Desconocido

## 9 Más gráficos de tortuga



Volvamos al módulo turtle que empezamos a utilizar en [el Capítulo 4 .](ch04.xhtml#ch04) En este capítulo, aprenderemos que las tortugas de Python pueden hacer mucho más que dibujar simples líneas negras. Puedes utilizarlas para dibujar formas geométricas más avanzadas, crear diferentes colores e incluso rellenar tus formas con color.

### Empezando por el cuadrado básico

Anteriormente hemos utilizado el módulo turtle para dibujar formas sencillas. Vamos a importar el módulo turtle y a crear el objeto Turtle :

>>> import turtle  
>>> t = turtle.Turtle()

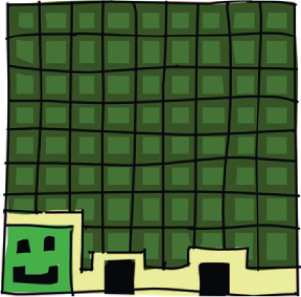
En el [Capítulo 4](ch04.xhtml#ch04) utilizamos el siguiente código para crear un cuadrado:

>>> t.forward(50)  
>>> t.left(90)  
>>> t.forward(50)  
>>> t.left(90)  
>>> t.forward(50)  
>>> t.left(90)  
>>> t.forward(50)  
>>> t.left(90)

En el [Capítulo](ch06.xhtml#ch06) 6 , aprendimos sobre los bucles for . Con nuestros nuevos conocimientos, podemos utilizar un bucle for para simplificar este código, de la siguiente manera:

>>> t.reset()  
>>> for x in range(1, 5):  
 t.forward(50)  
 t.left(90)

En la primera línea, le decimos al objeto Turtle que se reinicie. A continuación, iniciamos un bucle for que contará de 1 a 4 con range(1, 5) . Con las líneas siguientes, en cada ejecución del bucle, avanzamos 50 píxeles y giramos 90 grados a la izquierda. Como hemos utilizado un bucle for , este código es un poco más corto que el de la versión anterior: quitando la línea reset , hemos pasado de ocho líneas a tres.

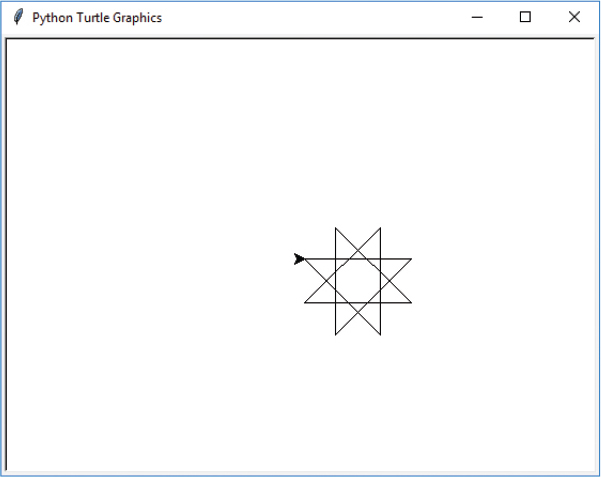


### Dibujar estrellas

Ahora, con unos sencillos cambios en nuestro bucle for , podemos crear algo aún más interesante. Introduce lo siguiente:

>>> t.reset()  
>>> for x in range(1, 9):  
 t.forward(100)  
 t.left(225)

Este código produce una estrella de ocho puntas, como en la [Figura 9-1 .](ch09.xhtml#ch09fig01)



Figura*9-1:* Estrella de ocho puntas

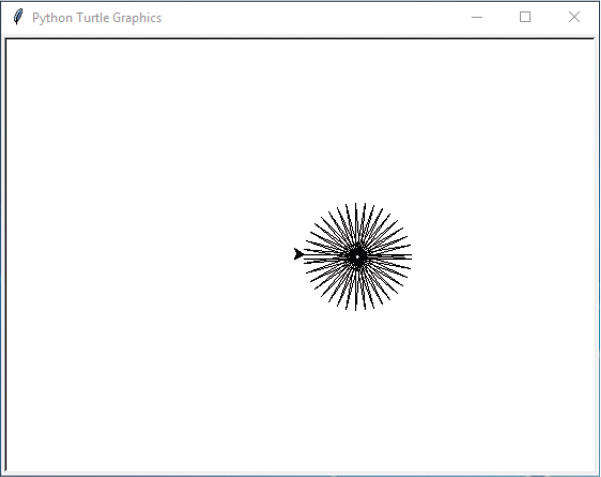
El código en sí es muy similar al que utilizamos para dibujar un cuadrado, con algunas excepciones:

* En lugar de hacer un bucle cuatro veces con range(1, 5) , hacemos un bucle ocho veces con range(1, 9) .
* En lugar de avanzar 50 píxeles, avanzamos 100 píxeles.
* En lugar de girar 90 grados, giramos 225 grados a la izquierda.

Desarrollemos un poco más nuestra estrella. Utilizando un ángulo de 175 grados y haciendo un bucle 37 veces, podemos hacer una estrella con aún más puntos. Introduce el código siguiente:

>>> t.reset()  
>>> for x in range(1, 38):  
 t.forward(100)  
 t.left(175)

Puedes ver el resultado de ejecutar este código en la [Figura 9-2 .](ch09.xhtml#ch09fig02)

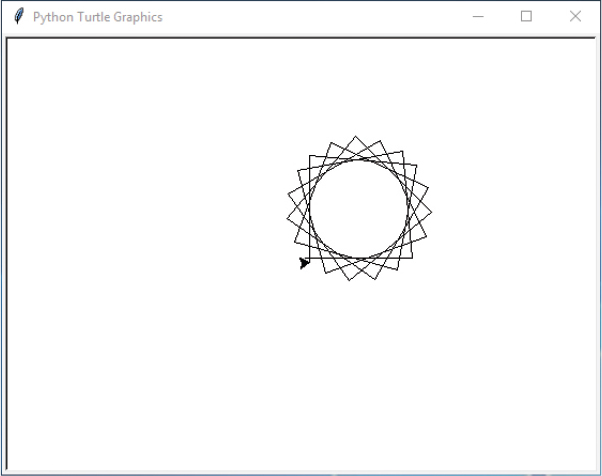


Figura*9-2: Estrella multipunto*

Ahora, prueba a introducir este código para producir una estrella en espiral:

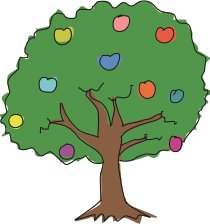
>>> t.reset()  
>>> for x in range(1, 20):  
 t.forward(100)  
 t.left(95)

Cambiando el grado del giro y reduciendo el número de bucles, la tortuga acaba dibujando un estilo de estrella bastante diferente, que puedes ver en la Figura 9-3 [.](ch09.xhtml#ch09fig03)



Figura*9-3: Estrella en espiral*

Utilizando un código similar, podemos crear una gran variedad de formas, desde un cuadrado básico hasta una estrella en espiral. Como puedes ver, el uso de los bucles for simplificó mucho el dibujo de estas formas. Sin los bucles de for , nuestro código habría requerido mucha escritura tediosa.

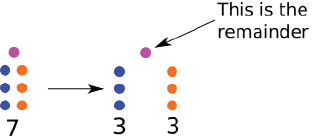


Probemos a utilizar una sentencia if para controlar cómo girará la tortuga y dibujar otra variación de estrella. En este ejemplo, queremos que la tortuga gire un ángulo la primera vez, y luego otro ángulo la siguiente:

t.reset()  
➊ for x in range(1, 19):  
 ➋ t.forward(100)  
 if x % 2 == 0:  
 t.left(175)  
 else:  
 t.left(225)

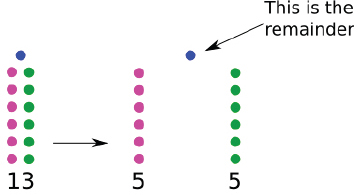
Aquí, creamos un bucle que se ejecutará 18 veces ➊ y le diremos a la tortuga que avance 100 píxeles ➋ . También hemos añadido la sentencia if , que comprueba si la variable x contiene un número par  utilizando un operador de módulo *.* El operador de módulo es el % del código x % 2 == 0 , que es una forma de decir: *"x* mod 2 es igual a 0".

