technocamps

Inspiring | Creative | Fun
Ysbrydoledig | Creadigol | Hwyl

UNDEB EWROPEAIDD
EUROPEAN UNION

Llywodraeth Cymru
Welsh Government

Cronfa Gymdeithasol Ewrop
European Social Fund

Prifysgol Abertawe
Swansea University

CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

PRIFYSGOL BANGOR UNIVERSITY

Cardiff Metropolitan University | Prifysgol Metropolitan Caerdydd

University of South Wales
Prifysgol De Cymru

i.t.wales

PRIFYSGOL ABERYSTWYTH UNIVERSITY

PRIFYSGOL glyndŵr Wrecsam | Wrexham glyndŵr UNIVERSITY

technocamps

Introduction to Python Programming

Activity: Draw the Image

1. Draw a square.

2. Inside that square draw another 2 smaller squares.

3. Draw a triangle attached to the big square.

4. Finally inside the larger square draw a rectangle that touches the square.

What Is Programming?

Programming is telling the computer what to do using a set of ordered instructions.

The set of ordered instructions is called an **algorithm.**

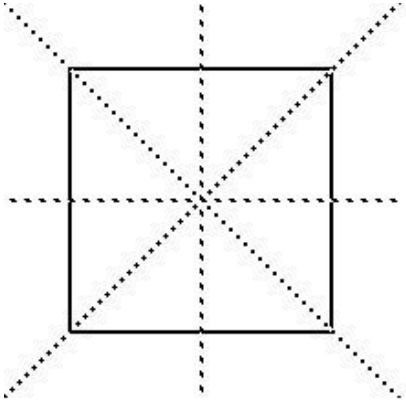
The language used to tell the computer what to do is called a **programming language**.

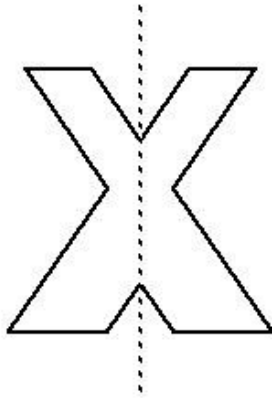
Lines of Symmetry

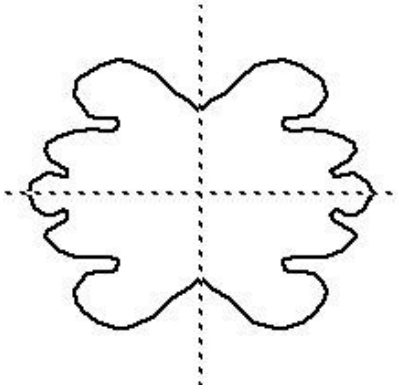
Lines of Symmetry

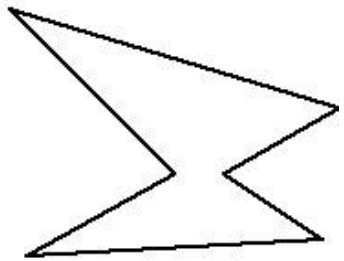
A 2D shape is symmetrical if a line can be drawn through it so that either side of the line looks exactly the same.

The line is called a line of symmetry.





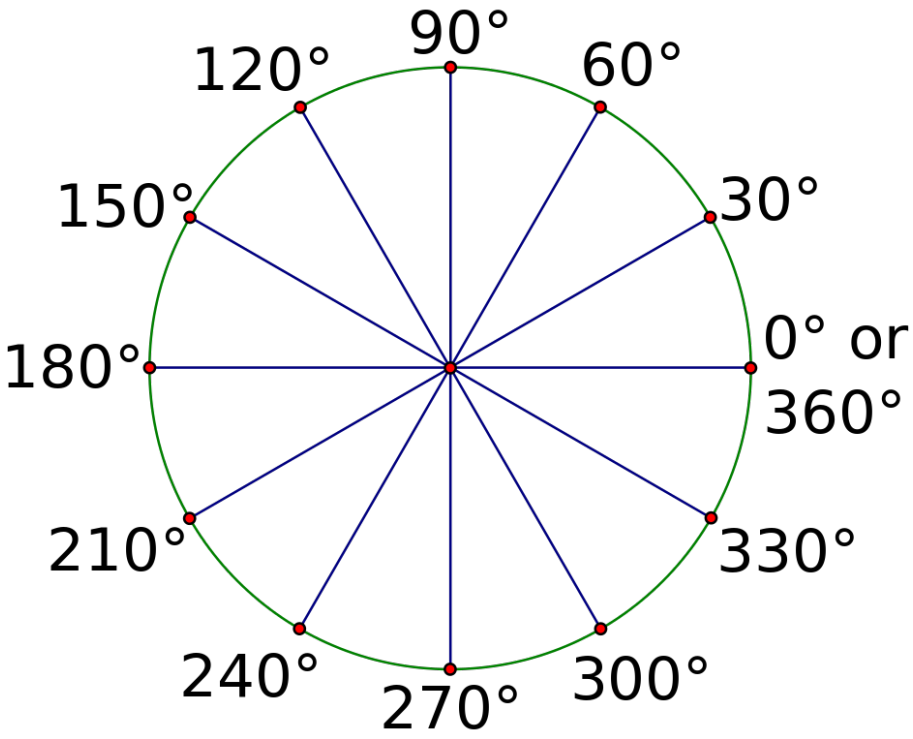




Angles

Angles

An angle is the **amount of turn** between two lines around their common point (the vertex).



