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. Use for loops to create shapes with turtle graphics programming

For Loops in Turtle Graphics

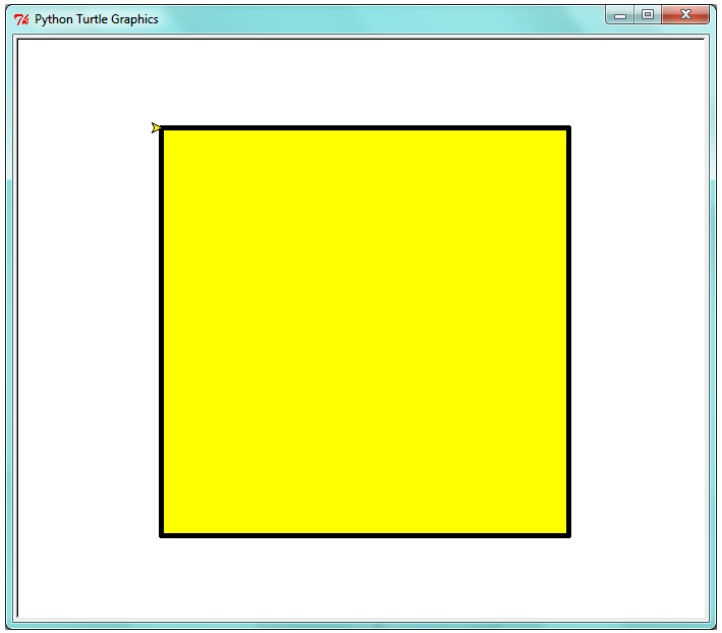
- We have seen examples of using while loops with turtle graphics
- Let's look at the use of for loops with turtle in this presentation
- Every example discussed in this presentation could also be made using while loops, if you wanted to do that
- There isn't much difference using either of the two types of loop in these examples

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



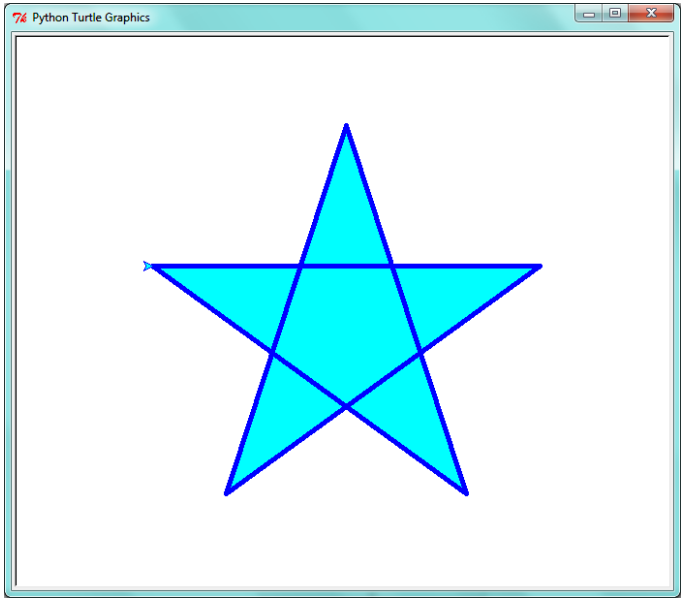
The letter 'i' is quite commonly used for the loop variable of a loop ('i' for 'index'), although you can use any variable name

Drawing a Star Shape Using a For Loop

- You can alter the program to draw a star shape
- The for loop runs five times instead of four times for the five lines of a star, like this:

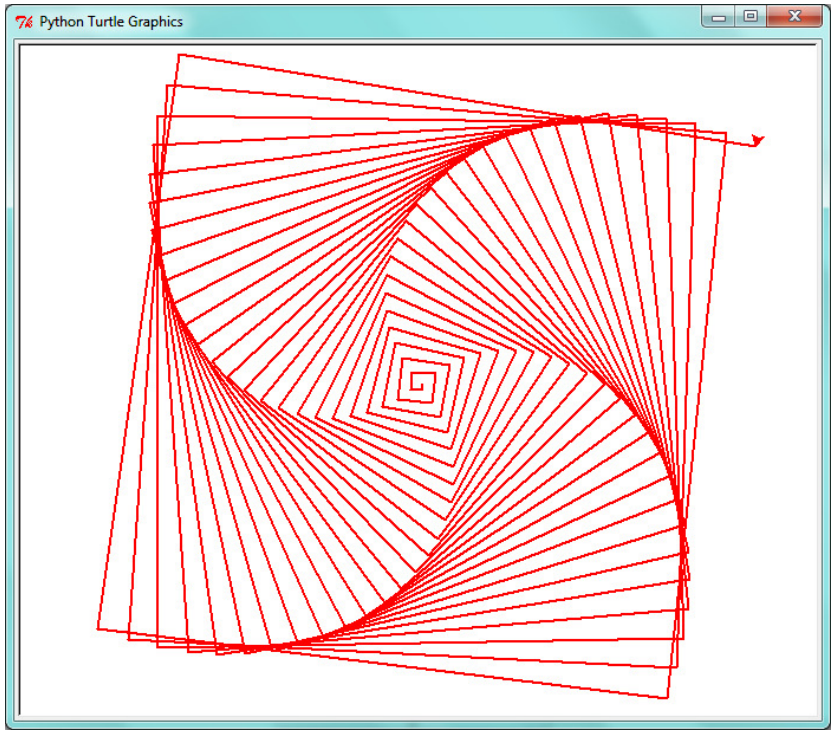
```
for _ in range(5):
    turtle.forward(400)
    turtle.right(144)
```

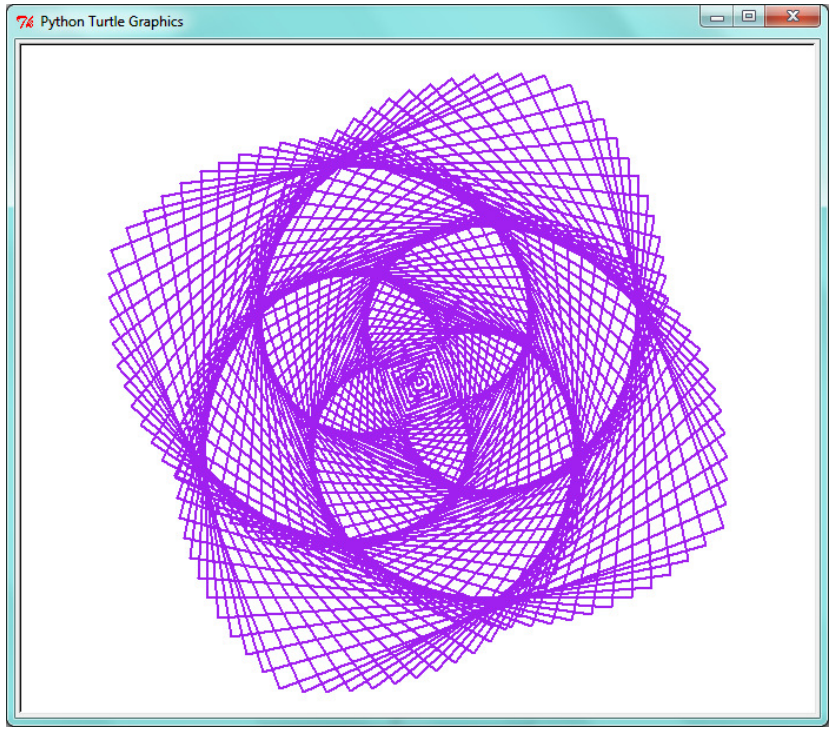
You can use an '_' instead of a variable here because the items (i.e. the numbers) are not referred to anywhere inside the loop



Spiral Patterns Created Using Turtle

- In the following two examples, patterns are created using for loops and some cleverly chosen numbers
- The turtle draws ‘squares’ using a slightly different angle and a slightly different length for each side





