COMP1021
Introduction to Computer Science

Using Loops with Turtle Graphics

Gibson Lam and David Rossiter

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Use while loops to create shapes with turtle graphics programming

While Loops in Turtle Graphics

- Let's look at some examples which use loops in turtle graphics programming
- Loops are very useful in turtle graphics since you often need to use a turtle to do things repeatedly
- For example, to draw a square with a length of 10 you need to ask the turtle to move forward and turn four times, as shown on the right:

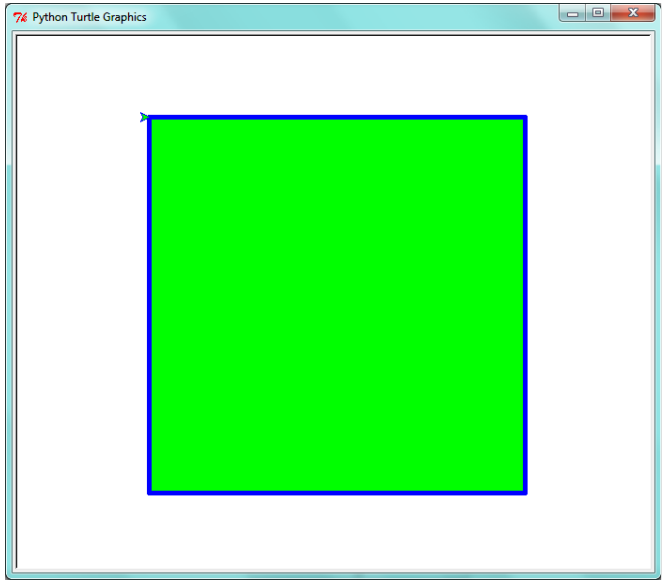
```
turtle.forward(10)
turtle.right(90)
turtle.forward(10)
turtle.right(90)
turtle.forward(10)
turtle.right(90)
turtle.forward(10)
turtle.right(90)
```

Drawing a Square Using a Loop

- You can use a while loop to draw a square, using the code shown below:

```
side = 0
while side < 4:
    turtle.forward(400)
    turtle.right(90)

    side = side + 1
```

Run the loop four times, i.e. the loop will be executed with these values, `side` = 0, 1, 2, 3

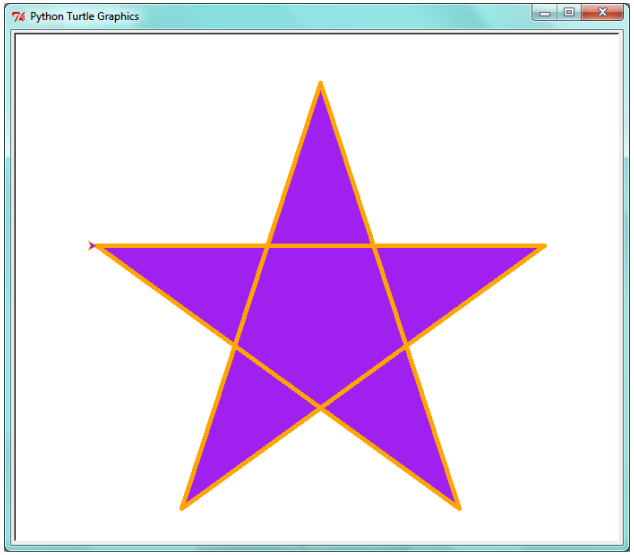


Drawing a Star Shape

- The example on the previous slide uses a while loop to draw four sides
- Similarly you can use a loop to draw a star shape with five sides, i.e.:

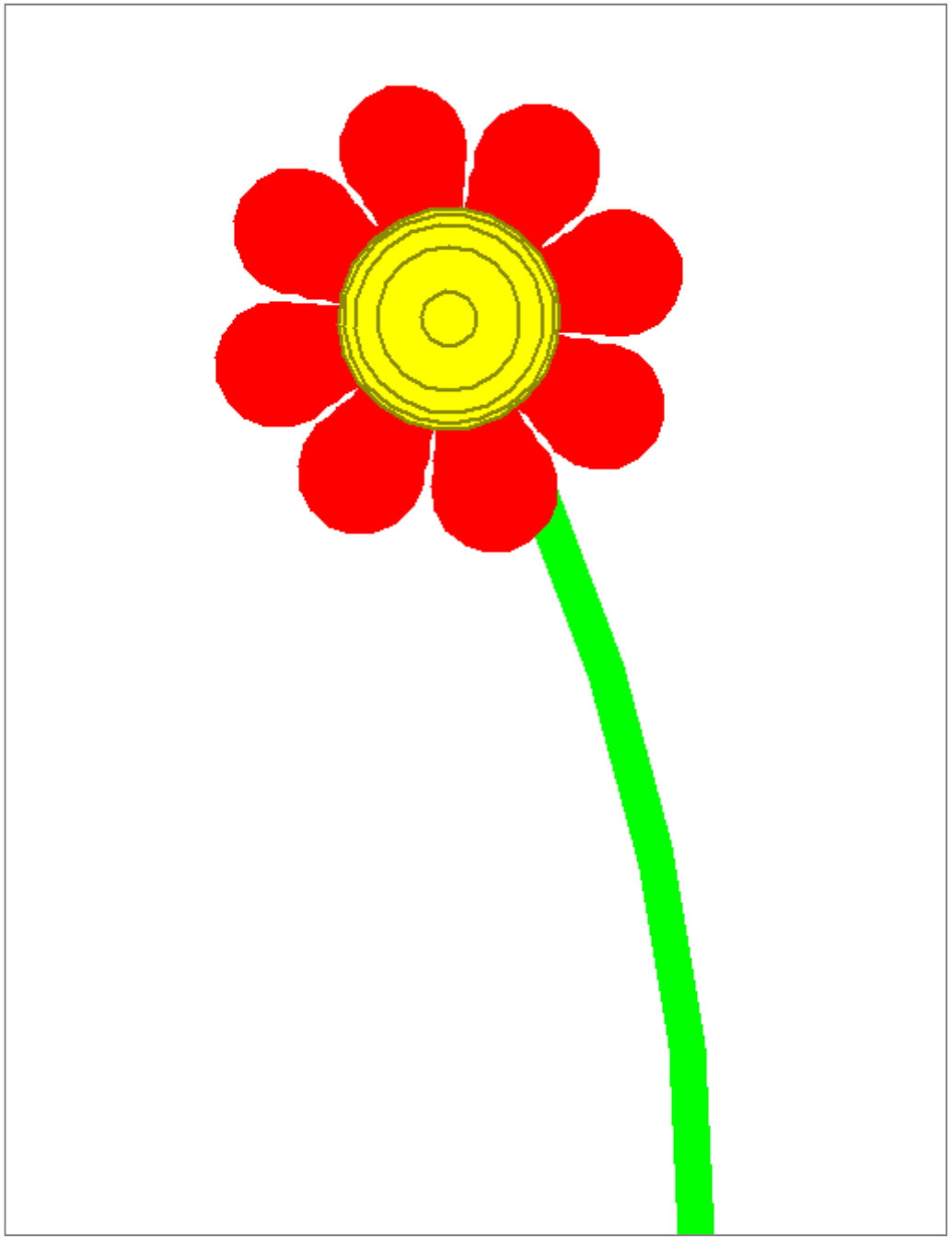
Run the loop five times, i.e. `side` = 0, 1, 2, 3, 4

```
side = 0
while side < 5:
    turtle.forward(400)
    turtle.right(144)

    side = side + 1
```