Activity: Geometry

Identify the 2D shapes from the given properties

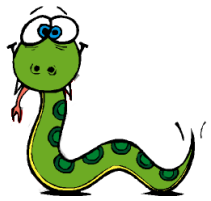
a) 2 sets of equal sides b) 4 equal angles (90°) c) 2 lines of symmetry
a) 4 equal sides b) 4 equal angles (90°) c) 4 lines of symmetry
a) 3 equal sides b) 3 equal angles (60°)

Geometry in Programming

- We already know that programming is telling the computer what to do.
- We are going to tell the computer to draw us some 2D geometry shapes.
- Computers will only **do exactly what we instruct it to do.**
- Clear instructions are essential.

Activity: What Is Python?

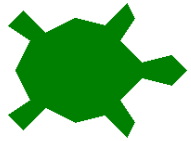
Python



- It is neither a snake nor parseltongue from Harry Potter.
- It is a programming language which tells the computer what to do using algorithms.
- It is free.
- It is easy to learn, read and code.
- It is interactive and portable.
- It is high level and flexible.

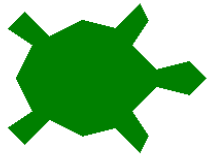
The author of the **Python** programming language is a Dutch man called **Guido van Rossum**, named it after his favourite TV series **Monty Python's Flying Circus**

Turtle Graphics in Python



- We will learn Python using Turtle.
- Assuming that a turtle can understand simple commands, how can we make the turtle create some fun mathematical graphics (like geometrical shapes) using these commands?
- Python has many libraries (set of commands).
- Today, we will use Turtle Graphics to create 2D shapes.

Activity: Draw a Square



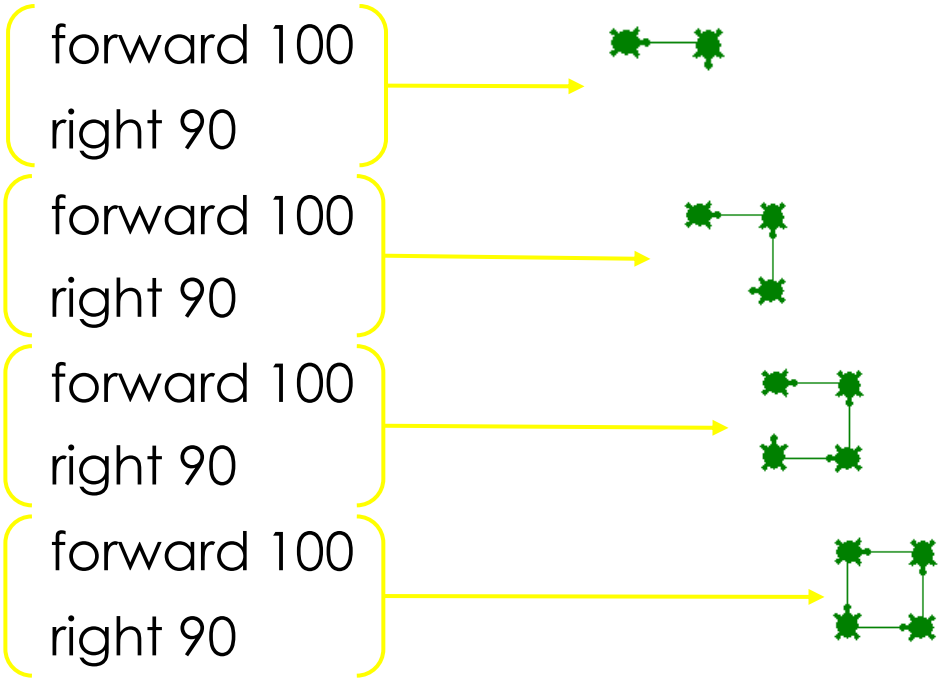
Let us assume that you are the turtle. You only understand two instructions.

a) **forward x:** You can only go forward x steps e.g. forward 100 (move 100 steps forward).

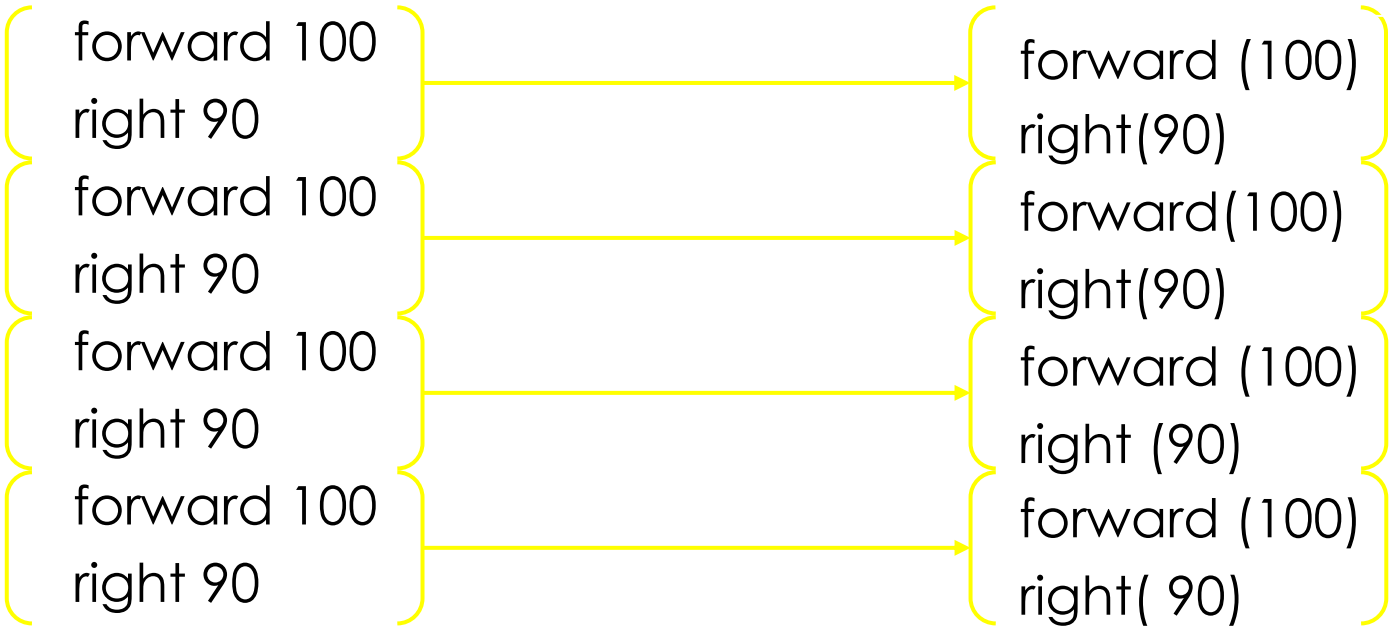
b) **right y:** You can only turn right (clockwise) at the given y angle e.g. right 90 (turn clockwise 90 degrees).