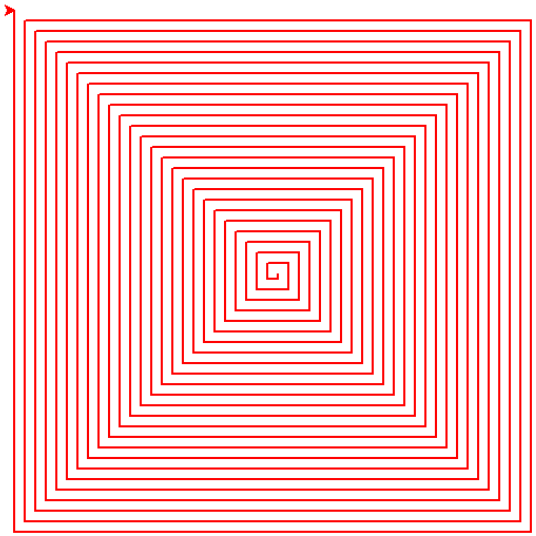
Spiral Pattern 1

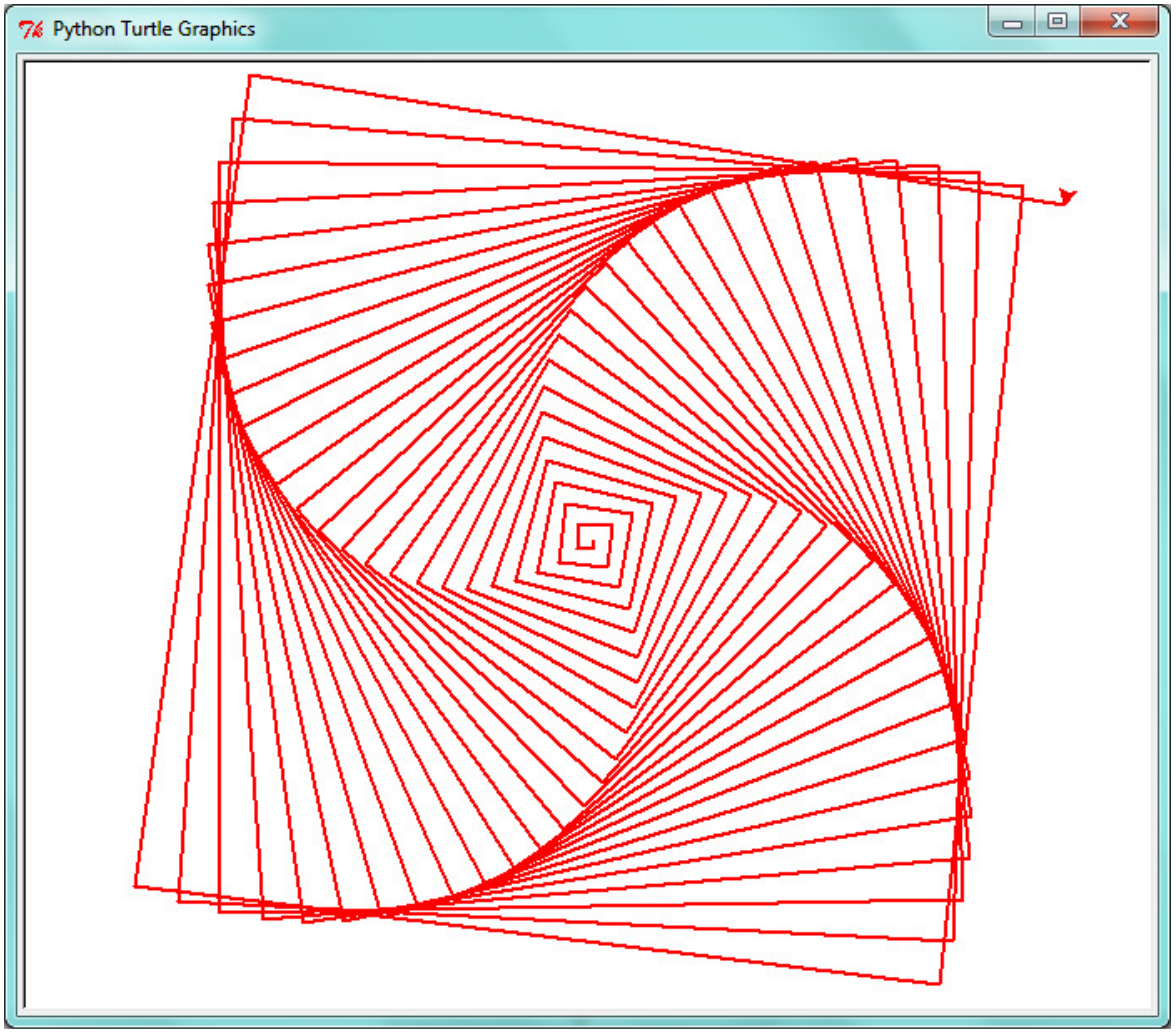
- This pattern is generated using this code:

```
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

- This pattern is generated using this code:

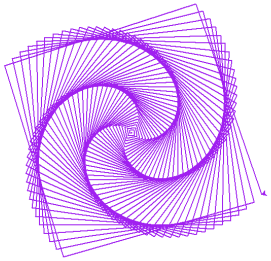
```
for i in range(0, 400, 2):
    turtle.forward(i)
    turtle.right(89)
```