El código x % 2 pregunta: "¿Cuál es la cantidad que sobra cuando divides el número de la variable x en dos partes iguales?". Por ejemplo, si dividiéramos 7 bolas en dos partes, obtendríamos dos grupos de 3 bolas (haciendo un total de 6 bolas), y el resto (la cantidad que sobra) sería 1 bola, como se muestra en [la Figura 9-4 .](ch09.xhtml#ch09fig04)



Figura*9-4: Dividir* 7 bolas en dos partes iguales

Si dividiéramos 13 bolas en dos partes, obtendríamos dos grupos de 6 bolas y nos quedaría 1 bola ( [Figura 9-5](ch09.xhtml#ch09fig05) ).

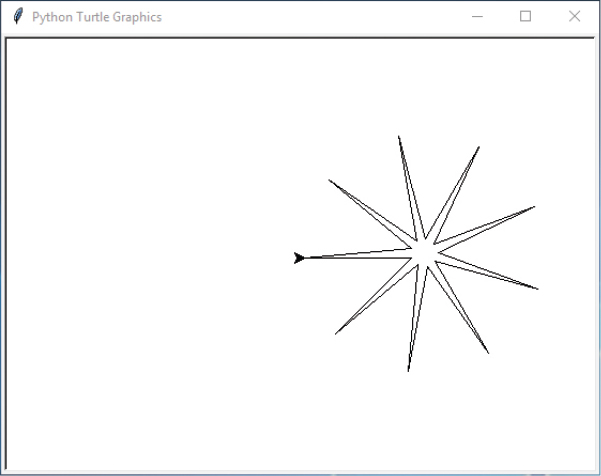


Figura*9-5:* Dividir 13 bolas en dos partes iguales

Cuando comprobamos si el resto es igual a cero después de dividir x entre 2, en realidad estamos preguntando si se puede dividir en dos partes sin resto. Este método es una buena forma de ver si un número de una variable es par, porque los números pares siempre se pueden dividir en dos partes iguales.

En la quinta línea de nuestro código, le decimos a la tortuga que gire 175 grados a la izquierda ( t.left(175) ) si el número de x es par ( if x % 2 == 0 ); en caso contrario ( else ), en la última línea, le decimos que gire 225 grados ( t.left(225) ).

[La Figura 9-6](ch09.xhtml#ch09fig06) muestra el resultado.



*Figura 9-6: Estrella de nueve puntas*

Si intentaste el reto de las cuatro espirales del capítulo anterior, puede que crearas cuatro objetos tortuga y copiaras el código cuatro veces con ligeras diferencias para cada tortuga, de modo que dibujaran la espiral en la dirección correcta. Con los bucles for y las sentencias if , podrías hacer lo mismo con un código mucho más sencillo.

### Dibujar un coche

La tortuga también puede cambiar de color y dibujar formas concretas. Para este ejemplo, dibujaremos un coche sencillo, aunque no de aspecto tonto.

Primero dibujaremos la carrocería del coche. En IDLE, selecciona **Archivo ▸ Nuevo Archivo** , e introduce el siguiente código en la ventana:

t.reset()  
t.color(1,0,0)  
t.begin\_fill()  
t.forward(100)  
t.left(90)  
t.forward(20)  
t.left(90)  
t.forward(20)  
t.right(90)  
t.forward(20)  
t.left(90)  
t.forward(60)  
t.left(90)  
t.forward(20)  
t.right(90)  
t.forward(20)  
t.left(90)  
t.forward(20)  
t.end\_fill()

