3. Hola, tortuguitas!

Hay muchos *módulos* de Python que proporcionan características muy poderosas que podemos usar en nuestros propios programas. Algunos de estos pueden enviar correo electrónico, o buscar páginas web. A continuación veremos en este capítulo crear tortugas y conseguir que dibujen formas y patrones.

Las tortugas son divertidas, pero el verdadero propósito del capítulo es enseñarnos a nosotros mismos un poco más de Python y el desarrollo de nuestro tema del *pensamiento computacional*, o *pensar como un científico de la computación*. La mayoría de Python cubre aquí, aunque se estudiarán en profundidad más adelante.

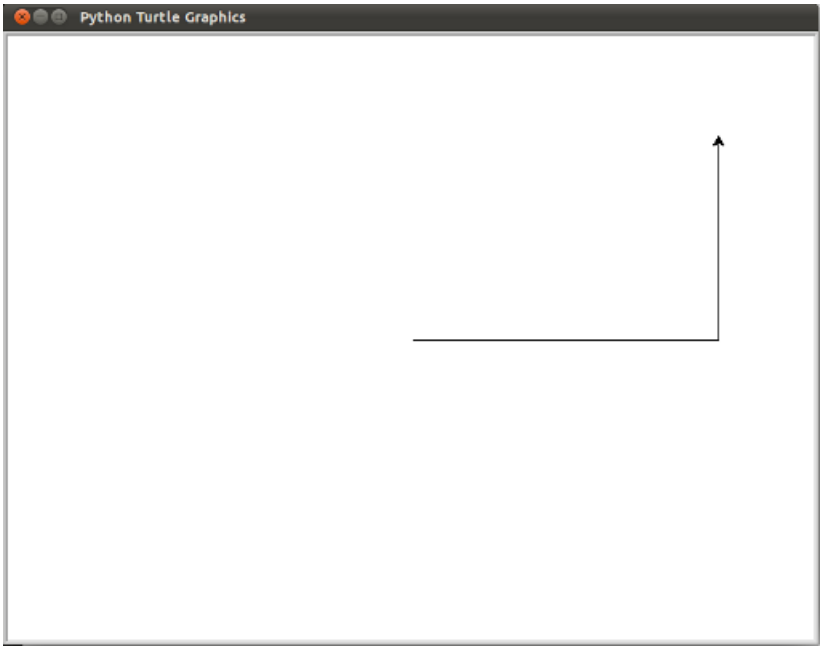
3.1. Nuestro primer programa de tortuga

Vamos a intentar un par de líneas en la terminal de Python para crear una nueva tortuga y empezar a dibujar un rectángulo. (Vamos a llamar a la variable que se refiere a nuestra primera tortuga alex, pero puede elegir otro nombre si usted sigue las reglas de denominación del capítulo anterior).

```
>>> import turtle
>>> turtle.setup(800, 600)
>>> alex = turtle.Turtle()
>>> alex.forward(300)
>>> alex.left(90)
>>> alex.forwrad(200)
>>> exit()
```

Después de la segunda orden, una nueva ventana se abrirá. El tercer comando coloca un cursor – llamando cariñosamente a una `tortuga` en este módulo – que hemos llamado `alex`.

Las siguientes tres líneas mueven alex hacia adelante, dan una vuelta hacia la izquierda, y lo mueve hacia adelante una vez más, completando dos lados de un rectángulo. Después de estos comandos se han introducido, aparecerá una ventana que se parece a esto:



Aquí hay un par de cosas que usted necesita entender acerca de este programa.

- La primera línea le dice a Python para cargar un módulo llamado `turtle`. Dicho módulo nos trae dos nuevos tipos que podemos utilizar: el tipo de `Turtle`, y el tipo de `Screen`. La notación de puntos `turtle.Turtle` significa *"El tipo de tortuga que se define en el módulo de tortuga"*. (Recuerda que Python distingue entre mayúsculas y minúsculas, así que el nombre del módulo, con una `t` minúscula, es diferente al carácter Tortuga.)
- A continuación, crear y abrir lo que llamamos una pantalla (que se perfer llamarlo de una ventana), que se le asigna a la variable `wn`. Cada ventana contiene un **lienzo (canvas)**, que es el área dentro de la ventana en la que podemos sacar. En la línea 3 creamos una tortuga. La variable de `alex` se hace

para referirse a esta tortuga. Estas tres primeras líneas nos se prepara, listo para hacer algunas cosas útiles.

