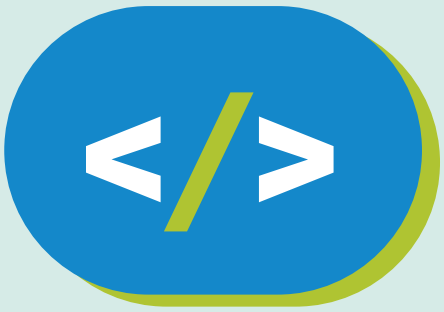


Kit de programación
Código Pi

Introducción a Python

APRENDER CONECTADOS

Ministerio de Educación,
Cultura, Ciencia y Tecnología
Presidencia de la Nación

Autoridades

Presidente de la Nación
Mauricio Macri

Jefe de Gabinete de Ministros
Marcos Peña

Ministro de Educación, Cultura, Ciencia y Tecnología
Alejandro Finocchiaro

Secretario de Gobierno de Cultura
Pablo Avelluto

Secretario de Gobierno de Ciencia, Tecnología e Innovación Productiva
Lino Barañao

Titular de la Unidad de Coordinación General del Ministerio de Educación, Cultura, Ciencia y Tecnología
Manuel Vidal

Secretaria de Innovación y Calidad Educativa
Mercedes Miguel

Subsecretario de Coordinación Administrativa
Javier Mezzamico

Directora Nacional de Innovación Educativa
María Florencia Ripani

ISBN en trámite

Índice

Introducción a Python

Lo que harás

En esta actividad tendrás tus primeros pasos dibujando formas, patrones y espirales usando el lenguaje de programación Python. Usarás un módulo llamado Turtle (tortuga). A lo largo del recorrido aprenderás cómo pensar en secuencia y el uso de bucles para repetir una secuencia. Este es un buen punto de partida para pasar de un lenguaje gráfico tipo Scratch a un lenguaje escrito como Python.

Lo que aprenderás

Al realizar patrones usando Python aprenderás:

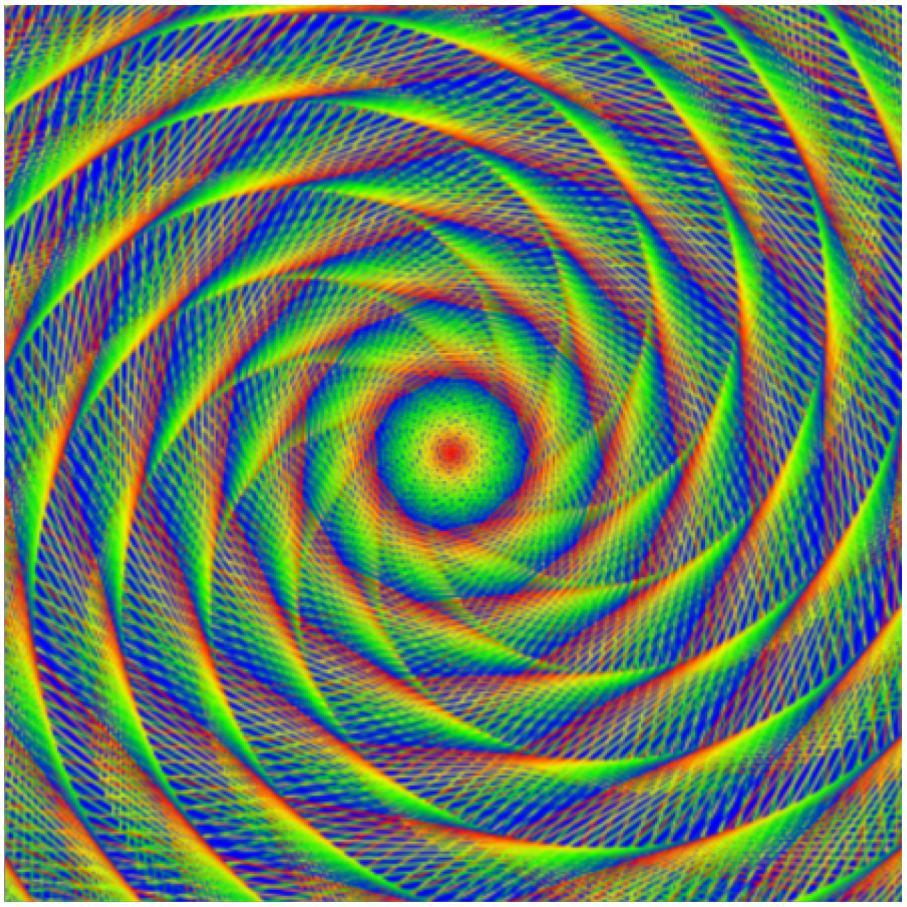
- A dar tus primeros pasos usando el lenguaje de programación Python
- Cómo dibujar líneas usando Python Turtle
- Cómo hacer giros
- Cómo cambiar el color del lápiz
- A usar bucles para repetir ciertas instrucciones y crear formas
- A usar mas bucles para crear patrones en espiral

Lo que necesitarás

- Una computadora con Python

¿Es arte, matemática o ciencias informáticas?