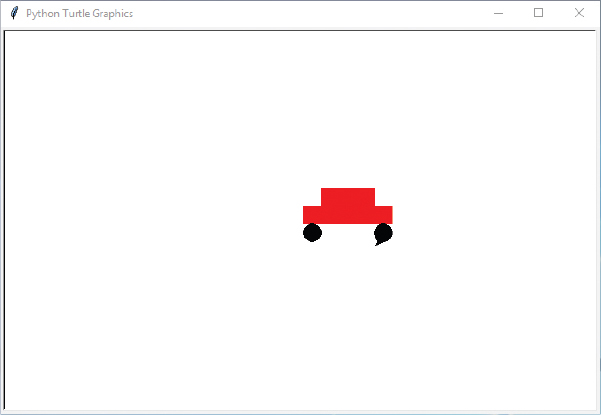
A continuación, dibujaremos la primera rueda:

t.color(0,0,0)  
t.up()  
t.forward(10)  
t.down()  
t.begin\_fill()  
t.circle(10)  
t.end\_fill()

Por último, dibujaremos la segunda rueda:

t.setheading(0)  
t.up()  
t.forward(90)  
t.right(90)  
t.forward(10)  
t.setheading(0)  
t.begin\_fill()  
t.down()  
t.circle(10)  
t.end\_fill()

Selecciona **Archivo ▸ Guardar como** y dale un nombre al archivo, por ejemplo *coche.py .* Selecciona **Ejecutar ▸ Ejecutar Módulo** para probar el código. Nuestro coche puede verse en la [Figura 9-7 .](ch09.xhtml#ch09fig07)



*Figura 9-7: Tortuga dibujando un* coche

Te habrás dado cuenta de que en este código se han colado unas cuantas funciones nuevas de turtle :

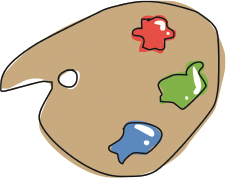
* color se utiliza para cambiar el color del lápiz.
* begin\_fill y end\_fill se utilizan para rellenar un área del lienzo con un color.
* circle dibuja un círculo de un tamaño determinado.
* setheading gira la tortuga para que mire en una dirección determinada.

Veamos cómo podemos utilizar estas funciones para añadir color a nuestros dibujos.

### Colorear cosas

La función color toma tres parámetros. El primero especifica la cantidad de rojo, el segundo la cantidad de verde y el tercero la cantidad de azul. Por ejemplo, para obtener el rojo brillante del coche, utilizamos color(1,0,0) , que le dice a la tortuga que utilice un rojo del 100%.

Esta receta de colores rojo, verde y azul se denomina *RGB* y es la forma en que se representan los colores en el monitor de tu ordenador. La mezcla de estos colores primarios produce otros colores, igual que cuando mezclas pintura azul y roja para hacer morado, o amarillo y rojo para hacer naranja. Los colores rojo, verde y azul se llaman *colores primarios* porque no puedes mezclar otros tonos para producirlos.



Aunque no estamos utilizando pintura para crear colores en un monitor de ordenador (estamos utilizando luz), puede ayudarte a entender el RGB imaginar que tienes tres botes de pintura: uno rojo, uno verde y uno azul. Cada bote está lleno, y a cada bote lleno le damos un valor de 1 (o 100%). A continuación, mezclamos toda la pintura roja y toda la pintura verde en una cuba para producir amarillo (es decir, 1 y 1 de cada, o el 100 por 100 de cada color).

Ahora volvamos al mundo del código. Para dibujar un círculo amarillo con la tortuga, utilizaríamos el 100 por cien de la pintura roja y verde, pero nada de azul, así

>>> t.color(1,1,0)  
 >>> t.begin\_fill()  
➊ >>> t.circle(50)  
 >>> t.end\_fill()

El 1,1,0 de la primera línea representa el 100 por 100 de rojo, el 100 por 100 de verde y el 0 por 100 de azul. En la línea siguiente, le decimos a la tortuga que rellene las formas que dibuje con este color RGB, y luego le decimos que dibuje un círculo ➊ . La última línea le dice a la tortuga que rellene el círculo con el color RGB.

