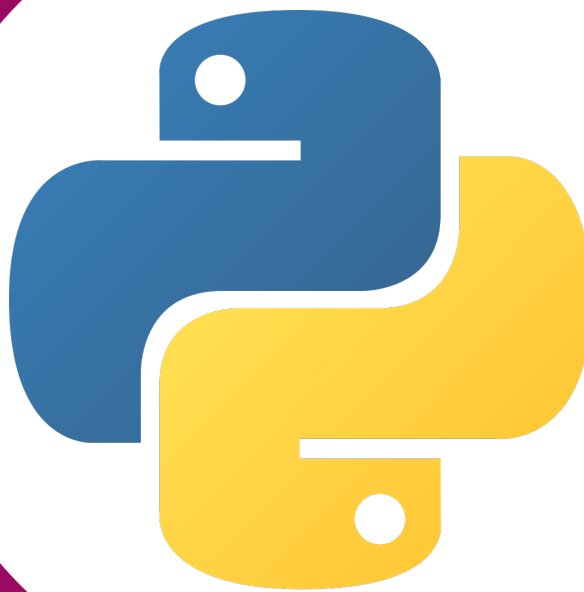
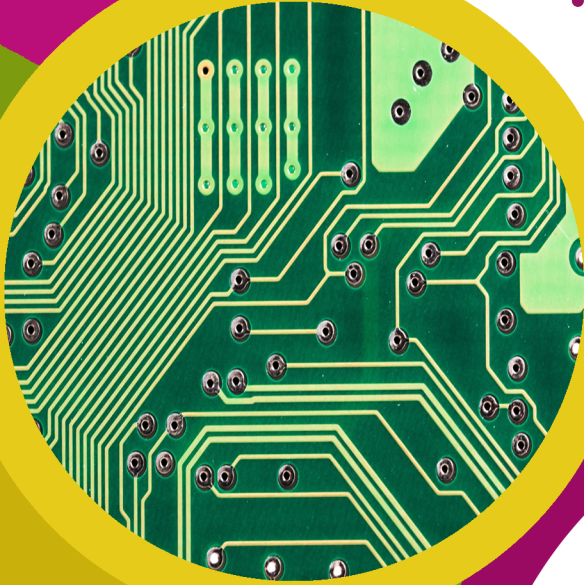


# technocamps

## Python Mathematics Workbook



## Links to Science and Technology AoLE

### Computing:

**(PS3)** I can use conditional statements to add control and decision-making to algorithms.

**(PS3)** I can identify repeating patterns and use loops to make my algorithms more concise.

**(PS3)** I can explain and debug algorithms.

**(PS2)** I can follow algorithms to determine their purpose and predict outcomes.

## Links to Other AoLEs

### Mathematics and Numeracy:

#### Geometry:

**(PS3)** I can explore and consolidate my understanding of the properties of two-dimensional shapes to include the number of sides and symmetry.

**(PS3)** I can demonstrate my understanding of angle as a measure of rotation and I can recognise, name and describe types of angles.

#### Algebra:

**(PS2)** I have explored patterns of numbers and shape. I can recognise, copy and generate sequences of numbers and visual patterns.

## The Four Purposes and Cross-Curricular Skills

Throughout this activity you will be able to use your **Critical Thinking and Problem Solving** skills. You will be able to recognise potential problems with algorithms which are not in the correct order and debug any errors you encounter when writing your programs.

You will have the opportunity to show your **Creativity and Innovation** when asked to write algorithms to produce your own shapes and patterns, as well as experiment with variables to produce unique and funky patterns.

You will also use skills covered in the **Data and Computational Thinking** strand of the **DCF** to understand the importance of the order of statements, when creating and refining your algorithms. You will be able to detect errors and use iteration to improve efficiency of your code.

## Why Is Learning This Important?

The world is becoming more and more digital and programming skills are now highly desirable in almost every industry imaginable. Using these skills in a creative and innovative way is important for producing digital media which is also becoming more and more popular. This resource will allow you to develop your understanding of a modern and popular programming language used by many big companies including Google, Facebook, and even NASA. You will develop your understanding of angles and symmetry of shapes, through decomposing the creation of shapes into a sequence of instructions using the “turtle” library in Python. You will also understand how to make your algorithms more efficient by identifying patterns in your code and using loops to make your programs more concise.

## Draw the Image

Follow the instructions to draw the image:

## Geometry

### Shape 1:

- 2 sets of equal sides
- 4 equal angles ( $90^\circ$ )
- 2 lines of symmetry

### Shape 2:

- 4 equal sides
- 4 equal angles ( $90^\circ$ )
- 4 lines of symmetry

### Shape 3:

- 3 equal sides
- 3 equal angles ( $60^\circ$ )

## What is Python?

Python is

---

---

---

---



## First Turtle Program

Write a program to create a square and name it as **square.py**

Write a program to create a rectangle and name it as **rectangle.py**

Hint: Remember the properties of a rectangle with respect to the sides.

Useful hints:

Every program should start with importing the turtle library

```
import turtle
```

Create the turtle object

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

Command to set the shape of the turtle object to turtle

```
pen.shape("turtle")
```

Command to go forward

```
pen.forward(50)
```

Command to turn right

```
pen.right(90)
```

## Colour the Shapes

Modify the **square** program to draw the square with **red** lines and fill it with **green** colour.

Modify the **rectangle** program to draw the rectangle with **orange** lines and fill it with **purple** colour.