Write the instructions to draw a square using only those two instructions.

Square - Instructions



Square - The Turtle Way



First Turtle Program

```
import turtle
```

Get the turtle commands

First Turtle Program

```
import turtle
pen = turtle.Turtle()
```

Get the turtle commands

Create a new Turtle

First Turtle Program

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
```

Get the turtle commands

Create a new Turtle

Set the shape of the turtle

First Turtle Program

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.forward(100)
```

Get the turtle commands

Create a new Turtle

Set the shape of the turtle

Move the turtle forward by 100 steps

First Turtle Program

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.forward(100)
pen.right(90)
```

Get the turtle commands

Create a new Turtle

Set the shape of the turtle

Move the turtle forward by 100 steps

Turn right at 90 degrees

First Turtle Program

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
```

Get the turtle commands

Create a new Turtle

Set the shape of the turtle

Move the turtle forward by 100 steps

Turn right at 90 degrees

Do it 3 more times

Activity: First Turtle Program

- Open IDLE.
- Write a program to draw a square and save it as **square.py**.
- Write a program to draw a rectangle and save it as **rectangle.py**.

Turtle Commands

Commands	Description	Example
left(x)	Turns anticlockwise x degrees in angle.	`left(180)`
backward(x)	Moves backward x number of steps.	`backward(100)`
color("name")	Sets the colour of the turtle's pen.	`color("blue")`
fillcolor("name")	Sets the colour to fill the shape drawn by the turtle.	`fillcolor("red")`
begin_fill()	Marks the area to fill the colour set in fillcolor().	
end_fill()	Marks the end of the area to fill the colour set in fillcolor().	

Square with Colours

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.color("yellow")
pen.fillcolor("green")
pen.begin_fill()
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
pen.end_fill()
```

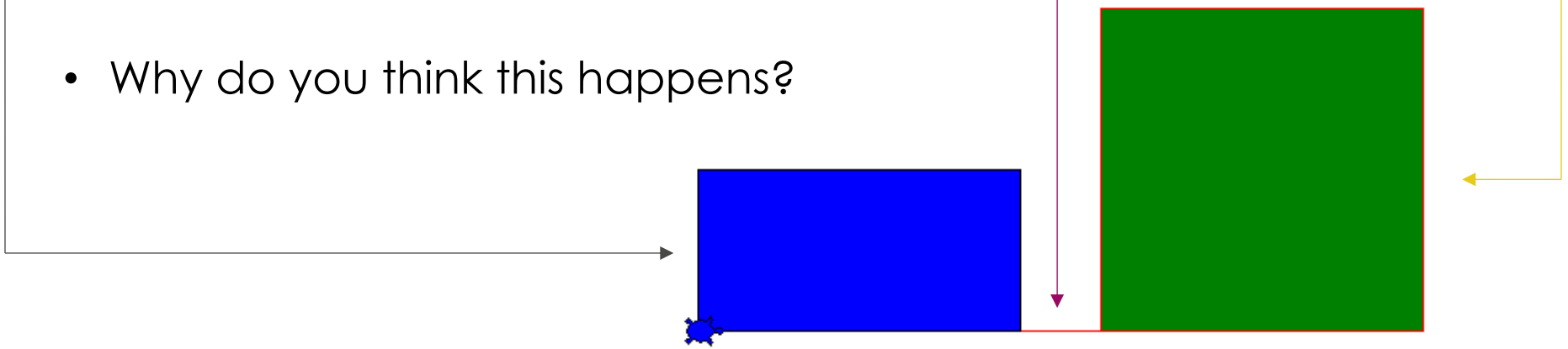
Activity: Colour the Shapes

Modify the SQUARE program to draw the square with red lines and fill it in green.

Modify the RECTANGLE program to draw the rectangle with orange lines and fill it in purple.

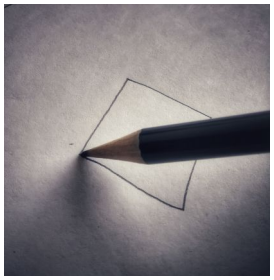
Pen Commands

- I want to draw a Square with red line and fill it with green.
- I want to draw a Rectangle next to my square with black line and fill it with blue.
- When I run the code the resulting graphic has a red line between them.
- Why do you think this happens?

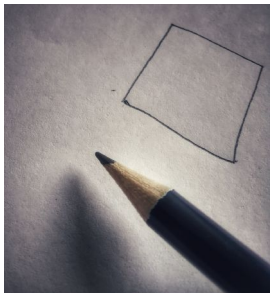


Penup and Pendown

When the turtle is drawing something the **pen** is **down.**



If we do not want to draw something then we need to have the **pen** in the **up** position.



Penup and Pendown

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
#set the line colour
to red
pen.color("red")
#draw the triangle
pen.forward(100)
pen.right(120)
pen.forward(100)
pen.right(120)
pen.forward(100)
pen.right(120)
#lift the pen up
pen.penup()
#move 150 steps
forward
pen.forward(150)
#put the pen down
pen.pendown()
#change the line
colour to green
pen.color("green")
#draw another triangle
pen.forward(100)
pen.right(120)
pen.forward(100)
pen.right(120)
pen.forward(100)
pen.right(120)
```