- En las líneas 4–6, instruimos al **objecto** de alex que se mueva, y que gire. Hacemos esto mediante la **invocación**, o la activación, los **métodos** de alex — estas son las instrucciones que todas las tortugas saben cómo responder.
- La última línea también desempeña un papel: la variable `wn` se refiere a la ventana activa. Cuando se invoca el método `exitonclick`, detiene la ejecución del programa, y espera a que el usuario haga clic con el ratón en algún lugar de la ventana. Cuando este evento click se produce, la respuesta es cerrar la ventana de la tortuga y la salida (la ejecución de la parada) del programa de Python.

Adelante, haga clic en la ventana despúes de haber introducido el último comando.

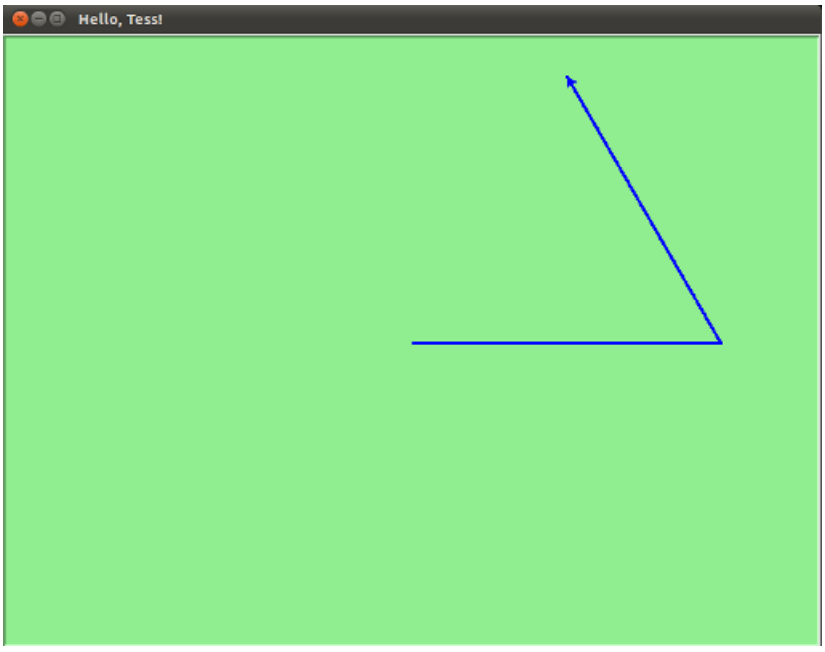
Un objeto puede tener varios métodos — las cosas que puede hacer — y que también pueden tener **atributos** — (a veces son llamas *propiedades*). Por ejemplo, cada tortuga (turtle) tiene un atributo de *color*. El método de invocación `alex.color("rojo")` que hará alex rojo, y el dibujo será de color rojo también.

El color de la tortuga (turtle), la anchura de la pluma, la posición de la tortuga dentro de la ventana, la cual forma en que se enfrenta, y así sucesivamente, son partes de su **estado** actual. Del mismo modo, el objeto de la ventana tiene un color de fondo y un texto en la barra de título, y un tamaño y posición en la pantalla. Todos ellos forman parte del estado del objecto de la ventana.

Un buen número de métodos que existen, que nos permiten modificar las tortugas y los objetos de la ventana. Vamos a mostrar un par. Sólo hemos comentado las lineas que son diferentes a los ejemplos anteriores (y que hemos utilizado un nombre de variable diferente para esta tortuga). Además, vamos a poner este ejemplo en un script llamado `tess.py`, ya que es conseguir un poco demasiado tiempo para escribir una y otra vez en el intérprete:

```
1  import turtle
2
3  turtle.setup(800, 600)      # establecer el tamaño de la ventana de 800 por          600 píxeles
4  wn = turtle.Screen()        # establecer wn al objeto de la ventana
5  wn.bgcolor("lightgreen")    # establecer el color de fondo de la ventana
6  wn.title("Hola, Tess!")     # establecer el título de la ventana
7  tess = turtle.Turtle()
8  tess.color("blue")           # hacer tess azul
9  tess.pensize(3)              # establecer el ancho de la pluma
10 tess.forward(300)
11 tess.left(120)
12 tess.forward(300)
13
14 wn.exitonclick()
```

La ejecución de este programa creará una ventana gráfica que se ve así:



Cuando ejecutamos este programa, esta nueva ventana aparece, y permanecerá en la pantalla hasta que haga clic en él.

