COMP1021
Introduction to Computer Science

Using For Loops with Turtle Graphics

Gibson Lam and David Rossiter

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Explain the difference between while loops and for loops
  2. Use for loops to create patterns with graphics programming
  3. Use nested for loops to create patterns with graphics programming

For Loops in Turtle Graphics

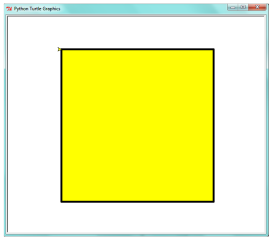
- Let's look at using for loops with graphics
- The basic difference between while loops and for loops:
- While loops – sometimes you don't know how many times the loop will repeat
- For loops – you exactly control the start value, end value and increment value, so you can work out exactly how many times the loop will repeat

Drawing a Square Using a For Loop

- Let's use a for loop to make a square:

```
for i in range(4):
    turtle.forward(400)
    turtle.right(90)
```

The content of the loop is executed four times, to draw four sides of the square

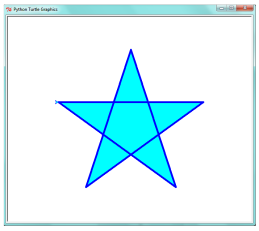


The letter 'i' is quite commonly used for the loop variable of a loop ('i' for 'index'), although you can use any variable name

Drawing a Star Shape Using a For Loop

- You can alter the program to draw a star shape
- This for loop runs five times to create the five lines of the star:

```
for i in range(5):
    turtle.forward(400)
    turtle.right(144)
```

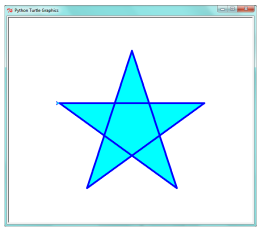


Drawing a Star Shape Using a For Loop

- You can alter the program to draw a star shape
- This for loop runs five times to create the five lines of the star:

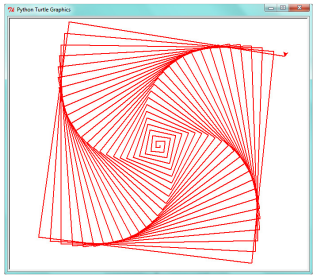
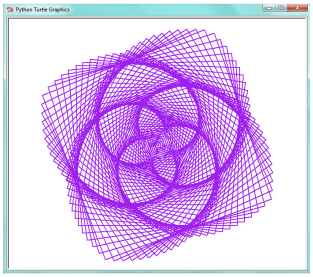
```
for _ in range(5):
    turtle.forward(400)
    turtle.right(144)
```

You can use an '_' instead of a variable here because the items (i.e. the numbers) are not referred to anywhere inside the loop



Spiral Patterns Created Using Turtle

- In the following two examples patterns are created using for loops with some cleverly chosen numbers

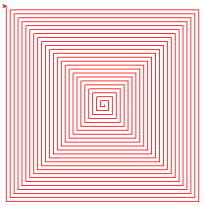



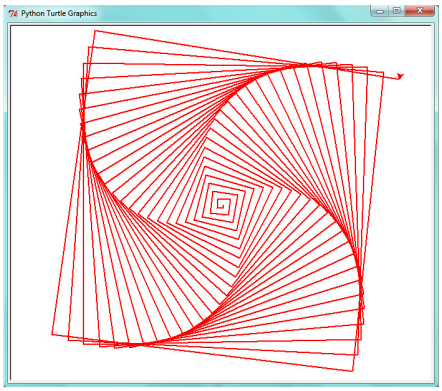
Spiral Pattern 1

```
for i in range(0, 500, 5):
    turtle.forward(i)
    turtle.right(91)
```

Run 100 times, where i = 0, 5, …, 495

Turning by 91 degrees creates a kind of spiral pattern whereas turning by 90 degrees will produce this:





Spiral Pattern 2

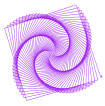
```
for i in range(0, 400, 2):
    turtle.forward(i)
    turtle.right(89)
```

Run 200 times, where i = 0, 2, …, 398

```
for i in range(401, 0, -2):
    turtle.forward(i)
    turtle.right(89)
```

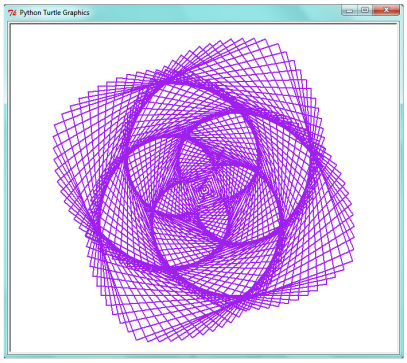
Run 201 times

The first loop makes this:



+ =

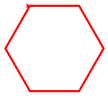
The second loop makes this:



Drawing a 'Flower' Using a Nested Loop

- In this example, a nested for loop (a for loop inside another for loop) is used to draw a flower
- The inner loop draws a hexagon and the outer loop uses the inner loop ten times to draw the flower:

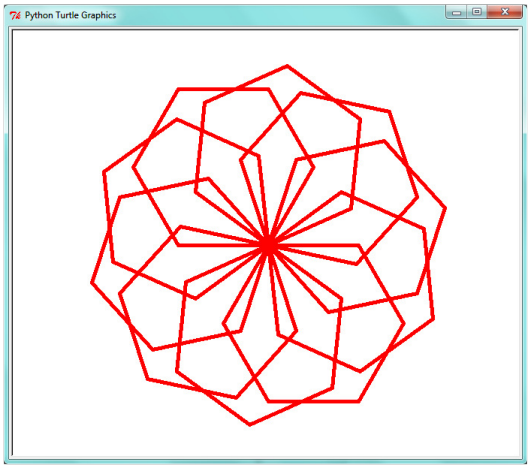
Draw a single hexagon using the inner loop



```
for _ in range(10):
    for _ in range(6):
        turtle.forward(120)
        turtle.right(60)
    turtle.right(36)
```

The outer loop draws hexagons around one full circle (10 * 36 = 360)

The Flower Pattern Created By Hexagons



Drawing a Pyramid of Dots

- In this example, a nested loop draws a pyramid of turtle dots using `turtle.dot()`
- The code is shown below:

```
size = 20
```

Create a single row of dots in the inner loop, e.g.:

```
for i in range(0, 15, 2):
    for j in range(i + 1):
        turtle.dot(size)
        turtle.forward(size)
```

Move the turtle to the starting point of the next row

```
turtle.backward(size * (i + 2))
turtle.right(90)
turtle.forward(size)
turtle.left(90)
```

Drawing the Rows of Dots

```
for i in range(0, 15, 2):
    for j in range( i + 1 ):
        ...
```

- As you can see from the loops, the inner loop runs a number of times based on the value of the outer loop
  - The first time the inner loop runs, it draws 1 dot
  - The second time it runs, it draws 3 dots
    …
  - The last time it runs, it draws 15 dots

turtle.dot() and turtle.up()

- You have learned that the turtle does not draw lines when you run `turtle.up()` before you move the turtle
- However, `turtle.dot()` is not affected by `turtle.up()` or `turtle.down()`
- In our example, `turtle.up()` has been used at the start of the program but the dots can still be drawn

```
import turtle

turtle.color("brown")
turtle.speed(0)
turtle.up()
turtle.hideturtle()
...
```

A Pyramid of Dots