### Una función para dibujar un círculo relleno

Para que sea más fácil experimentar con los colores, vamos a crear una función a partir del código que hemos utilizado para dibujar un círculo relleno:

>>> def mycircle(red, green, blue):  
 t.color(red, green, blue)  
 t.begin\_fill()  
 t.circle(50)  
 t.end\_fill()

Podemos dibujar un círculo verde brillante utilizando sólo la pintura verde, así

>>> mycircle(0, 1, 0)

O podemos dibujar un círculo verde más oscuro utilizando sólo la mitad de la pintura verde ( 0.5 ):

>>> mycircle(0, 0.5, 0)

Para jugar con los colores RGB de tu pantalla, prueba a dibujar un círculo primero con todo el rojo, luego con la mitad del rojo ( 1 y 0.5 ), y con todo el azul, luego con la mitad del azul, así:

>>> mycircle(1, 0, 0)  
>>> mycircle(0.5, 0, 0)  
>>> mycircle(0, 0, 1)  
>>> mycircle(0, 0, 0.5)

**NOTA**

*Si tu lienzo empieza a estar desordenado, utiliza t.reset() para borrar los dibujos antiguos. Recuerda también que puedes mover la tortuga sin dibujar líneas utilizando t.up() para levantar el lápiz y t.down() para volver a bajarlo.*

Varias combinaciones de rojo, verde y azul producirán una enorme variedad de colores, como el dorado:

>>> mycircle(0.9, 0.75, 0)

O rosa claro:

>>> mycircle(1, 0.7, 0.75)

Y aquí tienes dos tonos diferentes de naranja:

>>> mycircle(1, 0.5, 0)  
>>> mycircle(0.9, 0.5, 0.15)

¡Intenta mezclar tú mismo algunos colores!

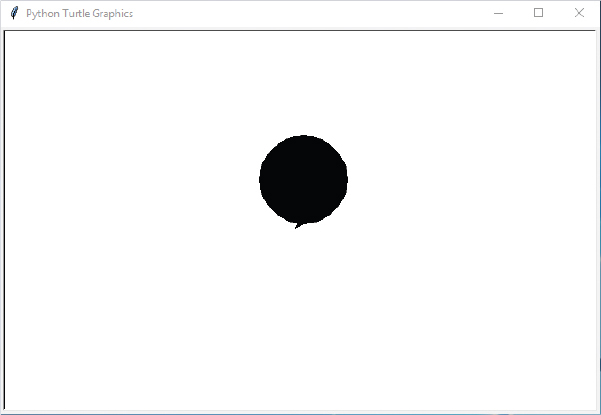
### Crear blanco y negro puros

¿Qué ocurre cuando apagas todas las luces por la noche? Todo se vuelve negro. Lo mismo ocurre con los colores en un ordenador. Sin luz no hay color, así que un círculo con 0 para todos los colores primarios crea el negro:



>>> mycircle(0, 0, 0)

[La Figura 9-8](ch09.xhtml#ch09fig08) muestra el resultado.



*Figura 9-8: Círculo negro*

Si utilizas el 100% de los tres colores, obtendrás el blanco. Introduce lo siguiente para borrar tu círculo negro:

>>> mycircle(1, 1, 1)

### Una función para dibujar cuadrados

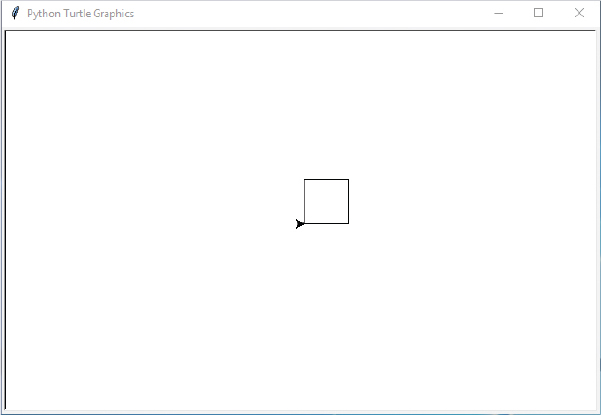
Ahora haremos algunos experimentos más con formas. Utilizaremos la función dibujar cuadrado del principio del capítulo y le pasaremos el tamaño del cuadrado como parámetro:

>>> def mysquare(size):  
 for x in range(1, 5):  
 t.forward(size)  
 t.left(90)

Prueba tu función llamándola con el tamaño 50, así:

>>> mysquare(50)

Esto produce el cuadrado pequeño de [la Figura 9-9 .](ch09.xhtml#ch09fig09)

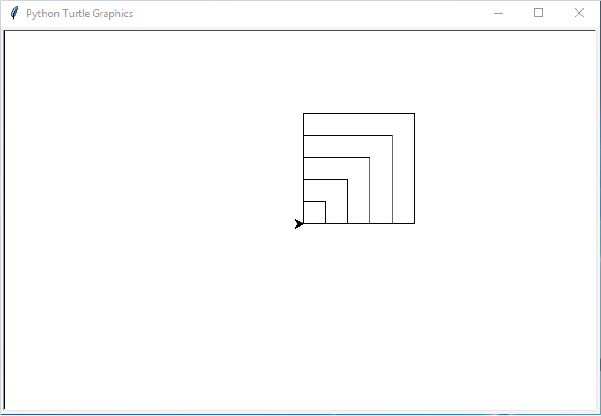


*Figura 9-9: Tortuga dibujando un* cuadrado pequeño

Ahora vamos a probar nuestra función con distintos tamaños. El código siguiente crea cinco cuadrados consecutivos de tamaño 25, 50, 75, 100 y 125:

>>> t.reset()  
>>> mysquare(25)  
>>> mysquare(50)  
>>> mysquare(75)  
>>> mysquare(100)  
>>> mysquare(125)

Estos cuadrados deberían tener el aspecto de [la Figura 9-10 .](ch09.xhtml#ch09fig10)



*Figura 9-10: Tortuga dibujando múltiples* cuadrados

### Dibujar cuadrados rellenos

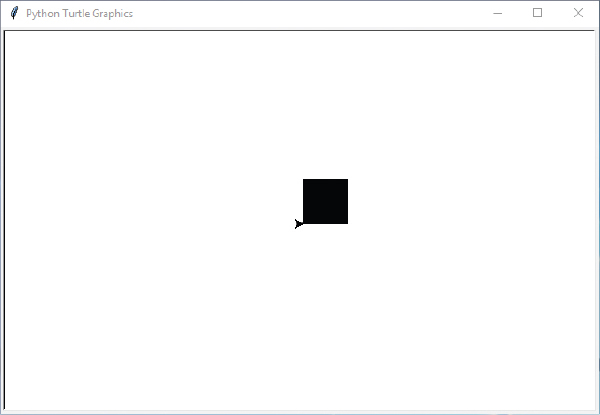
Para dibujar un cuadrado relleno, tenemos que reiniciar el lienzo, empezar a rellenarlo y volver a llamar a la función cuadrado, con este código:

>>> t.reset()  
>>> t.begin\_fill()  
>>> mysquare(50)

Deberías ver un cuadrado vacío hasta que termines de rellenar:

>>> t.end\_fill()

Tu cuadrado debería parecerse a [la Figura 9-11 .](ch09.xhtml#ch09fig11)



*Figura 9-11: Tortuga dibujando un cuadrado relleno*

Cambiemos esta función para que podamos dibujar un cuadrado relleno o sin relleno. Para ello, necesitamos otro parámetro y un código algo más complicado:

>>> def mysquare(size, filled):  
 if filled == True:  
 t.begin\_fill()  
 for x in range(1, 5):  
 t.forward(size)  
 t.left(90)  
 if filled == True:  
 t.end\_fill()

En la primera línea, cambiamos la definición de nuestra función para que tome dos parámetros: size y filled . A continuación, comprobamos si el valor de relleno es True con if filled == True . Si lo está, llamamos a begin\_fill para decirle a la tortuga que rellene la forma que hemos dibujado. A continuación, hacemos un bucle cuatro veces ( for x in range(1, 5) ) para dibujar los cuatro lados del cuadrado (avanzando hacia delante y hacia la izquierda) antes de comprobar de nuevo si filled es True . Si lo está, desactivamos el relleno con t.end\_fill , y la tortuga rellena el cuadrado de color.