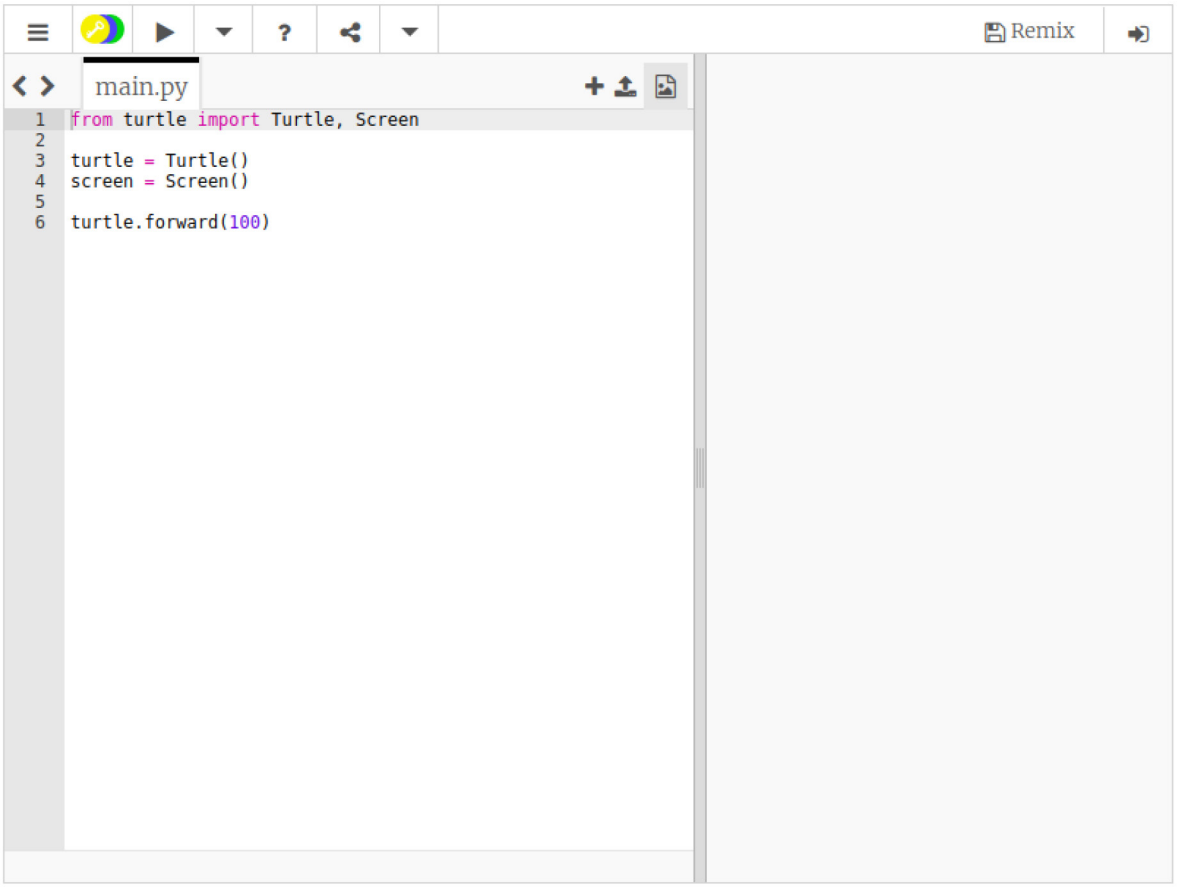
Echá un vistazo a la imagen de abajo. ¿Cómo la describirías? Es arte, matemática o ciencias computacionales?



Es una imagen generada por computadora, hacerla requiere cierto conocmientos de arte, matemática y ciencias informáticas. Veamos cómo podés crear imágenes como esta.

