Welcome to Micro-computers, Python Programming & Virtual Turtles

Today we are going to use the **Python** programming language and it's **Turtle Graphics** library to learn about programming!

Python is a beginner friendly programming language that is useful for teaching programming and is also widely used in web development, data science, artificial intelligence and more. **Python** is one of the most *in-demand* and *widely used* programming languages in the world.

Python's **Turtle Graphics** library will provide visual feedback while you learn to code.

The tiny computer sitting in front of you is called a **Raspberry Pi**. This model was $35 brand new and can be used for anything from programming, to web browsing, to home automation and robotics... they can even run low-demand or retro *video games*.

Turtle Setup & First Steps

First we have to import a library that will allow us to use **Turtle Graphics**.

Type the following code into your code editor to import the Turtle Graphics library.

```
import turtle
turtle = turtle.Turtle()
turtle.shape("turtle")
turtle.speed(1)
```

Then, type the following code **below the code you had previously typed**.
This will make Turtle "go forward 50 units".

```
turtle.forward(50)
```

You should now have 3 lines of code.

Next...

- Click the **"Run"** menu at the top of your code editor
- Click **"Run Module"**

After clicking **"Run Module"**, a new window should have opened.

In that new window there should be a short line starting at the center of the window and ending at our Turtle to the right.

Close that new window to get back to your editor.

Draw a Square

Now let's draw a square!

Delete all of the code in your editor except for the lines below.

```
import turtle
turtle = turtle.Turtle()
turtle.shape("turtle")
turtle.speed(1)
```

To draw a square, type the following code **below the code you had previously typed**.
(I know that's a *lot* of typing... don't worry, programmers are *much too lazy* to let this continue for long!)

```
turtle.forward(50)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
```

After typing that code into your editor...

- Click the **"Run"** menu at the top of your code editor
- Click **"Run Module"**

After clicking **"Run Module"**, a new window should have opened.

In that new window you should see Turtle draw a square! Nice job!

Close that new window to get back to your editor.

Draw a Square... With a Loop!

Phew, that was too much typing... let's be a ***true*** programmer and get lazy ;).

We're going to draw another square, but this time we'll use a **loop** to save a ton of typing.

Delete all of the code in your editor except for the lines below.

```
import turtle
turtle = turtle.Turtle()
turtle.shape("turtle")
turtle.speed(1)
```

Loops allow us to run code over and over again as many times as we specify.
Loops are a fundamental building block of most programming languages and are essential to writing almost any program you can think of... AND they save a LOT of typing!

To draw a square using a **loop**, type the following code **below the code you had previously typed**.

```
for number in range(0, 4):
    turtle.forward(50)
    turtle.right(90)
```

After typing that code into your editor...

- Click the **"Run"** menu at the top of your code editor
- Click **"Run Module"**

After clicking **"Run Module"**, a new window should have opened.

In that new window you should see the same square as before, but that took a *lot* less code... now we're getting somewhere!

Close that new window to get back to your editor.

Write a Function That Draws Any Shape

Squares are VERY EXCITING, I know, but triangles and hexagons are cool too... how can we make our code more flexible so we can draw different shapes?

Functions & Methods
You have been working with **methods** on the `turtle` object for some time now. When you type `turtle.forward(50)` you are calling the `forward` **method** on the `turtle` object.

Functions and **methods** are slightly different things, *but the differences don't matter to us today, so for now* ***consider "functions" and "methods" as the same thing***.

Functions (and methods) are named, reusable blocks of code that often accept "arguments" to allow changes in their behavior.

When you type `turtle.forward(50)` you are passing the number **50** as an **argument** into the `forward()` **method**.

What will happen if we pass **100** as the **argument** into the `forward()` **method**?
The `forward()` **method** will simply change it's output... pushing Turtle forward more.

Ok, now let's create a function...
Delete all of the code in your editor except for the lines below.

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

```
turtle.shape("turtle")
turtle.speed(1)
```

To create a **function** that can draw shapes of various sizes…
Type the following code **below the code you had previously typed**.

```
def drawShape(sides):
    degrees = 360 / sides
    for number in range(0, sides):
        turtle.forward(25)
        turtle.right(degrees)
```

We have **defined a function** that can draw shapes, but we haven't **called that function** yet.

To **call the function** that you defined above (and to move Turtle out of the way for each shape)…
Type the following code **below the code you had previously typed**.

```
drawShape(3)
turtle.forward(50)
drawShape(4)
turtle.forward(50)
drawShape(5)
turtle.forward(50)
drawShape(6)
```

After typing that code into your editor…

- Click the **"Run"** menu at the top of your code editor
- Click **"Run Module"**

After clicking **"Run Module"**, a new window should have opened.

In that new window you should see a triangle, square, pentagon and hexagon.

Turtle did all of that while you wrote the same amount of code as when you drew just one square!
That is the power of a function.

Close that new window to get back to your editor.

If's & Elses

The final building block of programming that we're going to cover is one of the most important, the **if statement**.

The **if statement** allows a program to make decisions by running a block of code when a condition is **true**.
If the condition is **false**, then the code inside the **if statement** will not be ran and, if you coded for it, an **else statement** or **else-if statement** can be hit.

Let's improve the function we wrote above by adding an **if statement** and an **else statement** that will act as safety checks.

Delete all of the code in your editor except for the lines below.

```
import turtle
turtle = turtle.Turtle()
turtle.shape("turtle")
turtle.speed(1)
```

To improve the function we wrote above by adding some safety checks...
Type the following code **below the code you had previously typed**.

```
def drawShape(sides):
    if sides >= 3:
        degrees = 360 / sides
        for number in range(0, sides):
            turtle.forward(25)
            turtle.right(degrees)
    else:
        turtle.left(45)
        turtle.right(90)
        turtle.left(45)
```

This line of code...

```
if sides >= 3:
```

...is our new safety check.

The `>=` symbols mean "greater than or equal to".

So, the code `if sides >= 3` means...

- If sides is "greater than or equal to" 3, that condition will be **true** and...
  The code immediately following the **if statement** will be ran. Meaning the shape will be drawn.
- If sides is "less than" 3, that condition will be **false** and we will "fall into" the **else statement**...
  The code immediately following the **else statement** will be ran and Turtle will "shake his head" in disapproval.

To call the function that you defined above and test our new safety checks...
Type the following code **below the code you had previously typed**.

```
import time
drawShape(0)
time.sleep(1)
drawShape(2)
time.sleep(1)
drawShape(3)
```

After typing that code into your editor...

- Click the **“Run”** menu at the top of your code editor
- Click **"Run Module"**

After clicking **“Run Module”**, a new window should have opened.

In that new window you should see Turtle “shake it’s head” twice (hit our new safeties), and then Turtle should draw a triangle.

Close that new window to get back to your editor.

Congratulations!

You’ve made it to the end of the programming crash course!

Please let the helpers know if you have any more questions about programming, what we do at work, Turtle Graphics, Raspberry Pi computers… or anything else.

*Thank you **so much** for stopping by!*