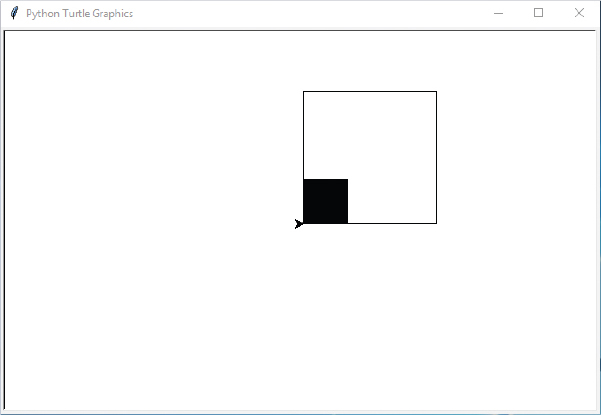
Ahora podemos dibujar un cuadrado relleno con esta línea:

>>> mysquare(50, True)

O podemos crear un cuadrado sin relleno con esta línea:

>>> mysquare(150, False)

Después de estas dos llamadas a la función mysquare , obtenemos [la Figura 9-12](ch09.xhtml#ch09fig12) , que se parece un poco a un ojo cuadrado.



*Figura 9-12: Tortuga dibujando un* ojo cuadrado

Pero no tiene sentido detenerse aquí. Puedes dibujar todo tipo de formas y llenarlas de color.

### Dibujar estrellas rellenas

Para nuestro ejemplo final, añadiremos color a la estrella que hemos dibujado antes. El código original tenía este aspecto

for x in range(1, 19):  
 t.forward(100)  
 if x % 2 == 0:  
 t.left(175)  
 else:  
 t.left(225)

Ahora haremos una función mystar . Utilizaremos las sentencias if de la función mysquare y añadiremos el parámetro size :

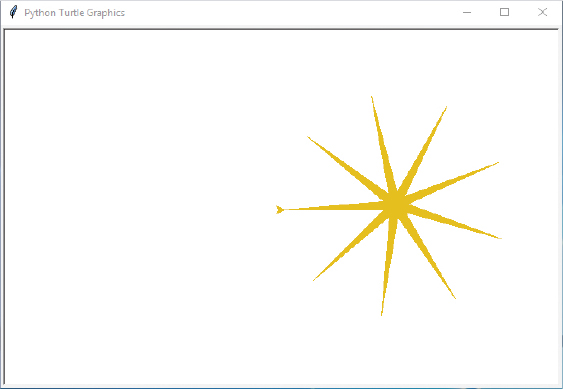
>>> def mystar(size, filled):  
 if filled == True:  
 t.begin\_fill()  
 for x in range(1, 19):  
 t.forward(size)  
 if x % 2 == 0:  
 t.left(175)  
 else:  
 t.left(225)  
 if filled == True:  
 t.end\_fill()

En las dos primeras líneas de esta función, comprobamos si filled es True ; si lo es, comenzamos el llenado. Volvemos a comprobarlo en las dos últimas líneas, y si relleno es True , dejamos de rellenar. Además, como en la función mysquare , pasamos el tamaño de la estrella en el parámetro size , y utilizamos ese valor cuando llamamos a t.forward .

Ahora pongamos el color en dorado (90 por ciento rojo, 75 por ciento verde y 0 por ciento azul), y volvamos a llamar a la función:

>>> t.color(0.9, 0.75, 0)  
>>> mystar(120, True)

La tortuga dibujará la estrella rellena de la [Figura 9-13 .](ch09.xhtml#ch09fig13)