Dibujando una línea

¡La imagen de arriba está hecha exclusivamente con líneas! Para comenzar, debés saber cómo dibujar una línea usando un poco de código en Python. Podés utilizar el entorno de Python interactivo llamado Trinket para escribir código y luego probarlo.



• Hacé click en Run ▶ para ver cómo funciona este código.

• Ahora probá de cambiar el número en la línea `turtle.forward(100)`, hacé click en Run ▶ nuevamente y observa lo que pasa.

Girando

Ya usaste código para escribir una línea. ¡Buen trabajo! Ahora vamos a probar de hacer que la tortuga gire. Para hacer esto necesitarás darle instrucciones a la tortuga para que no solamente se mueva para adelante sino también para girar a la derecha o izquierda.

```
from turtle import Turtle, Screen

turtle = Turtle()
screen = Screen()

turtle.forward(100)
turtle.right(90)
turtle.forward(100)
```

• ¿Qué creés que pasará con el código de arriba? Hacé click en **Run** para ver si estabas en lo cierto.

`turtle.right(90)` gira el cursor 90 grados a la derecha. También podés girarlo a la izquierda con `turtle.left(90)`. Para cambiar el ángulo del giro simplemente cambiá el número de grados.

• Completá el cuadrado que has comenzado. Para hacerlo debés agregar las restantes líneas de código y apretar **Run**. Sigue intentando hasta que lo logres.

Desafío

Probá completar cada uno de los siguientes desafíos.

• Dibujá un rectángulo: dos de los cuatro lados tienen que ser mas largos.

• Dibujá un triángulo: ¿cuántos grados necesitás girar?

• Dibujá una cruz: funciona bien la combinación de movimiento hacia atrás y hacia adelante.

• Dibujá un círculo: ¿qué pasa si girás demasiado?

Cambiando de colores

El color predeterminado para el lápiz que usa la tortuga es el negro, y el color predeterminado para el fondo es el blanco. Podés cambiar los colores para hacer que tus formas se vean mejor aún.

- Observá el siguiente código. Contiene tres variables llamadas `R`, `G`, y `B`.

```
from turtle import Turtle, Screen

turtle = Turtle()
screen = Screen()
#screen.colormode(255)

R = 255
G = 255
B = 0

screen.bgcolor((R, G, B))
```

Las variables son un método para almacenar un valor y darle nombre. Por ejemplo, hay una variable llamada `R` con el valor `255`.

La siguiente línea de código está comentada en el Trinket de arriba, pero la necesitarás si estás usando otro editor. Por lo tanto, si no estás trabajando en Trinket, debés eliminar el símbolo `#` para des-comentar la línea.

```
screen.colormode(255)
```

- Corré el código y observa qué pasa.

- Probá de cambiar los valores de las tres variables y observá qué pasa. (Nota: el máximo valor es 255, y por encima de el no surtirá efecto). ¿Qué crees que representan R, G y B?

Respuesta

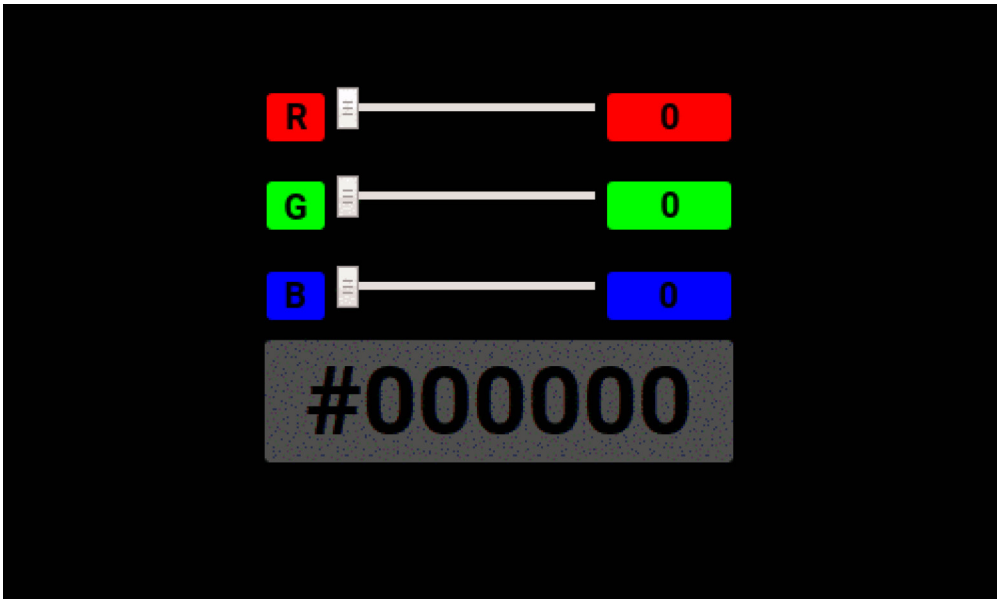
`R`, `G`, y `B` representan cuando rojo (r), verde (g) y azul (b) estarás utilizando en tu color. Cada uno puede tener un valor entre `0` y `255`.