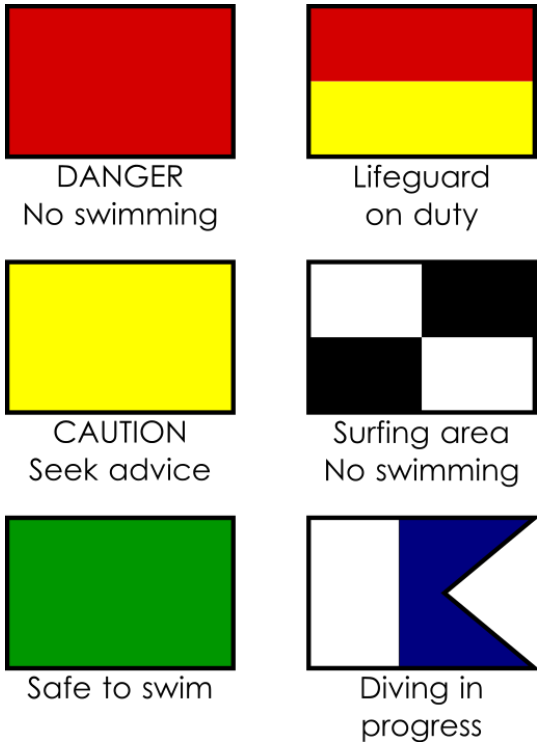
Comments start with **#.** Python ignores them.

Activity: Pen Exercises

Draw two shapes of your choice (square/rectangle/triangle) next to each other with some space between them.

Activity: Fun Flags

Can you draw one or more of these lifeguard flags or a flag of the world using Turtle Graphics?



So Far...

- Python is a **programming language.**
- Turtle Graphics is a library in Python used for graphics.
- Simple commands like forward(), backward(), left(), right() etc are used to create shapes.
- Commands like penup(), pendown(), fillcolor(), color(), begin_fill(), end_fill() etc are used to fill the shapes with amazing colours.

Repeating Instructions

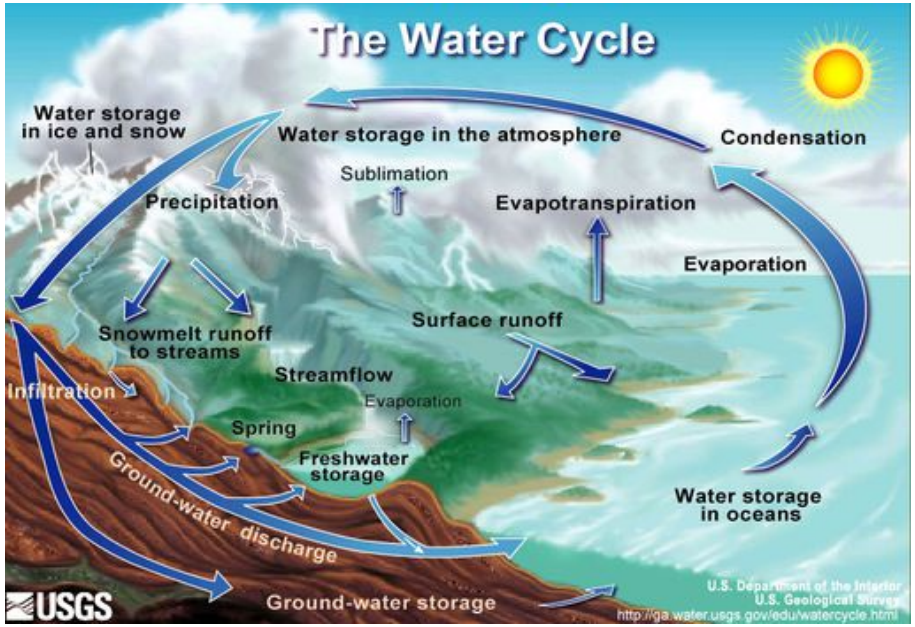
```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.color("green")
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
```

Let us review our first SQUARE program again.

There are some commands which are **repeated** more than once.

For example the lines in green are repeated **4** times.

Iteration – A Vicious Cycle



- Iteration is the process where the steps are repeated more than once.
- The process is also called ***looping***.

For Loops

A **For Loop** is a statement in Python that allows you to repeat a set of commands (or instructions).

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.color("green")
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
```

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.color("green")
for i in range(4):
    pen.forward(100)
    pen.right(90)
```

Dissecting the Loop

Let us try and understand the loop a little bit better:

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

`i`	**Loop variable**	A counter to keep track of the number of times the loop body is run.
`range(4)`	**Range**	The number of times the loop body has to be repeated.
`pen.forward(100)` `pen.right(90)`	**Loop Body**	The commands to be repeated.

Indentation in Loops

It is important to indent the code for loops.

Indentation is used to tell Python that the code belongs to the loop.

It helps the code look neat and clean.

