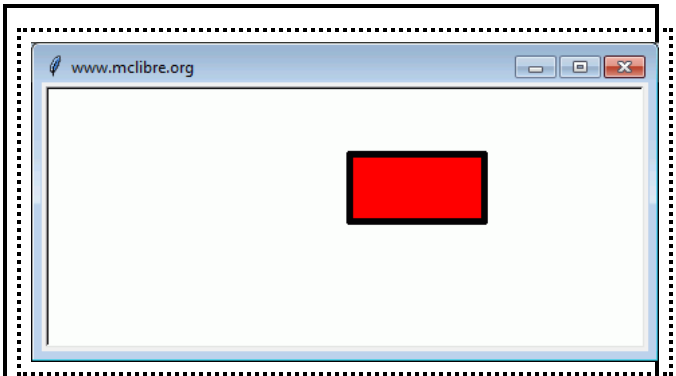
La función **begin_fill()** indica a Python que las figuras que se dibujen a partir de ese momento se deben rellenar. La función **end_fill()** indica a Python que las figuras deben dejar de rellenarse.

La función **fillcollor(color)** permite establecer el color de relleno, de la misma manera que la función pencolor(). El color se puede definir tanto como combinación de rojo, verde y azul, como con nombre de colores de Tk.

Si no se establece un color de relleno distinto, el color de relleno predeterminado es el negro.

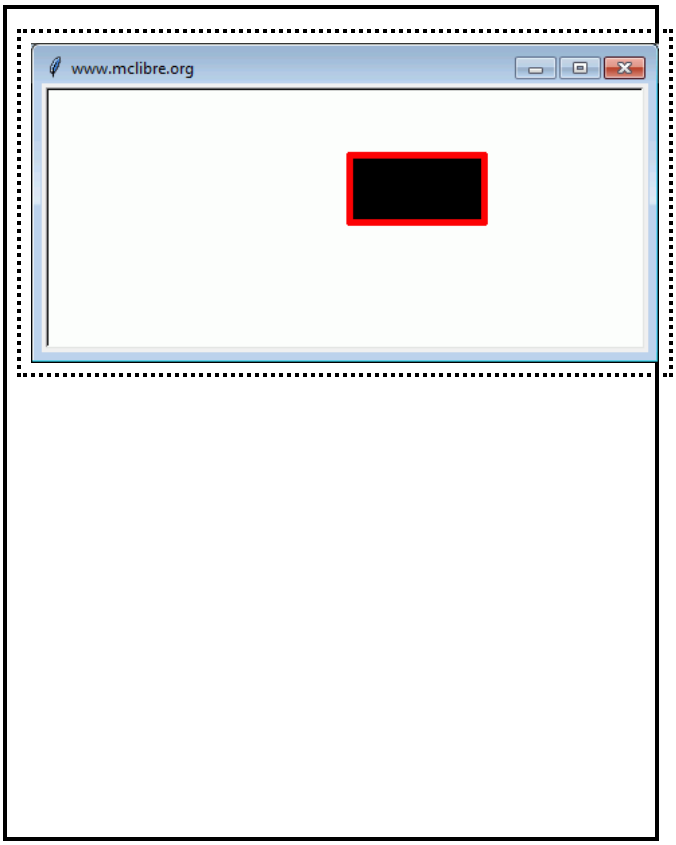
- Para rellenar una figura, se debe llamar a la función begin_fill(), dibujar la figura y llamar a la función end_fill().

```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
title("www.mclibre.org")
hideturtle()

pensize(5)
fillcolor("red")
begin_fill()
goto(100, 0)
goto(100, 50)
goto(0, 50)
goto(0, 0)
end_fill()

exitonclick()
```



