**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.
2. Create an new Repl by selecting the “Python with Turtle” language / environment.
3. Begin all of your turtle programs with the following code to create a “pen”:

import turtle

myPen = turtle.Turtle()

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

import turtle

myPen = turtle.Turtle()

myPen.circle(60)

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:

Shape A) import turtle

myPen = turtle.Turtle()

myPen.forward(60)

myPen.left(90)

myPen.forward(60)

myPen.left(90)

myPen.forward(60)

myPen.left(90)

myPen.forward(60)

Shape B) import turtle

myPen = turtle.Turtle()

myPen.forward(20)

myPen.left(90)

myPen.forward(40)

myPen.right(90)

myPen.forward(40)

myPen.left(90)

myPen.forward(20)

myPen.left(90)

myPen.forward(40)

myPen.right(90)

myPen.forward(40)

myPen.left(90)

myPen.forward(20)

myPen.left(90)

myPen.forward(40)

myPen.right(90)

myPen.forward(40)

myPen.left(90)

myPen.forward(20)

myPen.left(90)

myPen.forward(40)

myPen.right(90)

myPen.forward(40)

Shape C) import turtle

myPen = turtle.Turtle()

myPen.color("blue")

myPen.circle(40)

myPen.up()

myPen.down()

myPen.color("red")

myPen.forward(40)

myPen.left(90)

myPen.forward(80)

myPen.left(90)

myPen.forward(80)

myPen.left(90)

myPen.forward(80)

myPen.left(90)

myPen.forward(80)

**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 is where you can pick a point on your screen where you want your turtle to draw.

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

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.)

import turtle

myPen = turtle.Turtle()

myPen.speed(100)

myPen.color("blue")

myPen.circle(40)

myPen.up()

myPen.down()

myPen.color("red")

myPen.forward(40)

myPen.left(90)

myPen.forward(80)

myPen.left(90)

myPen.forward(80)

myPen.left(90)

myPen.forward(80)

myPen.left(90)

myPen.forward(80)

* 1. The basic pattern must be repeated several times with a shift in starting position each time.

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(100)

for hi in"12345":

myPen.circle(40)

myPen.forward(50)

**Level 3: Defining a Function**

1. Google the keywords “Python Function Syntax”.
   1. **Explain what the “def” keyword does:** Marks the start of the function header
   2. **Explain any special rules regarding the function name:**  A colon has to be used to mark the end of a function header
   3. **Explain what the parameters (or arguments) do:** Allow to do multiple arguments
   4. **Where should the colon “:” be placed:** At the end of a function header
   5. **Explain how to write Python statements that make up the function body:** The elif statement allows you to check multiple expressions for true and execute a block of code as soon as one of the conditions evaluates to true.
   6. **Explain Return Statement:**  a return statement causes execution to leave the current subroutine and resume at the point in the code immediately after where the subroutine was called, known as its return address.
2. Provide an example of a simple function that uses one or more parameters.

def my\_function(color=”black”):

print (“My favourite color is” + color)

my\_function(“Blue”)

my\_function(“red”)

my\_function()

my\_function(“green”)

* 1. Write the function definition below

This function tells the program to print certain words or letters in a line. For example, something different can be printed with the print command.

* 1. Write some code to call the function below

def my\_function(color=”black”)

print(color+” is a good colour”)

my\_pattern(“blue”)

my\_pattern()

1. Convert your basic pattern (from Level 2 above) into a function
2. The function name should be “my pattern”
3. The parameters should be the x and y starting position for your pattern
4. Your function does not need to use the “return” statement

def my\_pattern(color=”Red”)

print(color+” is a good color”)

my\_pattern(“blue”)

my\_pattern()

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.9
2. Provide a listing of your function definition and repeating pattern loop below.

def my\_pettern (color=”Red”)

print(color+” is a good color”)

my\_pattern(“blue”)

my\_pattern()

my\_pattern(“green”)

my\_pattern(“black”)