It also makes it more readable.

```
import turtle

pen = turtle.Turtle()

pen.shape("turtle")

pen.color("green")

for i in range(4):

    pen.forward(100)

    pen.right(90)
```

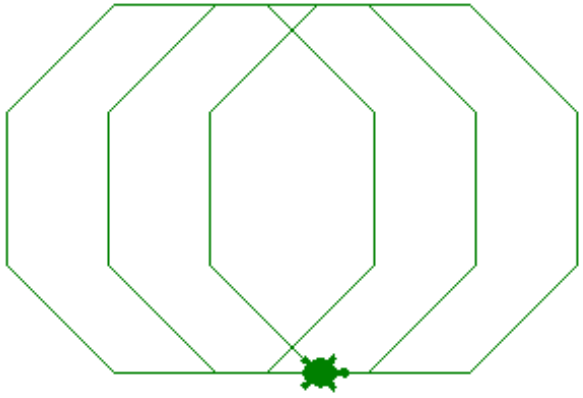
Activity: Let's Loop

- You are given a couple of programs where the commands are repeated.
- Can you change those programs to use loops and identify the shapes?
- Don't forget the indentation.

The Three Octagons

We can combine the Pen commands (penup() and pendown()) to produce some very cool graphics.

```
#Three Octagons
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.color("green")
#draw the first octagon
for i in range(8):
    pen.forward(75)
    pen.left(360/8)
#lift the pen up
pen.penup()
#move to a different
#location
pen.forward(50)
#put the pen down
pen.pendown()
#draw the second
octagon
for i in range(8):
    pen.forward(75)
    pen.left(360/8)
#lift the pen up
pen.penup()
#move to a different
# location
pen.forward(50)
#put the pen down
pen.pendown()
#draw the third octagon
for i in range(8):
    pen.forward(75)
    pen.left(360/8)
```



Activity: Loopy Designs

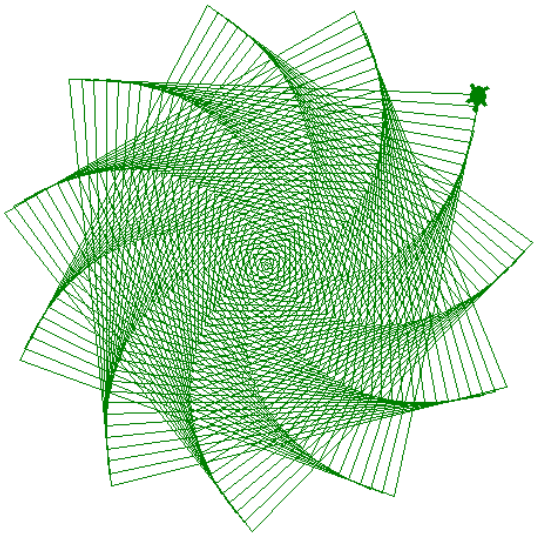
Use the loops and pen commands to create some funky designs.