Run 200 times, where i = 0, 2, …, 398

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

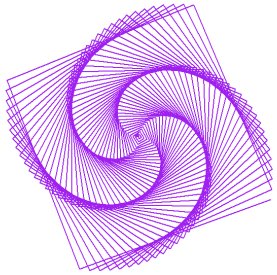
Run 201 times

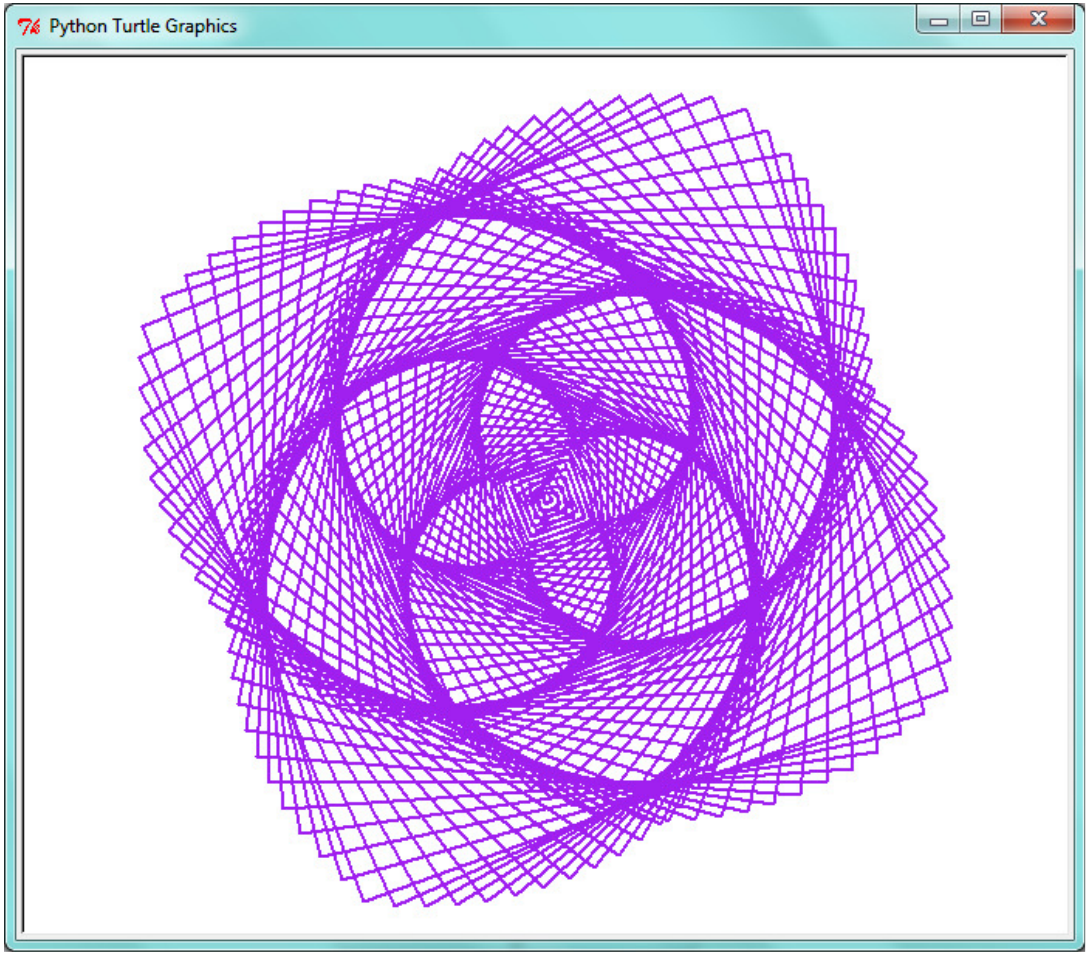
The first loop makes this:



+ =

The second loop makes this:

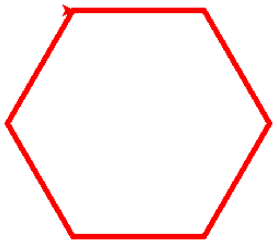




Drawing a ‘Flower’ Using a Nested Loop

- 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



