**Outline**

Develop a better understanding of procedural sequencing by solving shape drawing challenges using the turtle environment.

**Objectives**

* Use correct terminology to describe programming concepts;
* Describe the types of data that computers can process and store (e.g., numbers, text);
* Explain the difference between constants and variables used in programming;
* Use variables, expressions, and assignment statements to store and manipulate numbers and text in a program

**Materials**

* Python Turtle Development Environment at: https://repl.it/
* PythonWorksheetII form the GitHub Repository
* Web links identified in the questions below

**Level 1: Drawing Basic Shapes With Python Turtle**

1. Open the document PythonWorksheetII from the class GItHub repository.   
   Read over “Part III” at the end of the PythonWorksheetII document.

I have read over Part III.

1. Create a new Repl by selecting the “Python with Turtle” language / environment.

I have created a new Repl.

1. Begin all of your turtle programs with the following code to create a “pen”:

import turtle

myPen = turtle.Turtle()

I have done this

1. Create a program to draw a red circle.
   1. Provide a listing of your program code below:

import turtle

myPen = turtle.Turtle()

myPen.color("red")

myPen.circle(100)

1. Create a program to draw any three of the shapes described in “Part III” of   
   the PythonWorksheetII document.
   1. Provide a listing of your program code below:

Circle within square:

import turtle

myPen = turtle.Turtle()

myPen.color("red")

myPen.circle(100)

myPen.color("Green")

myPen.forward(100)

myPen.left(90)

myPen.forward(200)

myPen.left(90)

myPen.forward(200)

myPen.left(90)

myPen.forward(200)

myPen.left(90)

myPen.forward(100)

Square(200x200):

myPen.color("Green")

myPen.forward(100)

myPen.left(90)

myPen.forward(200)

myPen.left(90)

myPen.forward(200)

myPen.left(90)

myPen.forward(200)

myPen.left(90)

myPen.forward(100)

Plus/Cross sign:

import turtle

myPen = turtle.Turtle()

myPen.color("Green")

myPen.forward(100)

myPen.right(90)

myPen.forward(40)

myPen.right(90)

myPen.forward(100)

myPen.left(90)

myPen.forward(100)

myPen.right(90)

myPen.forward(40)

myPen.right(90)

myPen.forward(100)

myPen.left(90)

myPen.forward(100)

myPen.right(90)

myPen.forward(40)

myPen.right(90)

myPen.forward(100)

myPen.left(90)

myPen.forward(100)

myPen.right(90)

myPen.forward(40)

myPen.right(90)

myPen.forward(100)

**Level 2: Using a Loop**

1. Google the keywords “Python Turtle Methods”.
   1. Explain how the “goto” method works and how you could use it when drawing repeated shapes.

The goto method takes the drawing turtle from one point to another point. In this process the turtle draws a straight line between the two points if the pen is down. Goto can be used to go to a certain point on the grid and this could help when making repeated shapes. Sometimes repeated shapes need symmetry and goto can help with that because it takes the turtle to a certain point on a grid and the x or y could be matched to give precise symmetry.

* 1. List some other useful methods not listed in “Part III” at the end of the PythonWorksheetII document.

myPen.fillcolor(“red”) Used to fill a polygon space with color

myPen.dot Used to leave a data at the position of the turtle

1. Create a repeating pattern on your screen. The pattern must meet the following requirements:
   1. The basic pattern must be made up of several individual Turtle methods (e.g. changes of colour, changes of direction, size, motion, etc.)
   2. The basic pattern must be repeated several times with a shift in starting position each time.

import turtle

myPen = turtle.Turtle()

myPen.speed(15000)

for i in [0,1,2,3,4,5,6,7,8,9,10] :

myPen.speed(1000000000000000000000000000)

myPen.color("blue")

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.left(5)

myPen.forward(50)

myPen.color('green')

myPen.circle(100)

myPen.circle(50)

1. Use a Python Loop to create your repeating pattern
   1. The Loop may be a Counted Loop or a Conditional Loop
   2. The indented block of code for the loop should be your basic pattern.
2. Provide a listing of your repeating pattern loop below.

import turtle

myPen = turtle.Turtle()

myPen.speed(15000)

for i in [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,3,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51, 52, 53, 54, 55, 56, 57, 58, 59,

60, 61, 62, 63, 64, 65, 66, 67, 68, 69,

70, 71, 72, 73, 74, 75, 76, 77, 78, 79,

80, 81, 82, 83, 84, 85, 86, 87, 88, 89,

90, 91, 92, 93, 94, 95, 96, 97, 98, 99,

100] :

myPen.speed(1000000000000000000000000000)

myPen.color("pink")

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.left(5)

myPen.forward(50)

myPen.color('yellow')

myPen.circle(50)

myPen.circle(25)

**Level 3: Defining a Function**

1. Google the keywords “Python Function Syntax”.
   1. Explain what the “def” keyword does

Def is used to define a function. If you write anything between def then function name and end, then that code will be executed every time when you write the function name.

* 1. Explain any special rules regarding the function name

The function name must uniquely identify that specific code, and it must be remarkable as you will be using the function name for that code in the future.

* 1. Explain what the parameters (or arguments) do

We can pass values to a function through parameters. There are not always needed.

* 1. Where should the colon “:” be placed

The colon should be place after the Function name and/or Parameter and in the same line as the Function name.

* 1. Explain how to write Python statements that make up the function body

Python statements can be anything that tell the turtle to do something. The body should have one or more valid functions.

* 1. Explain the “return” statement

The return stamen can be used to return a value from the function.

1. Provide an example of a simple function that uses one or more parameters.
   1. Write the function definition below

A function is a block of organized piece of code that could be reused again with one command.

import turtle

myPen = turtle.Turtle()

def square(Size) :

myPen.forward(Size)

myPen.right(90)

myPen.forward(Size)

myPen.right(90)

myPen.forward(Size)

myPen.right(90)

myPen.forward(Size)

* 1. Write some code to call the function below

square(100)

1. Convert your basic pattern (from Level 2 above) into a function

* The function name should be “my\_pattern”
* The parameters should be the x and y starting position for your pattern
* Your function does not need to use the “return” statement

import turtle

myPen = turtle.Turtle()

myPen.speed(15000)

def pattern(x,y):

for i in [0,1,2,3,4,5,6,7,8,9,10] :

myPen.goto(x,y)

myPen.speed(0)

myPen.color("blue")

myPen.goto(x,y)

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.left(5)

myPen.forward(50)

myPen.color('green')

myPen.circle(100)

myPen.circle(50)

pattern(0,0)

pattern(100,100)

1. Use a your basic pattern function and a Python Loop to create your repeating pattern
   1. The Loop may be a Counted Loop or a Conditional Loop
   2. Your function should be called from within the loop.
2. Provide a listing of your function definition and repeating pattern loop below.

import turtle

myPen = turtle.Turtle()

myPen.speed(15000)

def pattern(x,y):

for i in [0,1,2,3,4,5,6,7,8,9,10] :

myPen.goto(x,y)

myPen.speed(0)

myPen.color("blue")

myPen.goto(x,y)

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.right(90)

myPen.forward(100)

myPen.left(5)

myPen.forward(50)

myPen.color('green')

myPen.circle(100)

myPen.circle(50)

for i in [0,1,2,3,4,5,6,7,8,9,10] :

pattern(0,0)

pattern(0,150)

pattern(150,0)

pattern(150,150)