Loop it with a Variable

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.color("green")
for i in range(350):
    pen.forward( i )
    pen.right(98)
```

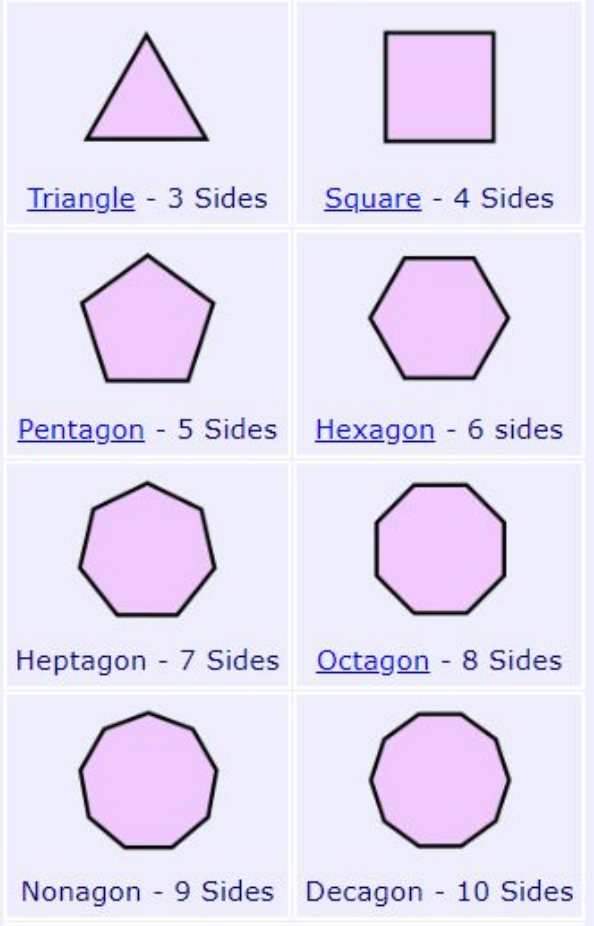
Remember, the loop variable is a counter

We can also use the loop variable (i) inside the design. The code on the left will produce the design below.



Extension Activity: Loop It with a Variable

2D Shapes



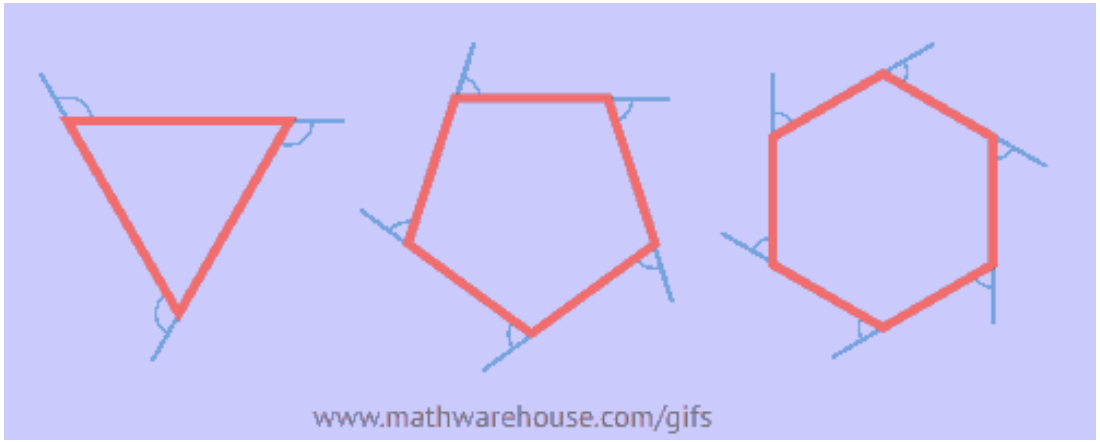
- Let us revise our 2D shapes
- We know from the picture that they all have a different number of sides
- What are the types of angles associated with 2D shapes?

Exterior Angles

Exterior angles are **outside** the polygon made by extending one of the sides.

Sum of all exterior angles = 360°

Why? If we keep moving around the polygon, extending the sides and measuring each exterior angle, we end up having a circle, which contains 360°.



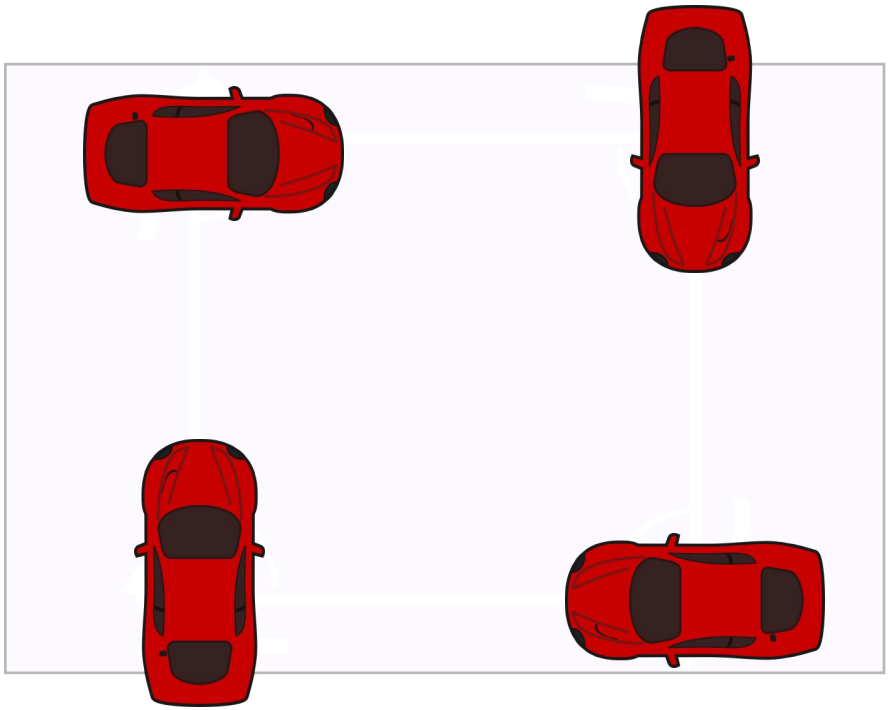
Size of Each Exterior Angle

All interior angles are equal for **regular** polygons, hence all exterior angles are equal too.

To work out **the size of each exterior angle** we use the formula:

360 ÷ number of sides

Applying Exterior Angle



In the whole motion we made 1 full rotation, turning 4 times. So the angle for each turn is

$$\frac{360°}{4} = 90°$$

Activity: Angles

Complete the table in your workbooks by filling in the number of sides, total sum of exterior angles and size of each exterior angle for the given regular polygons.

Angle Facts

We can calculate the exterior angle of any polygon using the formula: 360 ÷ number of sides

Degree	Angle	Shape
360 ÷ 3	120 °	Triangle
360 ÷ 4	90 °	Square / Rectangle
360 ÷ 5	72 °	Pentagon
360 ÷ 6	60 °	Hexagon
360 ÷ 7	51 °	Heptagon
360 ÷ 8	45 °	Octagon
360 ÷ 9	40 °	Nonagon
360 ÷ 10	36 °	Decagon

Can we rewrite the Turtle Program to create all these 2D shapes based on their number of sides?

Automate the Angles

- We need to ask the user **the number of sides** the shape they want to draw has.
- We then **store** their answer in a **variable**.
- **Use** that **variable** in the calculation of our exterior angle formula in the **loop body**.
- And also in the **range** part of the Loop.

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
sides = int(input("how many sides do you want?"))
pen.pendown()
for i in range(sides):
    pen.forward(100)
    pen.left(360/sides)
```