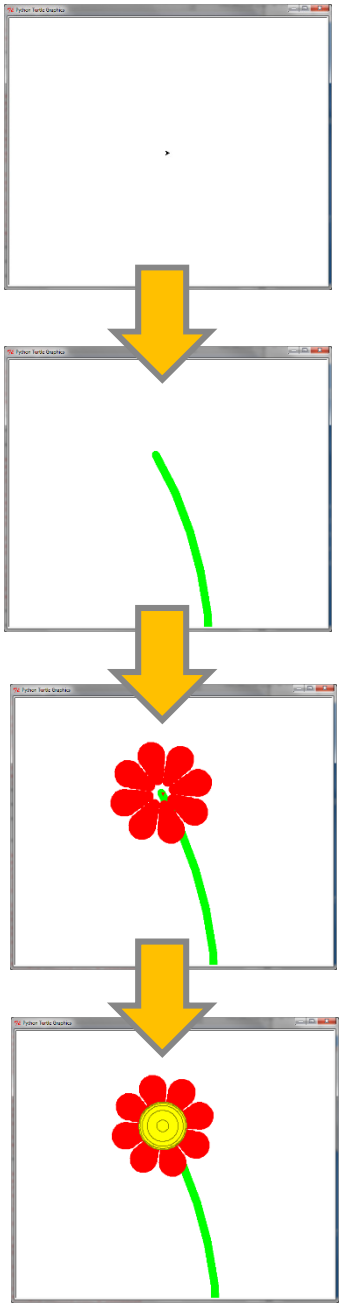
Using Nested Loops

- On the right is a flower image created by a single program
- It's a good example of using loops, including nested loops



The Program Stages

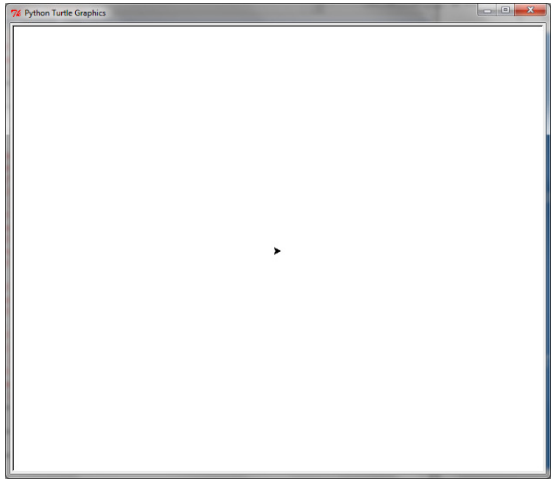
- Stage 0: get the turtle graphics started
  - import the turtle module, open the window
- Stage 1: create the curved stem
  - part of a circle
- Stage 2: draw the petals
  - uses a nested loop
- Stage 3: draw the central part
  - uses a loop



Stage 0 – Get the Graphics Started

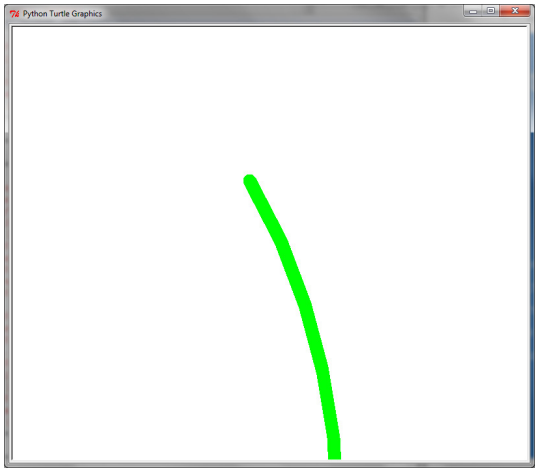
- Like many of the programs we have seen, the first step is to import the turtle module and set some initial parameters i.e.:

```
import turtle

turtle.speed(0)
```