Por lo tanto, para generar amarillo, podés probar lo siguiente:

```
R = 255
G = 255
B = 0
```

Colores RGB

Cuando queremos representar un color en un programa de computadora, podemos hacerlo definiendo la cantidad de rojo, azul y verde que componen a ese color. Esas cantidades generalmente se guardan en un solo byte y por lo tanto suele ser un número entre 0 y 255.



Aquí hay una tabla mostrando los valores para algunos colores:

Red (rojo)	Green (verde)	Blue (azul)	Color
255	0	0	Rojo
0	255	0	Verde
0	0	255	Azul
255	255	0	Amarillo
255	0	255	Magenta
0	255	255	Cyan

Podés encontrar un bonito selector de color para jugar en w3schools.

Podés cambiar el valor de tus variables tanto configurándolas con un nuevo valor como también mediante incrementos y decrementos a partir de un valor.

• También podés cambiar el color de la tortuga. Corré el código de abajo para ver qué pasa:

```
from turtle import Turtle, Screen

turtle = Turtle()
screen = Screen()

R = 255
G = 0
B = 124

turtle.color((R, G, B))
turtle.forward(100)
turtle.right(120)
turtle.forward(100)
```

Desafío

Probá completar cada uno de los desafíos a continuación:

- Completá el triángulo de arriba con un color a tu elección.
- Dibujá un cuadrado con los lados de cuatro tonos de rojo distintos.
- Dibujá una cruz hecha de cuatro colores diferentes.

Una pequeña ayuda

Para cambiar un color, podés simplemente sumar o restar valores de las variables originales.

Por lo tanto podrías modificar los colores haciendo lo siguiente:

```
R = 255
G = 0
B = 0

turtle.color((R, G, B))
turtle.forward(100)
turtle.right(120)

R -= 20
G += 20
B += 5

turtle.color((R, G, B))
turtle.forward(100)
turtle.right(120)
```

Repetición

Repetir líneas de código es una de las formas mas rápidas de hacer algo. Muchas veces, en las ciencias informáticas, tiene mas sentido repetir líneas de código que escribir otro grupo de instrucciones. Por ejemplo, el cuadrado que creaste antes usa las mismas dos instrucciones cuatro veces. En lugar de escribirlas cuatro veces, podés escribirlas una vez y agregar una instrucción para que se repitan.

En Python hay dos tipos de bucles que sueles usar: el bucle `while` y el bucle `for`. Si querés que una sección del código se repita para siempre, o hasta que se cumple cierta condición, entonces usar un bucle tipo while puede ser lo mejor. Si querés un bucle que se repita por una cantidad definida de veces, entonces es preferible el for.

- Aquí hemos usado el bucle `while True`. Esto significa que el código dentro del bucle (el código que está indentado) se repetirá por siempre. Podés probarlo en Trinket para ver qué es lo que hace, ¡recordá que se repetirá por siempre!

```
from turtle import Turtle, Screen

turtle = Turtle()

while True:
    turtle.forward(1)
    turtle.right(1)
```

Este tipo de bucle no será muy útil para dibujar formas con la tortuga, en las que necesitás mas precisión.

- En este ejemplo, se utilizó un bucle `for`. Presioná Run para ver lo que pasa.

```
from turtle import Turtle, Screen

turtle = Turtle()
screen = Screen()

turtle.penup()

for i in range(8):
  turtle.write(i)
  turtle.forward(20)
```

El bucle `for` repite las instrucciones una cantidad determinada de veces, en este caso 8 veces. El bucle `for` tiene asociada una variable (aquí se la llamó `i`). En este ejemplo, `i` arranca en `0` y se incrementa de a `1` cada vez. Vamos a aplicar esto al código para escribir un cuadrado:

```
from turtle import Turtle, Screen

turtle = Turtle()

for i in range(4):
    turtle.forward(100)
    turtle.right(90)
```

• Copiá y pegá este código en el editor de Trinket de arriba y correlo. Se le pide a la tortuga que repita las dos instrucciones cuatro veces para hacer un cuadrado.