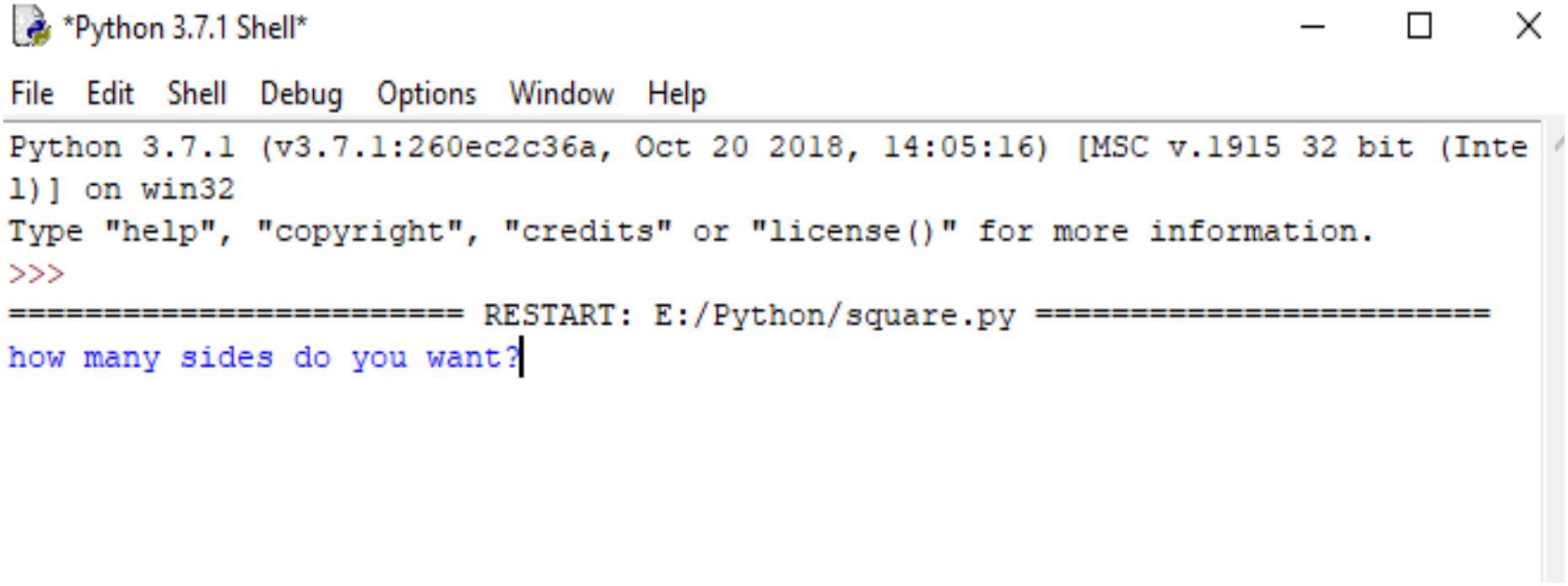
Input and Variables

```
sides = int(input("how many sides do you want? "))
```

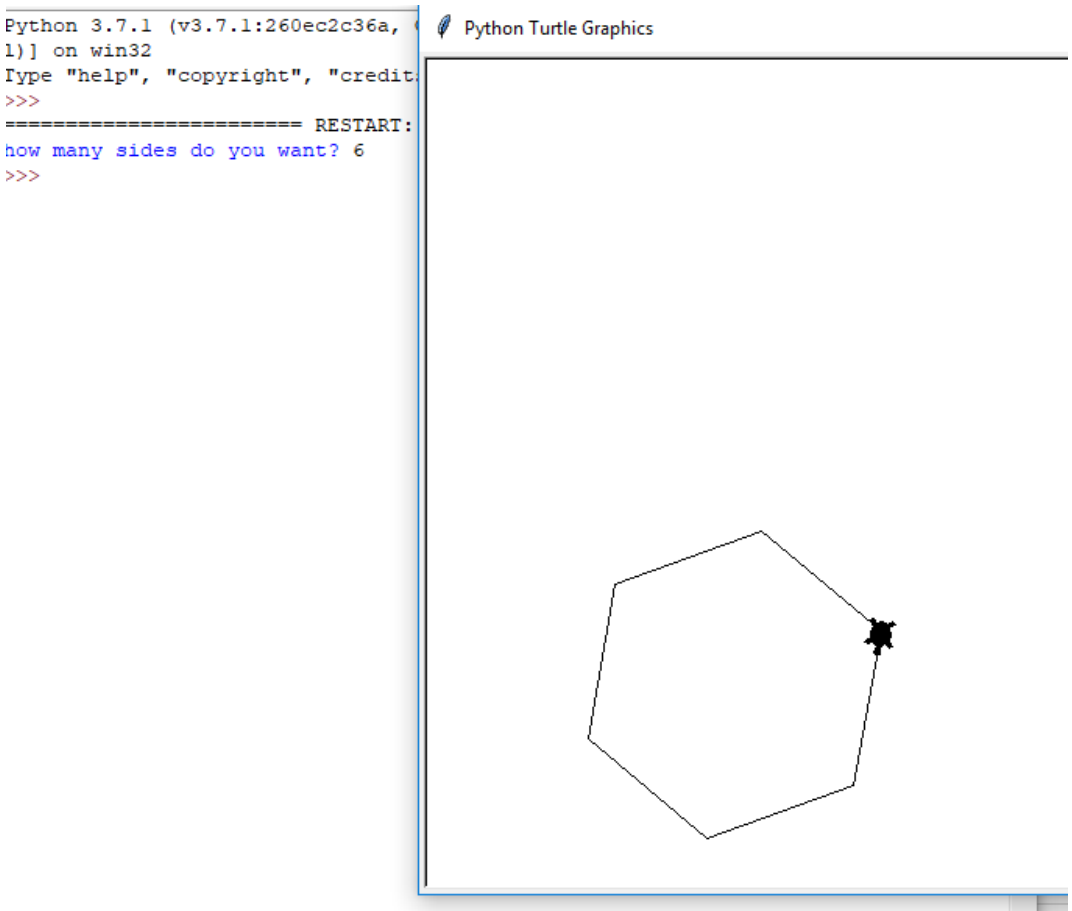
sides	The variable that will store the answer. **Variable is a name that stores the value of something.**
input	The command that will ask the user the question. The value will always be text.
int	The command to change from text to number.
how many sides do you want?	The question you want to ask the user.

How Does It Work?

When you run the code, it will **prompt** you on the screen to enter the number of sides.