Stage 1 – Create the Curved Stem

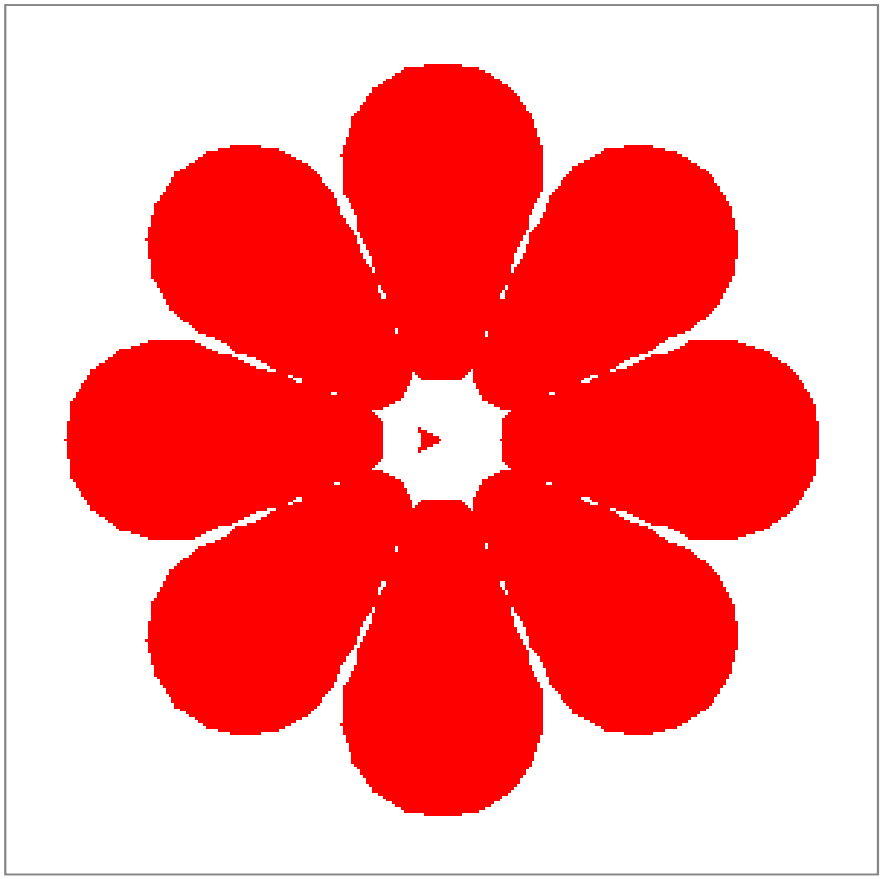
- We can create the stem of the flower using the turtle.circle() command i.e.




```
turtle.width(20)
turtle.color("green")

turtle.up()                 # Don't draw while we move
turtle.goto(100, -400)      # Move the turtle to bottom right
turtle.left(90)             # Point the turtle upwards
turtle.down()               # Start drawing from now onwards
turtle.circle(1000, 30)     # Draw circle with radius 1000
```

Stage 2 – Drawing a Flower Using Nested Loops

- We use a nested loop to draw the eight red petals of the flower, shown here:



- The inner while loop draws one petal using five circles

- The outer while loop uses the inner while loop to draw eight red petals

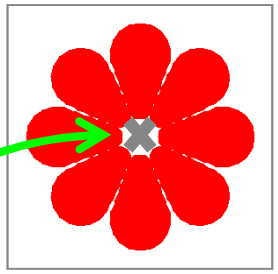
An Example of a Nested Loop Structure

```
while . . .condition . . . :
        . . .statement(s) . . .
        while . . .condition . . . :
                . . .statement(s) . . .
```

- As you already know, a loop inside another loop is called a nested loop
- It doesn't matter what type of loop it is; any type of loop inside any type of loop is called a nested loop
- So far we know about *while* loops, soon we will see another type of loop

Designing the Code

- Let's look at how we can draw the flower using two loops
  - **Outer loop:** repeat eight times for drawing eight petals
    - Move to the drawing position of the petal
    - **Inner loop:** repeat five times for drawing five circles
      - Move to the drawing position of the circle
      - Draw a circle of the appropriate size (the circles get larger each time)
      - Return to the starting position of the petal
    - Return to the centre position of the flower
    - Rotate the turtle by 45 degrees to draw the next petal