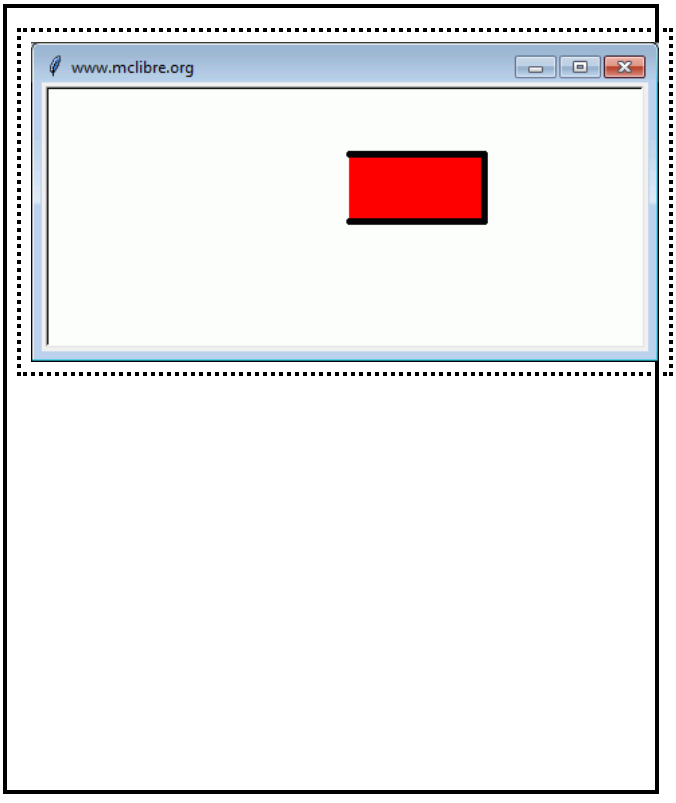
- Si no se establece un color de relleno distinto, el color de relleno predeterminado es el negro.

```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
title("www.mclibre.org")
hideturtle()

pencolor("red")
pensize(5)
begin_fill()
goto(100, 0)
goto(100, 50)
goto(0, 50)
goto(0, 0)
end_fill()

exitonclick()
```



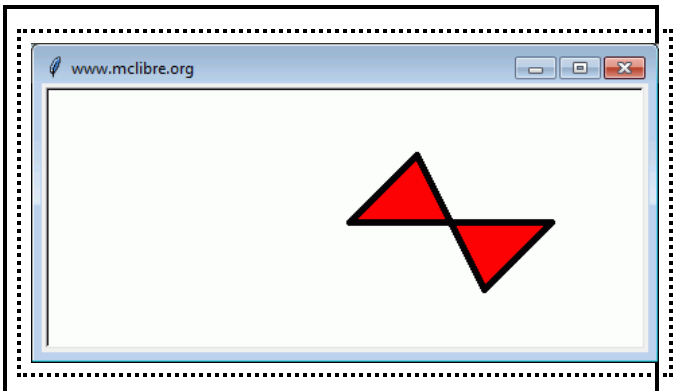
- Aunque realmente no es necesario dibujar la figura completa ya que Python rellena la figura aunque no se cierre la figura (es como si Python uniera el último punto de la figura con el primero).

```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
title("www.mclibre.org")
hideturtle()

pensize(5)
fillcolor("red")
begin_fill()
goto(100, 0)
goto(100, 50)
goto(0, 50)
end_fill()

exitonclick()
```



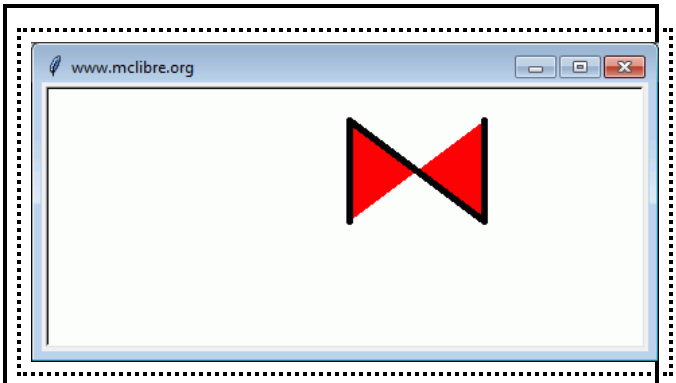
- Si las líneas de la figura se cruzan, Python rellena cada una de las partes cerradas.

```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
title("www.mclibre.org")
hideturtle()

pensize(5)
fillcolor("red")
begin_fill()
goto(50, 50)
goto(100, -50)
goto(150, 0)
goto(0, 0)
end_fill()

exitonclick()
```



- Si las líneas de la figura se cruzan, Python también rellena la figura aunque no se cierre la figura (es como si Python uniera el último punto de la figura con el primero).