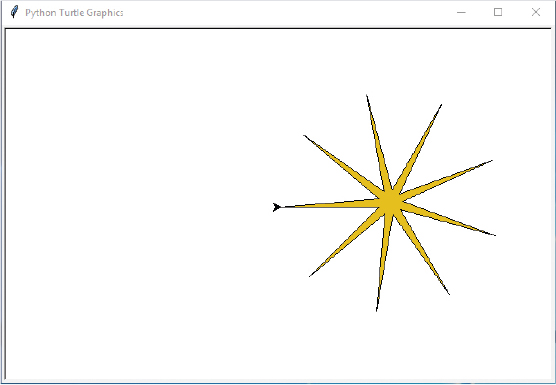


*Figura 9-13: Dibujo de una estrella dorada*

Para añadir un contorno a la estrella, cambia el color a negro y vuelve a dibujar la estrella sin relleno:

>>> t.color(0,0,0)  
>>> mystar(120, False)

Ahora la estrella es dorada con un contorno negro, como en [la Figura 9-14 .](ch09.xhtml#ch09fig14)



*Figura 9-14: Dibujo de una* estrella dorada con contorno

### Lo que has aprendido

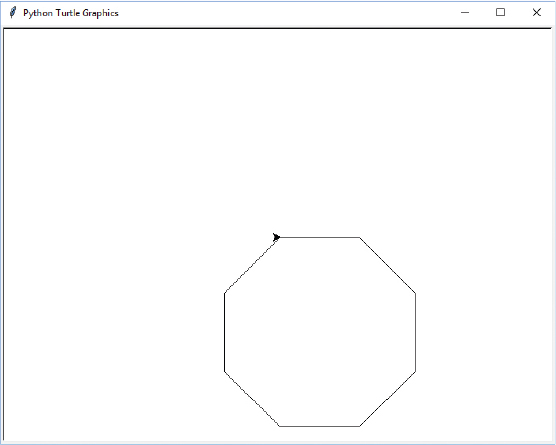
En este capítulo has aprendido a utilizar el módulo turtle para dibujar formas geométricas, utilizando los bucles for y las sentencias if para controlar lo que hace la tortuga en la pantalla. Cambiamos el color de la línea de la tortuga y rellenamos las formas que dibujaba. También reutilizamos el código de dibujo en algunas funciones para que sea más fácil dibujar formas con diferentes colores con una sola llamada a una función.

### Programación de puzzles

En los siguientes experimentos, dibujarás tus propias formas con la tortuga. Como siempre, puedes encontrar las soluciones en http://python-for-kids.com [*.*](http://python-for-kids.com)

#### #1: Dibujar un octógono

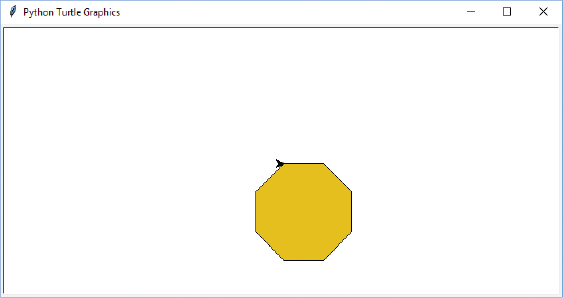
En este capítulo hemos dibujado estrellas, cuadrados y rectángulos. ¿Qué te parece crear una función para dibujar una forma de ocho lados como un octógono? (Pista: Prueba a girar la tortuga 45 grados.) Tu forma debería parecerse a la de [la Figura 9-15 .](ch09.xhtml#ch09fig15)



*Figura 9-15: Dibujar un* octógono

#### #2: Dibujar un octógono relleno

Ahora que tienes una función para dibujar un octógono, modifícala para dibujar un octógono relleno. Intenta dibujar un octógono con un contorno, como hicimos con la estrella. Debería tener un aspecto similar al de la [Figura 9-16 .](ch09.xhtml#ch09fig16)



*Figura 9-16: Dibujar un octógono relleno*

#### #3: Otra función para dibujar estrellas

Crea una función para dibujar una estrella que tome dos parámetros: el tamaño y el número de puntos. El principio de la función debería ser algo parecido a esto

def draw\_star(size, points):

#### #4: Cuatro espirales revisitadas

Toma el código que creaste para el puzzle de programación nº 4 del capítulo anterior (para crear cuatro espirales) y vuelve a dibujar las mismas espirales, sólo que esta vez, intenta utilizar bucles for y sentencias if para simplificar tu código.