• Una vez que hayas creado una forma usando un bucle, se puede repetir la forma una y otra vez poniendo esto dentro de otro bucle. Esta es una gran manera de dibujar espirales. Adaptá tu código para que quede así:

```
from turtle import Turtle, Screen

turtle = Turtle()

for i in range(30):
    for i in range(4):
        turtle.forward(100)
        turtle.right(90)
    turtle.right(25)
```

Se puede hacer una espiral girando un ángulo pequeño y luego moviéndose hacia adelante una pequeña cantidad. La sección de código para hacer el cuadrado está dentro de otro bucle `for` que lo repite 30 veces, cada vez girando el cursor 25 grados para hacer una forma de espiral continua.

Desafío

Probá de completar los siguientes desafíos:

• ¿Podés alterar el bucle `for` para que dibuje una espiral mas interesante usando una de las formas que hiciste antes, por ejemplo el triángulo o el círculo?

• Agregá algunas líneas extra donde alteres las variables `R`, `G` y `B` debería permitirte hacer una espiral multicolor. Hacé un intento de crear una espiral en arcoiris.

Una pequeña ayuda

Al igual que en el ejercicio anterior, podés sumar o restar valores de las variables `R`, `G` y `B`.

Simplemente modificá las variables dentro del bucle `for`:

```
from turtle import Turtle, Screen

turtle = Turtle()
screen = Screen()

R = 255
G = 0
B = 124

for i in range(30):
 turtle.color((R, G, B))
 for i in range(4):
     turtle.forward(100)
     turtle.right(90)
     ##AGREGA ALGO ACÁ
 turtle.right(25)
```

Probá esto para arrancar:

```
from turtle import Turtle, Screen

turtle = Turtle()
screen = Screen()

R = 255
G = 0
B = 124

for i in range(30):
 turtle.color((R, G, B))
 for i in range(4):
     turtle.forward(100)
     turtle.right(90)
     R -= 1
     G += 1
 turtle.right(25)
```

Mejores espirales

• Hacé un intento de leer el código que hay a continuación y adivinar qué es lo que hace. Luego ejecutalo para ver si estabas en lo correcto.

```
from turtle import Turtle, Screen

turtle = Turtle()
screen = Screen()
#screen.colormode(255)

R = 255
G = 0
B = 0

screen.bgcolor((255, 255, 255))

for i in range(2000):
  G += 1
  B += 0.5
  R -= 1
  turtle.color((R, G, B))
  turtle.forward(i)
  turtle.right(98)
```

• Probablemente la primera cosa que notes es que esto tardará un montón de tiempo en completarse. Podemos acelerar las cosas un poquito. Agregá la siguiente línea antes del bucle `for`:

```
turtle.speed(0)
```

• Corré el código nuevamente: debería ser un poco mas rápido.

• Nuestro código hizo un espiral multicolor al variar las variables `R`, `G` y `B`. Los colores son un poco repetitivos. ¿Podés hacerlo mas interesante?

Colores en bucles

Para obtener colores mas interesantes, podés escribir muchos colores en una larga lista y después ir cambiando el color de la tortuga utilizando los colores de la lista. Podés crear listas en Python usando los corchetes `[]`.

Debajo hay un ejemplo de una lista de colores RGB:

```
colours = [(85, 211, 136), (197, 196, 126), (235, 233, 166), (25,
135, 222), (211, 64, 159), (159, 165, 106), (178, 160, 125), (36,
192, 70), (231, 184, 204), (63, 203, 219)]
```

• El próximo paso se pone un poco complicado. Echá un vistazo a el código debajo, luego correlo para ver que sucede.

```
from turtle import Turtle, Screen

turtle = Turtle()
screen = Screen()
#screen.colormode(255)

screen.bgcolor((0, 0, 255))
turtle.speed(0)

colours = [(85, 211, 136), (197, 196, 126), (235, 233, 166), (25,
135, 222), (211, 64, 159),
           (159, 165, 106), (178, 160, 125), (36, 192, 70), (231,
184, 204), (63, 203, 219)]

for i in range(10):
  turtle.color(colours[i])
  turtle.forward(10)
```

La línea `turtle.color(colours[i])` le dice al programa que elija el “`i` avo” item de la lista. Recordá que icomienza por 0 y va hasta 9.