Extender este programa ...

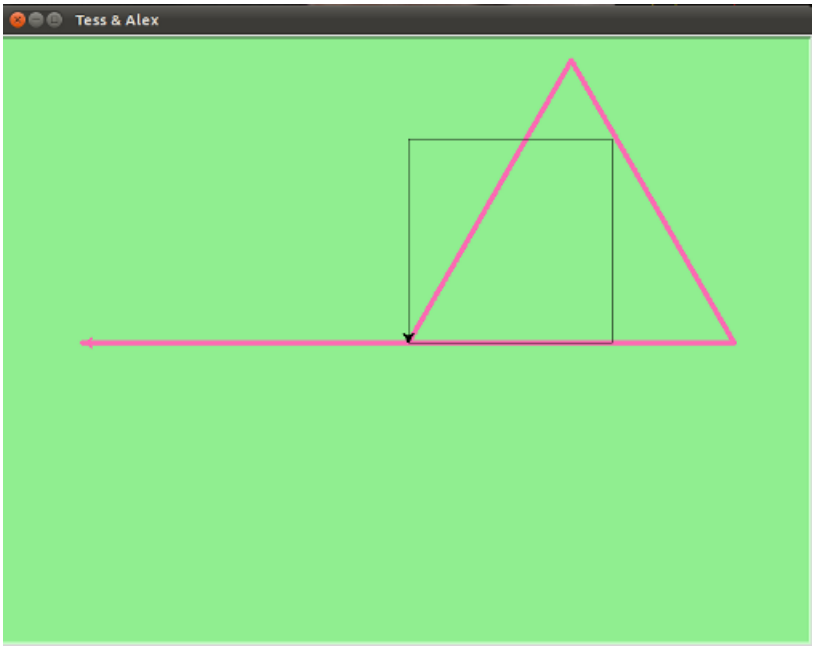
1. Modificaremos este programa antes de crear la ventana, se le pedirá al usuario que introduzca el color de fondo deseado. Se debe almacenar las respuestas del usuario en una variable, y modificar el color de la ventana de acuerdo a los deseos del usuario. (Pista: se puede encontrar una lista de nombres de colores permitidos en http://www.tcl.tk/man/tcl8.4/TkCmd/colors.htm. Incluye algunos bastante inusuales, como "peach puff" (bocanada de melocotón) y "HotPink" (rosado caliente).)
2. Los cambios similares para permitir al usuario, en tiempode ejecución, para establecer el color de Tess.
3. Haga lo mismo para el ancho de la pluma de Tess. *Sugerencia:* el diálogo con el usuario devolverá una cadena, pero el método de tess `pensize` espera que su argumento es un entero. Así que tendrás que convertir la cadena en un entero de pasar a `pensize`.

3.2. Casos — una manada de tortugas

Al igual que podemos tener muchos enteros diferentes en un programa, podemos tener muchas tortugas. Cada uno de ellos se llama una **instancia**. Cada instancia tiene sus propios atributos y métodos – por lo que Alex puede dibujar con un lápiz negro delgado y estar en alguna posición, mientras que Tess que podría estar pasando en su propia dirección con un bolígrafo de color rosa de grasa (fat pink). Así que aquí está lo que sucede cuando Alex termina su rectángulo, y Tess completa su triángulo, en un programa llamado :download: *tess_y_alex.py <recursos/cap03/tess_y_alex.py>*:

```
import turtle

# setup the window and its attributes
turtle.setup(800, 600)
wn = turtle.Screen()
wn.bgcolor("lightgreen")
wn.title("Tess & Alex")

# instantiate (create) tess and set her attributes
tess = turtle.Turtle()
tess.color("hotpink")
tess.pensize(5)

# instantiate alex
alex = turtle.Turtle()

# draw an equilateral triangle with tess
tess.forward(320)
tess.left(120)
tess.forward(320)
tess.left(120)
tess.forward(320)
tess.left(120)

# turn tess around and move her away from the origin
tess.right(180)
tess.forward(320)

# make alex draw a square
alex.forward(200)
alex.left(90)
alex.forward(200)
alex.left(90)
alex.forward(200)
alex.left(90)
alex.forward(200)

wn.exitonclick()
```

que genera este cuando se ejecuta:



Aquí están algunas *Cómo pensar como un experto en computación* observaciones:

- Hay 360 grados en un círculo completo. Si se suman todos las vueltas que un tortuga hace, *no importa las medidas que se produjo entre las curvas*, usted puede fácilmente averiguar si se suman a un múltiplo de 360. Esto debe convencernos de que Alex se enfrenta exactamente en la misma dirección que tenía cuando fue creado. (Convenios de la geometría a 0 grados mirando hacia el Este, y ese es el caso aquí.)
- Podríamos haber dejado fuera el último turno de Alex, pero eso no habría sido tan satisfactoria. Si se le pregunta a dibujar una forma cerrada como un cuadrado o un rectángulo, que es una buena idea para completar todas las vueltas y dejar a la tortuga donde comenzó, en la misma dirección con la que comenzó. Esto hace que el razonamiento sobre el programa y los trozos de la composición de código en programas más grandes más fácil para nosotros los seres humanos.
- Hicimos lo mismo con tess: sacó su triángulo, y se convirtió a través de un total 360 grados. Entonces ella se dio la vuelta y se trasladó a un lado. Incluso la línea en blanco 23 es una pista acerca de cómo *el fraccionamiento mental* del programador está trabajando: en cuanto a grandes movimientos de Tess se fragmentada como "dibujar el triángulo" (líneas 17-22) y luego "se alejan del origen" (líneas 25 y 26).
- Uno de los usos principales de los comentarios es registrar su fraccionamiento mental, y grandes ideas. No siempre son explícitas en el código.
- Y, uh-huh, dos tortugas pueden no ser suficientes para un rebaño, pero usted consigue la idea!

3.3. El bucle for

Cuando nosotros dibujamos el cuadrado, fue algo bastante tedioso. Tuvimos que pasar y luego a su vez curvar, ect, ect., en cuatro ocasiones. Si tuviéramos que dibujar un hexágono, o un Octógono, o un polígono de 42 lados, hubiera sido una pesadilla.

Por lo que un bloque de construcción básico de todos los programas debe de ser capaz de repetir el código, una y otra vez.

Python bucle **for** resuelve esto para nosotros.

Digamos que tengo algunos amigos, y nos gustaria enviarles un correo electrónico cada invitándolos a nuestro partido. Dado que no sabemos cómo enviar correo electrónico, sin embargo, por el momento sólo tendremos que imprimir un mensaje por cada amigo:

```
for f in ["Joe", "Amy", "Brad", "Angelica", "Zuki", "Thandi", "Paris"]:
  invitación = "Hola " + f + ". Por favor, ven a mi fiesta el sábado!
  print(invitación)
```

Cuando llevamos a cabo esto, la salida se ve asi:

```
Hola Joe.  Por favor, ven a mi fiesta el sábado!
Hola Amy.  Por favor, ven a mi fiesta el sábado!
Hola Brad.  Por favor, ven a mi fiesta el sábado!
Hola Angelina.  Por favor, ven a mi fiesta el sábado!
Hola Tsuki.  Por favor, ven a mi fiesta el sábado!
Hola Thandi.  Por favor, ven a mi fiesta el sábado!
Hola París.  Por favor, ven a mi fiesta el sábado!
```

Estudio de este ejemplo atentamente y tenga en cuenta lo siguiente:

- `["Joe", "Amy", "Brad", "Angelica", "Zuki", "Thandi", "Paris"]` es un nuevo tipo de datos, llamado **lista**, que puede contener un conjunto de valores. Vamos a estudiar las listas de manera más formal en un capítulo posterior, pero por ahorasólo tiene que saber que poseen un conjunto de valores separados por comas y encerrados entre corchetes ([,]).
- **f** en esta declaración `for` que se llama la **variable de bucle** (loop variable). Que se asigna a cada valor en la lista, uno a la vez y en el orden en que aparecen en la lista.
- Líneas 2 y 3 son el **cuerpo del bucle** (loop body). El cuerpo del bucle es siempre sangría. El sangrado se determina exactamente qué estados están "en el bucle".
- En cada *iteración* o *paso* del bucle, en primer lugar una comprobación se realiza para ver si hay elementos aún más para ser procesado. Si no hay ninguno a la izquierda (esto se llama la **condición de finalización** del bucle), el ciclo ha terminado. Ejecución del programa continúa en la siguiente sentencia después de que el cuerpo del bucle.
- Si hay temas que no hayan procesado, la variable de bucle se actualiza para hacer referencia al siguiente elemento de la lista. Esto significa, en este caso, que el cuerpo del bucle se ejecuta aquí siete veces, y cada vez que `f` se refieren a un amigo diferente.
- Al final de cada ejecución del cuerpo del bucle, Python vuelve a la `for` declaración, para ver si hay más elementos para ser manipulados.

