Assignment 6: CALLING AND WRITING FUNCTIONS

For this assignment, our objective is to learn how to call and write functions. We will be using a python library called turtle, which is a tool for teaching programming concepts and creating neat visualizations. We will be creating functions to draw cool patterns by calling the built-in turtle functions. You will create two custom functions that create visualizations. Documentation on turtle library here: https://docs.python.org/3/library/turtle.html

Deliverables:

1. a PDF document with your figures and your answers to the questions at the end of the assignment
    a. Include in your PDF a description of each of your functions
    b. How could you use functions in an interactive art setting?
2. Your modified eng103_base6.py file

Calling Functions:

To call functions, we will focus on two different types of functions: (1) functions with outputs and (2) functions without output.

Functions with output: Let's start with an example function with output:

```
def example_function(a,b,c):
    X = a+b+c
    return X
```

To call this function and store its output, we will have to set the function equal to a variable name. For example:

```
X = example_function(a,b,c)
```

However, it isn't necessary to use the same names for the variables as in the function definition. We can input any variables of the correct type and save the output as any variable name we like. For example:

```
triceratops = 3
trex = 2
velociraptor = 4
dinosaur_add = example_function(triceratops,trex ,velociraptor)
```

In this case, we set our inputs to the dinosaur names, and our output is saved to the dinosaur_add variable.

We can also put values directly into functions. For example:

```
X = example_function(1,4,3)
```

Functions without output: Not all functions have output. Let's look at an example function without output:

```
def example_function(a,b,c):
     X = a+b+c
     print(X)
```

To call this function, we just have to write the function name and give it acceptable inputs. For example:

```
a = 1
b = 3
c = 5
example_function(a,b,c)
```

Like before, we can also have inputs that aren't named the same as the variables in the function definition. For example:

```
ice = 1
water = 3
steam = 5
example_function(ice,water,steam)
```

This will have the same result as the prior example since the input variables have the same value.

Finally, we can also put direct values into the function as an input. For example:

```
example_function(1,3,5)
```

Writing Functions:

This week you will practice writing functions. The scaffold for writing a function is fairly simple. First, you define the function with its name and your desired inputs. For simplicity, we are not using classes in this base code so you do not need to worry about the "self" input that you have seen in previous weeks. You start your defining line with "def" then write your function name and the inputs in parentheses (not brackets) immediately after, and a colon at the end of the line. For example:

```
def function_name(inputs go here):
```

Then, hit enter and make sure everything you want inside the function is another level indented. This is how python will know what is included in your function and what is not.

```
def function_name(inputs go here):
     Code at this indent is part of the function
Code at this indent is NOT part of the function
```

Using turtle, your functions likely will not be returning any output, as the output is the visualization. However, if you wanted to have a function that returns some output, you will use "return" at the end of your function code. For example:

```
def function_name(a,b):
     X = a+b
     return X
```

STARTING:

1. Open your eng103_base6.py
2. Run the base code as is to make sure it is error free
    a. The turtle package should be pre-installed with your version of python. However, if you run the base code and it says it cannot find the turtle package, install it.

EDITING THE CODE:

In this assignment, and all future assignments, the places where you have to edit the code will be denoted in the comments as follows:

```
##### EDIT HERE ######
```

In this assignment, and all future assignments, the places where you have to STOP editing the code will be denoted in the comments as follows:

```
##### STOP EDITING HERE #####
```

This way, you know which parts you can change without losing functionality in the code or adding additional errors.

> *Reminder: SPACING MATTERS IN PYTHON. You will get errors if the code is not formatted correctly.*

WRITING YOUR FUNCTIONS:

For this assignment, write two custom functions to create some patterns using turtle. We have provided you with two example functions to give you an idea of the kind of output we are looking for. The base code is marked where to add in your custom functions. It is also marked where to call your custom functions in the main function at the bottom.

To save your output visualizations, you will need to screenshot the plot that pops up. Turtle supports saving the visualization automatically, but it comes out in a weird format so for simplicity we will be screenshotting. With the examples, it prints “SCREENSHOT NOW!” in the terminal when the visualization is done, and will pause for 30 seconds to allow for you to screenshot.

Below are some of the basic functions of turtle:
SOURCE: https://www.geeksforgeeks.org/turtle-programming-python
You can also find many examples by searching “turtle python examples”

Here is a comprehensive list of the turtle functions: https://docs.python.org/3/library/turtle.html

Feel free to use elements of examples you find and combine them to create your own functions! However, **DO NOT DIRECTLY COPY EXAMPLES FROM THE INTERNET**

FUNCTION NAME	INPUTS	DESCRIPTION
Turtle()	None	Creates and returns a new turtle object
forward()	amount	Moves the turtle forward by the specified amount
backward()	amount	Moves the turtle backward by the specified amount
right()	angle	Turns the turtle clockwise
left()	angle	Turns the turtle counterclockwise

penup()	None	Picks up the turtle’s Pen
pendown()	None	Puts down the turtle’s Pen
up()	None	Picks up the turtle’s Pen
down()	None	Puts down the turtle’s Pen
width()	amount	Sets the pen line width
color()	Color name	Changes the color of the turtle’s pen
fillcolor()	Color name	Changes the color of the turtle will use to fill a polygon
bgcolor()	Color name	Changes the background color
heading()	None	Returns the current heading
position()	None	Returns the current position
goto()	x, y	Move the turtle to position x,y
begin_fill()	None	Remember the starting point for a filled polygon

end_fill()	None	Close the polygon and fill with the current fill color
dot()	None	Leave the dot at the current position
stamp()	None	Leaves an impression of a turtle shape at the current location
shape()	shapename	Should be ‘arrow’, ‘classic’, ‘turtle’ or ‘circle’
clear()	None	Clears the plot area