Result



Once the number of sides is entered, the Turtle draws the shape for you.

Activity: Draw Any Shape

The code is given to you but it is not arranged in the correct order.

Based on what we have learned, rearrange the code to make it work.

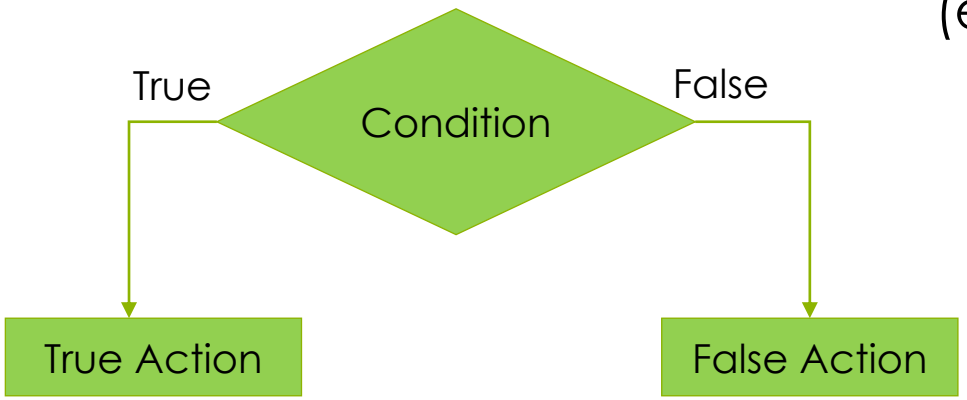
Extension Activity: Colour the Shape

- Ask the user which colour they would like to use to fill their shape.
- Store that in a variable called **usercolor**.
- Use the variable in fillcolor() command.

Conditionals

If it is raining outside, then take the umbrella.

If homework is completed then go and play Xbox, else sit and complete it.



These are some real life situations where we use conditions.

We take some action if the condition is true (e.g. do the homework), and another action if the condition is false (e.g. play Xbox).

There are only two outcome: True or False

Python Conditionals

if (condition):

➡ **then do these things**

else:

➡ **do these things**

In our example,

If sides is equal to 4:

    ask the user for the length of two sides and draw a square or a rectangle

Else:

    go ahead with the formula 360 ÷ sides

Conditionals

`if (sides==4):`	**Conditional Statement**. Note the : (colon) at the end and the == (equal to) sign to check if the variable "sides" is equal to 4.

`side1 = int(input("What is the length of side 1? "))`
`side2 = int(input("What is the length of side 2? "))`
`…….`
	Code that is executed if the condition is true. In other words, if the sides value is 4 then these lines of code will be run.

`else:`	**Conditional else.** Note the : (colon) at the end.

`for i in range(sides):`
`pen.forward(100)`
`pen.left(360/sides)`
	If the sides value is not equal to 4 then these lines of code will be run.

Using Conditionals

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
sides = int(input("how many sides do you want? "))
pen.pendown()
if (sides==4):
    side1 = int(input("What is the length of side 1? "))
    side2 = int(input("What is the length of side 2? "))
    for i in range(2):
        pen.forward(side1)
        pen.left(90)
        pen.forward(side2)
        pen.left(90)
else:
    for i in range(sides):
        pen.forward(100)
        pen.left(360/sides)
```

What would happen if side1 & side2 = 100

What would happen if side1 = 100 and side2 = 200?

Activity: Conditional Shapes

Reorder the automated shape program and complete the conditions we have just discussed.

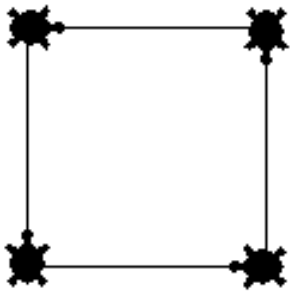
Stamp

```
import turtle

pen = turtle.Turtle()

pen.shape("turtle")

for i in range(4):

    pen.forward(100)

    pen.right(90)

    pen.stamp()
```

In our earlier scenarios, the turtles move from one location to another.

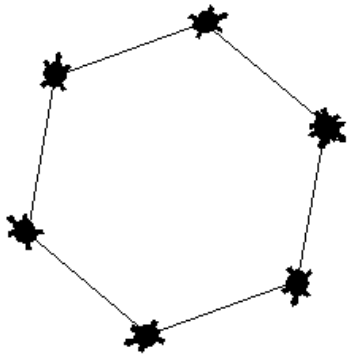
Stamping is the process of making an impression of the turtle on the screen.

This works even if the **pen is up.**



Activity: Stamp the Turtles

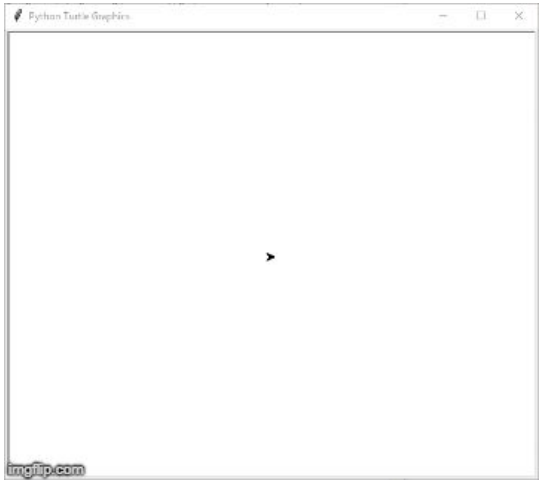
Using the commands create random shapes and stamp the turtle on the screen for every shape.



Activity: Turtle Review

Activity: The House

Use the concepts learned today with the help of Turtle library, draw a house we did as the first activity today.



Questions/ Feedback!