3.4. Flujo de la ejecución del bucle for

Como un programa se ejecuta, el intérprete siempre comprueba que la declaración está a punto de ser ejecutado. Llamamos a esto el **control de flujo**, el **flujo de ejecución** del programa . Cuando los humanos ejecutan los programas, a menudo usan sus dedos para señalar a cada instrucción en turno. Por lo que podría pensar en el flujo de control como "dedo que se mueve en Python".

Control de flujo hasta ahora ha sido estrictamente de arrriba a abajo, una declaración a la vez. El lazo de bucle `for` cambia esto.

Control de flujo es a menudo fácil de visualizar y comprender si se dibuja un diagrama de flujo. Esto muestra los pasos exactos y la lógica de cómo funciona la instrucción `for` para ejecutar.

3.5. El bucle simplifica nuestro programa de tortuga

Para dibujar un cuadrado, nos gustaría hacer lo mismo cuatro veces — mover la tortuga, y darle vuelta. Antes utilizábamos 8 líneas que alex utilizaba para dibujar cuatro lados de un cuadrado. Esto hace exactamente lo mismo, pero utilizando sólo tres líneas:

```
for i in [0, 1, 2, 3]:
    alex.forward(250)      # forward (adelante)
    alex.left(90)          # left (izquierda)
```

Mientras que ahorrar unas líneas de código que puede ser conveniente, pero no es el problema aquí. Lo que es mucho más importante es que hemos encontrado un *patrón de repetición* de las declaraciones, y reorganizado nuestro programa para repetir el patrón. Encontrar los trozos y de alguna manera conseguir nuestros programas organizados alrededor de los trozos es una habilidad vital en *Cómo pensar como un científico de la computación*.

Los valores en la lista, `[0, 1, 2, 3]`, se les proporcionó para que el cuerpo del bucle ejecuta cuatro veces. Podríamos haber usado cualquiera de los cuatro valores, pero estos son los convencionales para su uso. De hecho, son tan populares que Python nos da una especial función `incorporada` de serie para este propósito.

```
>>> for i in range(4):
...     print(i)
...
0
1
2
3
>>>
```

Tenga en cuenta que:

- `range` (rango) crea un objeto que puede ofrecer una secuencia de valores para el bucle `for`.
- Científicos de la computación le gusta contar desde 0! Cuando se pasa un `range` (rango) argumento de tipo entero, los valores asignados a la variable de bucle empezar a 0, y subir por uno a uno menos que el valor del argumento.

También podemos crear una lista de números al pasar la salida de una llamada a la función de `range` (rango) a la función de tipo de la `list` (lista):

```
>>> list(range(6))
[0, 1, 2, 3, 4, 5]
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Intente lo siguiente ...

¿Qué `list(range(0))` a cambio? ¿Qué pasa con la `list(range(1))`?

Nuestro pequeño truco antes para asegurarse de que `Alex` hizo la última vuelta para completar 360 grados ha dado sus frutos: si no hubiéramos hecho eso, entonces no habría sido capaz de utilizar un bucle para el cuarto lado de el cuadrado. Se habría convertido en un "caso especial", diferente de los otros lados. Cuando sea posible, nos gustaría mucho preferidamente hacer nuestro código ajustado a un patrón general, en lugar de tener que crear un caso especial.

Así que para repetir algo cuatro veces, un buen programador de Python podría hacer esto:

```
for i in range (4):
    alex.forward (250)
    alex.left (90)
```

A estas alturas ya debería ser capaz de ver la forma para cambiar nuestro programa anterior para que `tess` también pueda utilizar un bucle `for` para dibujar su triángulo equilátero.

¿Qué pasaría si hiciéramos este cambio? ...