• ¿Qué pasaría si quisieras una línea mas larga? Probá de cambiar el número de repeticiones en el `for` por `range(20)` y observá qué pasa. ¿Te da un error?

Módulo al rescate

En el ejemplo anterior, necesitabas un método para ir repitiendo entre los elementos de una lista, cuando `i` llega a 9 vuelve a comenzar y arranca por el elemento 0 de la lista nuevamente. Aquí es donde el operador módulo `%` puede ayudarte.

• Observá el código de abajo: ejecutalo y fijate si podés darte cuenta qué está pasando. Deberías obtener 0 al comenzar.

```
print(18 % 6)
```

• Probá cambiar los números en el comando `print` . Debajo hay algunos ejemplos para probar:

```
print(17 % 6)
print(12 % 6)
print(13 % 6)
print(6 % 6)
print(0 % 6)
print(1 % 6)
print(8 % 6)
print(11 % 6)
```

• ¿Pudiste entender qué está pasando? El operador `%` imprime el resto de la división. Por ejemplo, 15 ÷ 6 es 2 y el resto es 3. Por lo tanto 15 % 6 será 3. Podemos usar este operador para ayudar con el problema de pasarnos de el final de la lista. Si `range` es mas alto que el largo de la lista, podés hacer un `%` del largo de la lista.

• Observá el ejemplo de abajo y leé el código atentamente para comprender cómo está usado el operador módulo.

```
from turtle import Turtle, Screen

turtle = Turtle()
screen = Screen()
#screen.colormode(255)

screen.bgcolor((0, 0, 255))

colours =  [(28, 138, 126), (46, 120, 108), (143, 13, 132), (200,
18, 85), (78, 224, 239),
             (88, 20, 216), (36, 108, 248), (99, 237, 136), (146,
31, 91), (10, 154, 152),
             (127, 160, 117), (181, 203, 36), (201, 97, 179), (1,
3, 180), (21, 27, 214),
            (26, 94, 11), (161, 246, 62), (224, 51, 159), (51, 62,
148), (115, 224, 188)]

turtle.speed(0)

for i in range(2000):

# La línea de abajo se asegura que siempre se elija un color,
incluso si i es mayor que el largo de la lista.

  turtle.color(colours[i % len(colours)])
  turtle.forward(i)
  turtle.right(98)
```

Listas enormes

Ahora podés probar de crear una lista de colores que sea un poco mas larga que la anterior. Para hacer esto podés usar un bucle `while`. A diferencia del bucle `for`, el bucle `while` sigue repitiéndose hasta que se cumple una determinada condición.

• Observá el código de abajo. El bucle `while` se usa para incrementar gradualmente el valor de `G` hasta que llega a 255. En cada repetición se agrega el color a la lista.

```
colours = []

R = 255
G = 0
B = 0

while G < 255:
  colours.append((R, G, B))
  G += 5

print(colours)
```

• ¿Podés agregar dos bucles `while` adicionales para agregar mas colores? El próximo bucle debería decrementar gradualmente `R` hasta que llegue a 0. El último bucle, debería incrementar `B` hasta llegar a 255. Has un intento, si te quedás trabado te mostraremos cómo lograrlo en la última sección.

Uniendo todo

Ahora podés usar los bucles `while` en conjunto con el código de la espiral para una que sea realmente linda. Observá el código de abajo y asegurate entender lo que está haciendo. Probá de cambiar el valor de las variables en el bucle para ver qué efecto genera en el resultado.

```
from turtle import Turtle, Screen

turtle = Turtle()
screen = Screen()
#screen.colormode(255)

R = 255
G = 0
B = 0

screen.bgcolor((0, 0, 255))

turtle.speed(0)

colours = []

while G <= 255:
    colours.append((R, G, B))
    G += 1

while R >= 0:
    colours.append((R, G, B))
    R -= 1

while B < 255:
    colours.append((R, G, B))
    B += 1

while G > 0:
    colours.append((R, G, B))
    G -= 1

while R < 255:
    colours.append((R, G, B))
    R += 1
```

¿Con qué seguir?

- Aprendé cómo usar funciones para dibujar copos de nieve usando Turtle.
- Creá historias interactivas usando listas en Python con el recurso Storytime.
- Hacé tus primeros pasos controlando objetos físicos con Python y Raspberry Pi.

La fundación Raspberry Pi suministra contenidos de aprendizaje de programación sin cargo. Encuentre más información en https://projects.raspberrypi.org/en/ (inglés)