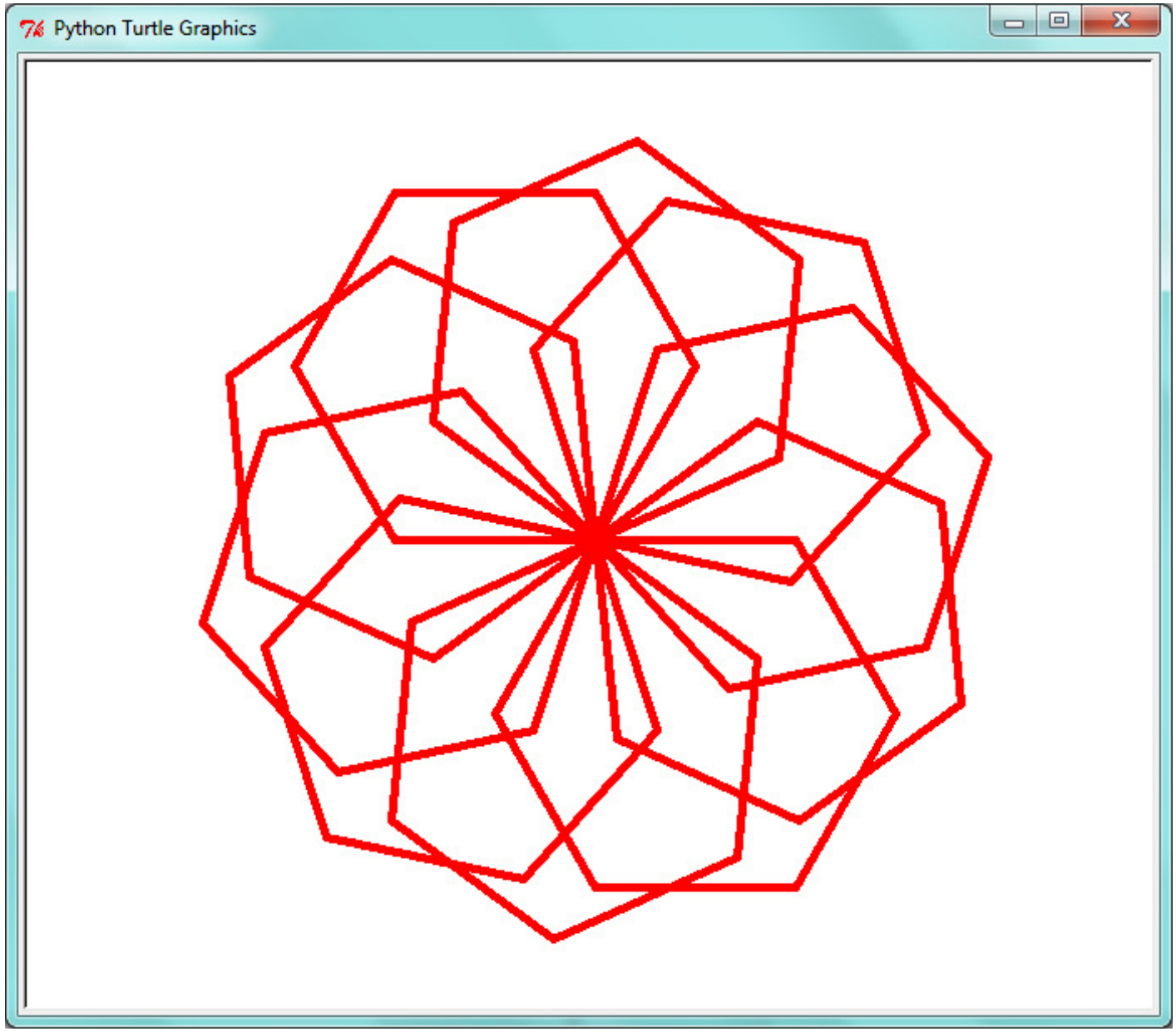
```
for _ in range(10):

    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

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

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

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

Move the turtle to the starting point of the next row

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