```
for clr in ["amarillo", "rojo", "púrpura", "azul"]:
      alex.color(clr)
      alex.forward(250)
      alex.left(90)
```

Una variable también se le puede asignar un valor que es una list (lista). Así que las listas también se pueden utilizar en otras situaciones que el bucle `for`. El código anterior se podría reescribir así:

3.6. A poco más de métodos de tortugas y trucos

Métodos de tortuga puede utilizar los ángulos negativos o las distancias. Así `tess.foward (-100)` se moverá tess hacia atrás y `tess.left (-30)` le da vuelta a la derecha. Además, dado que hay 360 grados en un círculo, de cumplir los 30 a la izquierda tendrá frente a usted en la misma dirección de giro de 330 a la derecha! (La animación en la pantalla será diferente, aunque — usted será capaz de decir si Tess se está convirtiendo en sentido horario o antihorario!)

Esto sugiere que no necesitamos tanto un método de girar a la izquierda y a la derecha — que podría ser minimalistas, y sólo hay un método. También hay un método *hacia atrás*. (Si usted es muy nerd, es posible disfrutar de decir `alex.backward (-100)` para mover alex hacia adelante!)

Parte de *pensar como un científico* es comprender mejor la estructura y relaciones ricas en su campo. Así que revisar algunos hechos básicos acerca de la geometría y las líneas de número, como estamos haciendo aquí es un buen comienzo si vamos a jugar con las tortugas.

Lápiz de la tortuga se pueden recoger o dejar. Esto nos permite mover una tortuga a un lugar diferente sin dibujar una línea. Los métodos son

```
alex. penup()
alex. forward(100)   # esto hace que mueva alex, pero no hay línea dibujada
alex. pendown ()
```

Cada tortuga puede tener su propia forma. Los que están disponibles "fuera de la caja" son la `arrow` (flecha), `blank` (espacio en blanco), `circle` (círculo), `classic` (clásico), `square` (cuadrado), `triangle` (triángulo), `turtle` (tortuga).

```
alex.shape("turtle")
```

Usted puede acelerar o ralentizar la velocidad de la tortuga de animación. (Animación controla la rapidez con la tortuga gira y se mueve hacia adelante). Los ajustes de velocidad se puede ajustar entre 1 (lento) a 10 (más rápido). Sin embargo, si se establece la velocidad a 0, que tiene un significado especial — desactivar la animación e ir lo más rápido posible.

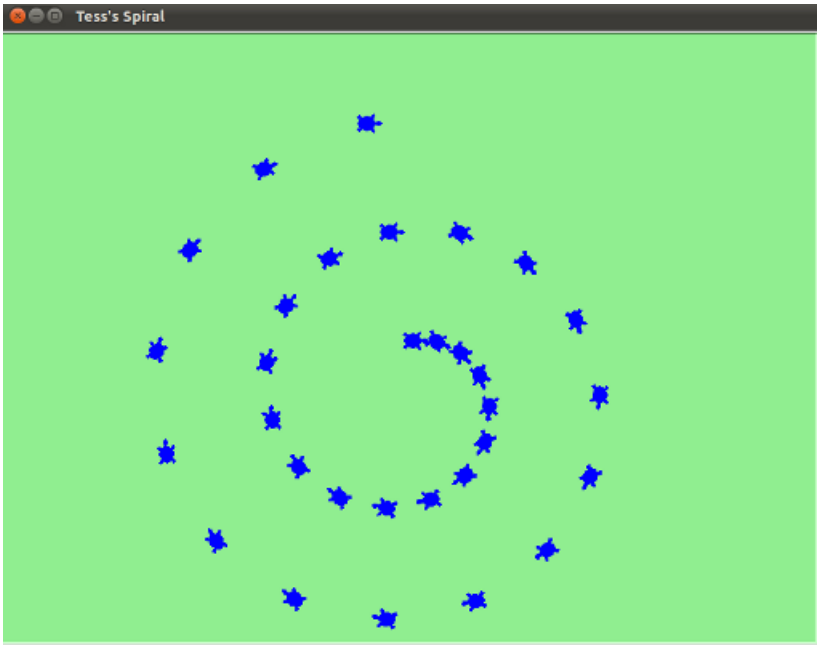
```
alex.speed(10)
```

Una tortuga puede crear un "sello" de su huella con el lienzo, y esto seguirá así después que la tortuga se haya movido a otra parte. Sellando los trabajos, aun cuando la pluma está hacia arriba.

Vamos a hacer un ejemplo que muestra algunas de estas nuevas características en **spiral.py**:

```
import turtle

turtle.setup(800, 600)
wn = turtle.Screen()
wn.bgcolor("lightgreen")
wn.title("Tess's Spiral")

tess = turtle.Turtle()
tess.shape("turtle")
tess.color("blue")

tess.penup()                    # this is new
size = 20
for i in range(30):
    tess.stamp()                # Leave an impression on the canvas
    size = size + 3             # increase the size on every iteration
    tess.forward(size)          # move tess along
    tess.right(24)              # and turn her

wn.exitonclick()
```

que genera esto cuando se ejecuta:



Ten cuidado ahora: con todos excepto una de las formas que se ve en la pantalla de aquí son las huellas creadas por el sello. Pero el programa sólo cuenta con una instancia de tortuga – ¿Se puede averiguar cuál es el tess real? (Pista: si usted no está seguro, escriba una nueva línea de código después del bucle for para cambiar el color de Tess, o poner su pluma y dibuja una línea, o para cambiar su forma, etc)

3.7. Glosario

attribute

Algún estado o valor que pertenece a un objeto particular. Por ejemplo, Tess tiene un color.

canvas

Una superficie dentro de una ventana de dibujo, donde se lleva a cabo.

control de flujo

Ver el flujo de ejecución en el próximo capítulo.

Sentencia For Loop

Una declaración en Python para la repetición conveniente de sentencias en el cuerpo del bucle.

LIST:

Un tipo de datos en Python que contiene una colección de valores. Listas en Python están entre corchetes ([,]) y elementos de la lista están separados por comas (,).

cuerpo del bucle

Cualquier número de instrucciones anidadas dentro de un bucle. La anidación se indica por el hecho de que las declaraciones se inserta debajo de la sentencia de bucle for.

variable de bucle

Una variable que se utiliza como parte de un bucle for. Se le asigna un valor diferente en cada iteración del bucle, y se utiliza como parte de la condición de terminación del bucle, ejemplo Un objeto que pertenece a una clase de Tess. Y Alex son las diferentes instancias de la Tortuga de clase

Método

Una función que se adjunta a un objeto. Invocar o activar el método hace que el objeto de responder de alguna manera, por ejemplo, hacia adelante es el método al decir tess.forward (100).

invocar

Un objeto tiene métodos. Nosotros usamos el verbo en el sentido de invocar activar el método. La invocación de un método se hace poniendo entre paréntesis después del nombre del método, con algunos argumentos posibles. Así wn.exitonclick () es una invocación del método exitonclick.

módulo

Un archivo que contiene definiciones de Python y declaraciones destinados a ser utilizados en otros programas Python. El contenido de un módulo se ponen a disposición del otro programa con la instrucción de importación.

Objeto

Una "cosa" a la que puede referir una variable. Esto podría ser una ventana de la pantalla, o una de las tortugas que ha creado.

alcance

Una función integrada en Python para generar secuencias de números enteros. Es especialmente útil cuando tenemos que escribir un bucle para que se ejecute un número determinado de veces.

terminación de condición

Una condición que se produce lo que provoca un bucle para detener la repetición de su cuerpo. En los bucles for que vimos en este capítulo, la condición de finalización ha sido, cuando hay elementos no asignar a la variable de bucle.

3.8. Ejercicios

1. Escriba un programa que imprima `Nosotros nos gusta trabajar con la tortuga de Python!` 100 veces.
2. El vocabulario utilizado en los objetos de software se inspiró en los objetos del mundo real. Vamos a invertir el proceso del pensamiento, y describir un objeto del mundo real, su teléfono celular, utilizando el mismo vocabulario que usamos con los objetos de software.

   Dé tres atributos y tres métodos de su teléfono móvil.
3. Crear una lista llamada `meses` que contenga 12 cadenas con los nombres de los meses del año en el orden en que ocurren.