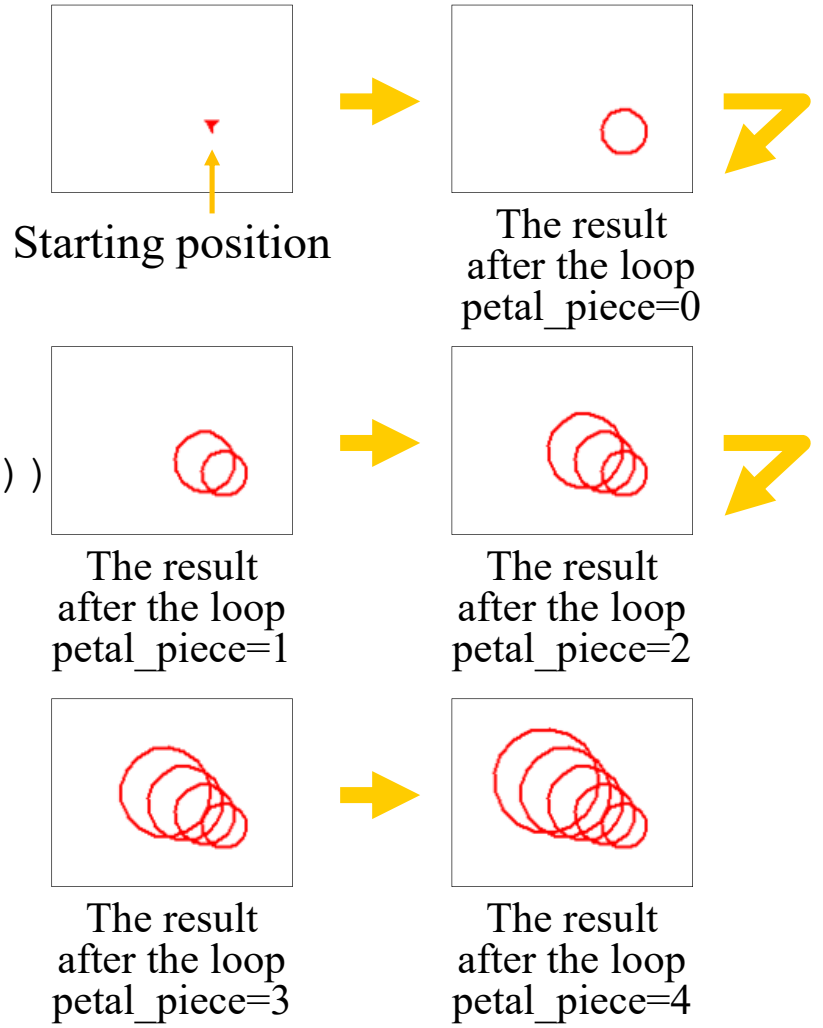


- We will show the main code in the following slides

The Inner Loop – Drawing a Petal

```
petal_piece = 0

while petal_piece < 5:
    turtle.up()
    turtle.forward(petal_piece * 20)
    turtle.left(90)
    turtle.down()

    turtle.begin_fill()
    turtle.circle(15 + (petal_piece*5))
    turtle.end_fill()

    turtle.up()
    turtle.right(90)
    turtle.backward(petal_piece * 20)
    turtle.down()

    petal_piece = petal_piece + 1
```

- This is the inner loop for drawing one petal, using 5 circles of different sizes
- Below we use unfilled circles to illustrate the process:

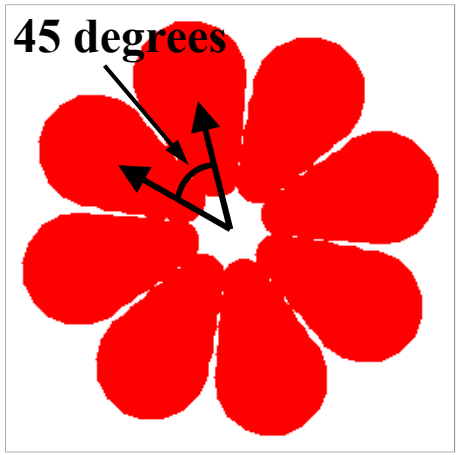


The Outer Loop – Drawing the Flower

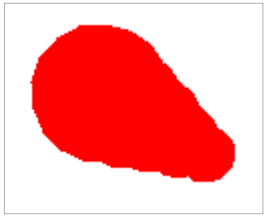
```
petal = 0

while petal < 8:
    turtle.up()
    turtle.forward(50)
    turtle.down()

    petal_piece = 0
    while petal_piece < 5:
        ...
        petal_piece
            = petal_piece + 1

    turtle.up()
    turtle.backward(50)
    turtle.down()

    turtle.left(45)
    petal = petal + 1
```