Useful hints:

Command to set the line colour

```
pen.color("yellow")
```

Command to set the fill colour

```
pen.fillcolor("green")
```

Command to indicate the beginning of the fill area

```
pen.begin_fill()
```

Command to indicate the end of the fill area

```
pen.end_fill()
```



## Pen Exercises

Draw 2 shapes of your choice (square/rectangle/triangle) next to each other with some space between them.

Useful hints:

Command to lift the pen up to stop drawing

```
pen.penup ()
```

Command to put the pen down to start drawing

```
pen.pendown ()
```

## Fun Flags

Using the Turtle commands we have learned so far, create one or more of the following lifeguard flags or a flag of the world.



DANGER  
No swimming



Lifeguard  
on duty



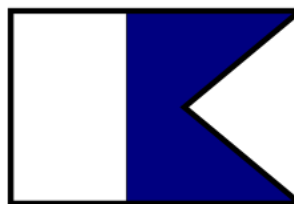
CAUTION  
Seek advice



Surfing area  
No swimming



Safe to swim



Diving in  
progress

## Let's Loop

Modify these programs to use loops and identify the shapes.

```
#Program 1
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.color("green")
pen.forward(100)
pen.right(72)
pen.forward(100)
pen.right(72)
pen.forward(100)
pen.right(72)
pen.forward(100)
pen.right(72)
pen.forward(100)
```

## Let's Loop

Modify these programs to use loops and identify the shapes.

```
#Program 2
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.color("green")
pen.forward(100)
pen.right(60)
pen.forward(100)
pen.right(60)
pen.forward(100)
pen.right(60)
pen.forward(100)
pen.right(60)
pen.forward(100)
pen.right(60)
pen.forward(100)
pen.right(60)
```

## Loopy Designs

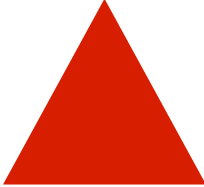
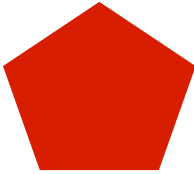

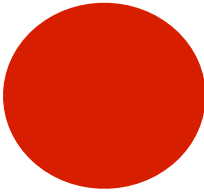
Using the loops, pen, and colour commands, try to program some fun designs. It need not be a specific geometrical shape.

Useful hints:

Syntax for a **for** loop in Python.

```
for i in range(<any number>):  
    <loop body>
```

## Angles

Polygons	Number of Sides	Size of each Exterior Angle
		
		
		
		

## Draw Any Shape

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

```
for i in range(sides):
    pen.forward(100)
    pen.right(360/sides)
```

```
pen.penup()
pen.backward(10)
pen.right(100)
pen.forward(100)
pen.pendown()
```

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

Update the above code to do the following:

- Ask the user for the colour they would like to use to fill their shape.
- Store it in **usercolour** variable
- Use the variable in fillcolor() command

## Conditional Shapes

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

```
for i in range(sides):
    pen.forward(100)
    pen.right(360/sides)
```

```
pen.penup()
pen.backward(10)
pen.right(100)
pen.forward(100)
pen.pendown()
```

```
for i in range(2):
    pen.forward(sidea)
    pen.right(90)
    pen.forward(sideb)
    pen.right(90)
```

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

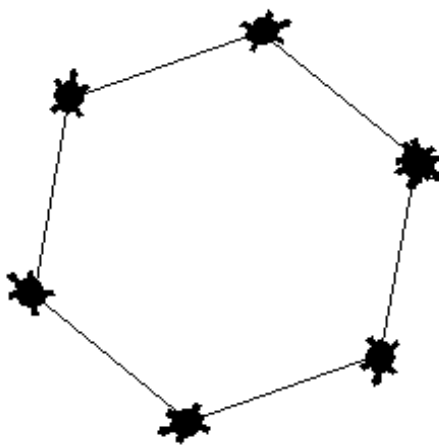
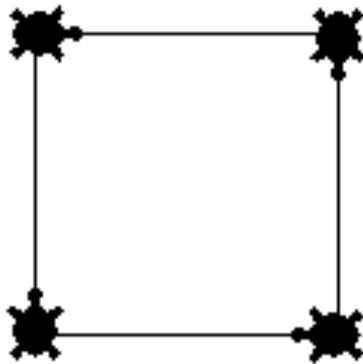
```
if (sides == __ ):
    side = <.....>
    sideb= <.....>
    <.....>
else:
    <.....>
```



## Stamp the Turtles

Using the commands we have learned today create random shapes and add `pen.stamp()` to the turtle on the screen on each edge of the shapes.

Sample shapes with stamps are provided below.



## Turtle Review

```
1. import turtle
2. pen = turtle.Turtle()
3. pen.shape("circle")
4. pen.width(2)
5. pen.penup()
6. randomNumber = int(input("Choose a number
                           between 20-200"))
7. if (randomNumber < 20):
8.     print("20 is the least value you can
           choose")
9. else:
10.     if(randomNumber > 200):
11.         print("You cannot choose more than
                200")
12.     else:
13.         for i in range(5, randomNumber, 2):
14.             pen.stamp()
15.             pen.forward(i)
16.             pen.right(24)
```

## Turtle Review

For the program in the previous page, copy the piece of code which contains the following:

a. **Variable name:**

b. **Loops:**

c. **Conditionals:**

d. **User input:**

e. **Turtle library:**



**technocamps**



@Technocamps



Find us on  
**Facebook**