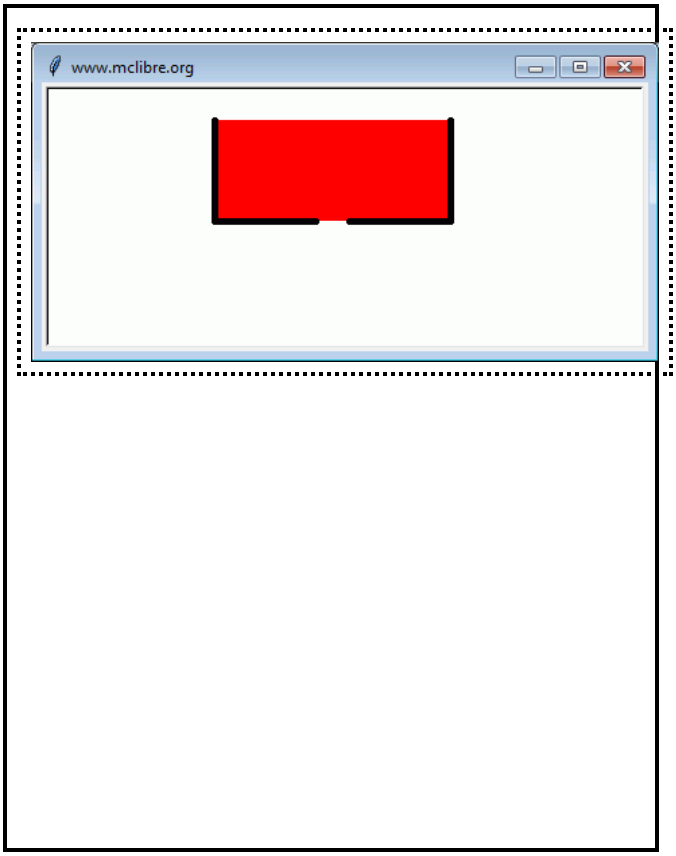
```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
title("www.mclibre.org")
hideturtle()

pensize(5)
fillcolor("red")
begin_fill()
goto(0, 75)
goto(100, 0)
goto(100, 75)
end_fill()

exitonclick()
```



- Cuando se levanta el lápiz, Python incluye el trayecto en la figura a rellenar ...

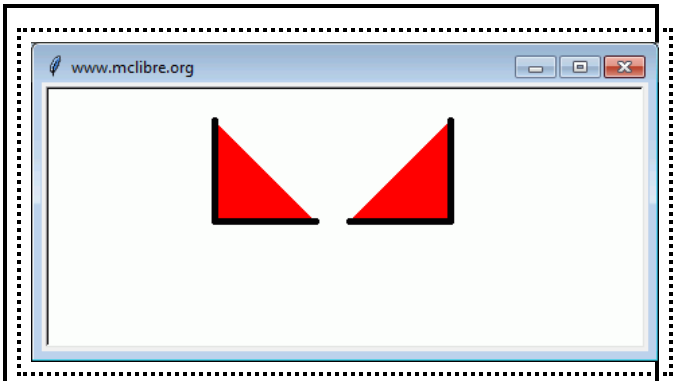
```
from turtle import *
```

```
setup(450, 200, 0, 0)
screensize(300, 150)
title("www.mclibre.org")
hideturtle()

pensize(5)
fillcolor("red")
begin_fill()
goto(75, 0)
goto(75, 75)
penup()
goto(-100, 75)
pendown()
goto(-100,0)
goto(-25, 0)
end_fill()

exitonclick()
```



- ... por lo que si se quieren dibujar figuras separadas, es mejor rellenar cada una de ellas de forma independiente ...

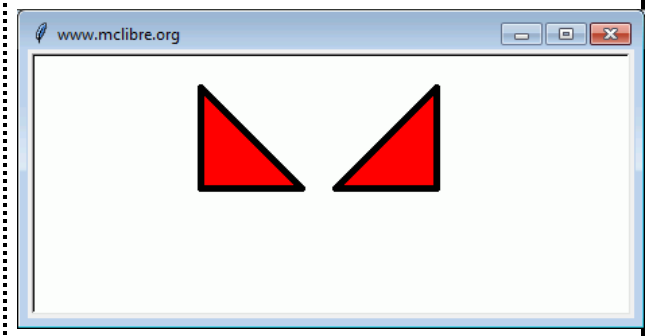
```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
title("www.mclibre.org")
hideturtle()

pensize(5)
fillcolor("red")
begin_fill()
goto(75, 0)
goto(75, 75)
end_fill()
penup()
goto(-100, 75)
pendown()
begin_fill()
goto(-100,0)
goto(-25, 0)
end_fill()
```



```
exitonclick()
```

- ... aunque si se trata de figuras cerradas a menudo no es necesario rellenarlas por separado (si los trayectos realizados con el lápiz levantado no cierran ninguna zona):

```
from turtle import *

setup(450, 200, 0, 0)
screensize(300, 150)
title("www.mclibre.org")
hideturtle()

pensize(5)
fillcolor("red")
begin_fill()
goto(75, 0)
goto(75, 75)
goto(0,0)
penup()
goto(-100, 75)
pendown()
goto(-100,0)
goto(-25, 0)
goto(-100, 75)
end_fill()

exitonclick()
```



Dibujar círculos: circle()

Por completar ...

Última modificación de esta página: 8 de marzo de 2023