The outer while loop uses the inner while loop 8 times, for 8 petals

Each petal has a different starting point, 50 pixels away from the center



The inner while loop draws five circles of a petal (this part is shown in the previous slide)



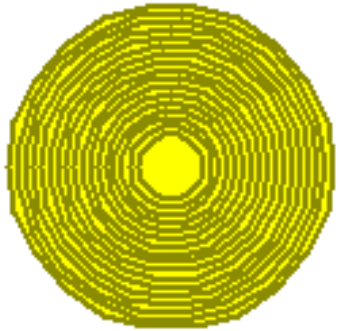
Here we go back to the starting point, the center

Each petal is drawn using a different angle by turning the turtle 45 degrees

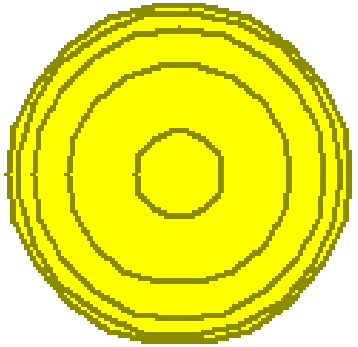
Move on to the next petal

Stage 3 - Drawing the Flower Centre

- The middle of the flower is drawn using a while loop, which draws a series of circles
- We first draw the largest circle, then the smaller ones
- We reduce the radius each time
- If we reduced it by 3 pixels each time, we would get the top image:
- However, let's make a more interesting pattern like the bottom:
- We do that by increasing the size of the reduction for each successive circle



The reduction in radius each time is 3



The reduction in radius is 3, 6, 12, 24

Drawing the Flower Centre

```
radius = 60
radius_reduction = 3

while radius > 10:
    turtle.up()
    turtle.forward(radius)
    turtle.left(90)
    turtle.down()

    turtle.begin_fill()
    turtle.circle(radius)
    turtle.end_fill()

    turtle.up()
    turtle.right(90)
    turtle.backward(radius)
    turtle.down()

    radius = radius - radius_reduction
    radius_reduction = radius_reduction * 2
```

The current turtle position is the center of the circle, so we temporarily move the turtle to the correct position (the side of the circle) before making the circle

Then we draw the yellow circle that we want

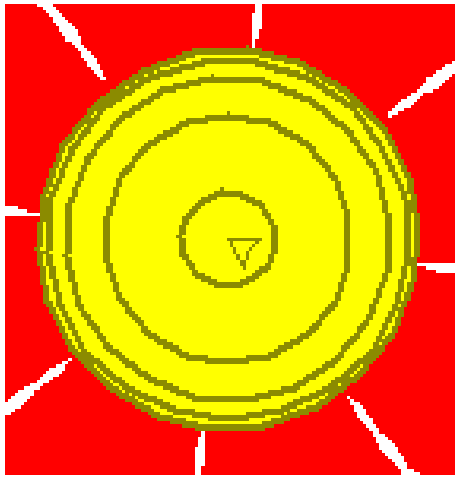
The we move the turtle back to the center position, ready for the next loop

We make the next circle smaller

Next time, the decrease in radius will be greater

Hiding the Turtle

- Sometimes the pictures in the turtle window will look better if the turtle is not visible inside the window
- In our previous flower example, the turtle is in the middle of the flower when the drawing finishes
- We hide the turtle near the end of the program using this line of code:



```
turtle.hideturtle()
```

- After running the code, the turtle is then hidden in the final result