4. Escribir un programa que utiliza un bucle para recorrer `meses` para imprimir

   ```
   Uno de los meses del año es Enero
   Uno de los meses del año es Febrero
   ... etc
   ```
5. Suponga que tiene la asignación x = [12, 10, 32, 3, 66, 17, 42, 99, 20]
   a. Escribir un bucle que imprima cada uno de los números en la nueva línea.
   b. Escribir un bucle que imprima cada número y su cuadrado en una nueva línea.

Desafíos adicionales (para los estudiantes mas inteligentes)...

c. Escriba un bucle que suma todos los números de la lista en una variable llamada `total`. Usted debe establecer la variable `total` para tener el valor 0 antes de comenzar a agregar, y imprima el vavol en `total` después del ciclo se ha complatado.
d. Imprimir el producto de todos los números en la lista. (Producto de todos los medios multiplicados entre si).

1. Use `for` bucles para hacer una tortuga dibujar estos polígonos regulares (regular significa que todos los lados tienen la misma longitud, todos los ángulos son iguales):
    - Un triángulo equilátero
    - Un cuadrado
    - Un hexágono (seis lados)
    - Un octógono (ocho lados)
2. Un estudiante borracho hace un giro al azar y luego toma 100 pasos hacia adelante, da otro giro al azar, y luego da otros 100 pasos mas, y sigue girando otra cantidad al azar, etc. El estudiante de ciencias sociales registra los angúlos de cada vez antes de los próximos 100 pasos que tomo el estudiante borracho. Sus datos experimentales son `[160, -43, 270, -97, -43, 200, -940, 17, -86]`. (Los ángulos positivos son hacia la izquierda.) Use la tortuga para dibujar el camino recorrido por nuestro amigo borracho.
3. Mejore su programa anterior para que también nos diga lo que el estudiante borracho donde ha finalizado dando tumbos. (Suponga que comienza en la partida 0).
4. Si se va a dibujar un polígono regular de 18 lados, ¿qué ángulo se tiene que utilizar para girar a la tortuga en cada esquina?
5. En el modo interactivo, anticiparse a lo que cada una de las siguientes líneas se hacen, y luego grabar lo que sucede. Puntuación mismo, dando un punto por cada uno que anticipar correctamente

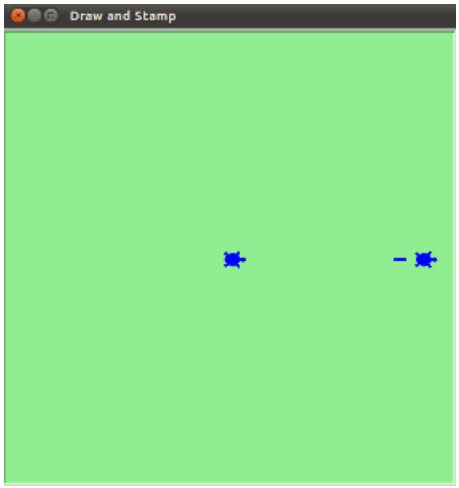
```
>>> import turtle
>>> wn = turtle.Screen()
>>> tess = turtle.Turtle()
>>> tess.right(90)
>>> tess.left(3600)
>>> tess.right(-90)
>>> tess.speed(10)
>>> tess.left(3600)                                    >>> tess.speed(0)
>>> tess.forward(-100)
```

6. Escribir un programa para dibujar una forma de esta manera:

Sugerencias

- Prube esto en un pedazo de papel, el movimiento y dar vuelta a su teléfono movil como si fuera una tortuga. Mira cuántas rotaciones completas tu móvil hace antes de completar la estrella. Ya que cada giro completo es de 360 grados, se puede averiguar el número total de grados que su teléfono fue girado. Si usted divide eso por 5, porque hay cinco puntos de la estrella. Usted sabra cuántos grados a su vez a la tortuga en cada punto.
- Puede ocultar una tortuga detrás de su capa de invisibilidad, si usted no quiere que se muestra. Todavía sacará sus líneas si su pluma es hacia abajo. El método es invocado como `tess.hideturtle()`. `tess.showturtle()` hace que la tortuga visible de nuevo.

7. Escriba un programa que atraiga a una pantalla que se ve algo como esto:

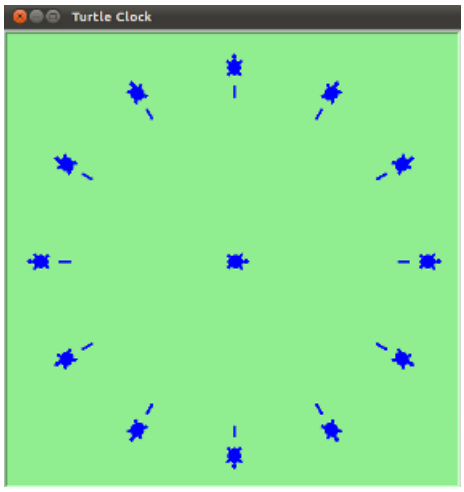


Hint

Use una combinación de los `penup`, `forward`, `pendown`, `stamp`, y `backward` metodos de la turtuga para conseguir que esto funcione.

La tortuga debe terminar de nuevo en donde empezo cuando el programa termine.

8. Ahora se basan en la solución del ejercicio anterior escribir un programa para dibujar una esfera de un reloj que se ve algo como esto:



9. Crear una tortuga, y asignele a una variable. Cuando se le pregunta por su tipo, ¿qué se obtiene?
10. ¿Cuál es el nombre colectivo para las tortugas? (Pista: no vienen en manadas.)