**Level 0: Teacher Demo of Sample Programs**

1. Sample program #1 is an example of a "Syntax Error". Follow the teacher demo and explain the characteristics of a syntax error. Consider the following criteria:  
   1. Did the program have an error before starting to run?

-Yes

* 1. Did the program encounter an error before it finished running?

-Yes

* 1. Did the program do what it was supposed to do?

-No

1. Sample program #2 is an example of a "Run-time Error". Follow the teacher demo and explain the characteristics of a run-time error. Consider the following criteria:  
   1. Did the program have an error before starting to run?

-No

* 1. Did the program encounter an error before it finished running?

-Yes

* 1. Did the program do what it was supposed to do?

-No

1. Sample program #3 is an example of a "Logic Error". Follow the teacher demo and explain the characteristics of a logic error. Consider the following criteria:  
   1. Did the program have an error before starting to run?.

-No

* 1. Did the program encounter an error before it finished running?

-No

* 1. Did the program do what it was supposed to do?

-No

**Level 1: Syntax Errors**

1. Research the definition of the word "Syntax". Summarize its meaning below and how it relates to computer languages and programming.

**Syntax** is the set of rules in a language that are required to make well-formed sentences. In programming, syntax can be referred to as the proper way to write code.

1. Research the definition of a "Syntax Error" related to computer programming. Summarize this definition below.

**Syntax Error** occurs when a part/whole of the code violates the syntax rules of the programming language. Every language has different syntax and transferring code from one language changes the syntax.

1. Explain why Sample Program #1 is an example of a "Syntax Error".

**Sample Program #1** has a **Syntax Error** because the code has an error before starting and it does not produce the expected results. In **Sample Program #1,** line 7 has a syntax error because there is an open bracket and line 17 has a syntax error because the name colours was not defined, it was spelled as colors (without a **u**).

1. Find and correct the syntax errors in Sample Program #1. Provide a listing of your corrected program below.
   * Use a "#" at the beginning of each line containing an error   
     to "Comment Out" the bad code
   * List the corrected code line underneath the commented-out error line

**Sample Program #1**

import turtle

myPen = turtle.Turtle()

circleColors = [(196,196,0),(196,0,196),(0,196,196)]

def drawCircle(rgb) :

#myPen.down(

myPen.down()

myPen.color(rgb)

myPen.begin\_fill()

myPen.circle(8)

myPen.end\_fill()

myPen.up()

myPen.forward(22)

circleNumber = 0

for circleIndex in range(3) :

#drawCircle(circleColours[circleNumber])

drawCircle(circleColors[circleNumber])

circleNumber = circleNumber + 1

**Level 2: Run-time Errors**

1. Research the definition of a "Run-time Error" related to computer programming. Summarize this definition below.

The term **Run-time** stands for the time when the program is undergoing execution. A **Run-time Error** occurs when the code runs without any errors but encounters errors in the middle of execution. This stops the code from running.

1. Explain why Sample Program #2 is an example of a "Run-time Error".

**Sample Program #2** is an example of a **Run-Time Error** because the program started running but encountered an error before finishing. It did not output the expected results.

1. Explain the difference between a "syntax error" and a "run-time error".

A **syntax error** is different from a **Run-time error** because the code does not run if there is a **syntax error** but the code does run if there is a **Run-time error.** A syntax error prevents the code from running whereas a runtime error causes the already-running code to stop.

1. Find and correct the run-time errors in Sample Program #2. Provide a listing of your corrected program below.
   * Use a "#" at the beginning of each line containing an error   
     to "Comment Out" the bad code
   * List the corrected code line underneath the commented out error line

import turtle

myPen = turtle.Turtle()

circleColours = [(196,196,0),(196,0,196),(0,196,196)]

def drawCircle(rgb) :

myPen.down()

myPen.color(rgb)

myPen.begin\_fill()

myPen.circle(8)

myPen.end\_fill()

myPen.up()

myPen.forward(22)

#circleNumber = 1

circleNumber = 0

#for circleIndex in range(3):

for circleIndex in range(3):

drawCircle(circleColours[circleNumber])

circleNumber = circleNumber + 1

**Level 3: Logic Errors**

1. Research the definition of a "Logic Error" related to computer programming. Summarize this definition below.

A **logic error** is occurs when the code runs properly but the output is not what is expected.

This type of error will not terminate the code. Rather, the output results are not what is desired.

1. Explain why Sample Program #3 is an example of a "Logic Error".

**Sample Program #3** in a **Logic Error** because all the circles are black, which is not the desired output.

1. Find and correct the logic errors in Sample Program #3. Provide a listing of your corrected program below.
   * Use a "#" at the beginning of each line containing an error   
     to "Comment Out" the bad code
   * List the corrected code line underneath the commented out error line

import turtle

myPen = turtle.Turtle()

circleColours = [(196,196,0),(196,0,196),(0,196,196)]

def drawCircle(rgb) :

myPen.color(rgb)

myPen.down()

myPen.begin\_fill()

myPen.circle(8)

myPen.end\_fill()

myPen.up()

myPen.forward(22)

numOfCircles = 3

#for circleIndex in range(2) :

for circleIndex in range(numOfCircles):

#circleNumber = numOfCircles - circleIndex - 1

drawCircle(circleColours[circleIndex])

1. Explain the difference between a "logic error" and a "syntax error".

The main difference between a **Logic Error** and a **Syntax Error** is that a syntax error prevents the code from running whereas a logic error does run the full code but does not produce the desired output.

1. Explain the difference between a "logic error" and a "run-time error".

The main difference between a **Logic Error** and a **Run-time** **error** is that the code stops in the middle of executing in a Run-time error whereas the code finishes running in a logic error.

**Level 4: Your Sample Program**

1. Create a sample program to show the different types of programming errors. Provide your program listing below.
   * Your program must be of your own design and must be different from the sample programs provided in this module.
   * Your program must contain at least one example of each of: a syntax error, a run-time error, and a logic error.
   * Provide the corrected code in a comment underneath the error code (using a "#" at the beginning of the comment line).

**Error Code:**

import turtle

myPen = turtle.Turtle()

myPen.speed(0)

#The line below has a syntax error (mission a semicolon : )

def inner\_diamond()

myPen.goto(0,0)

for i in range(0,11):

yFrom=10-i

xTo=i

myPen.penup()

myPen.goto(0,20\*yFrom)

myPen.pendown()

myPen.goto(20\*xTo,0)

for i in range(0,11):

yFrom=10-i

xTo=i

myPen.penup()

myPen.goto(0,-20\*yFrom)

myPen.pendown()

myPen.goto(20\*xTo,0)

#The code below should be range(0,11) for the expected results. (Logic Error)

for i in range(5,16):

yFrom=10-i

xTo=i

myPen.penup()

myPen.goto(0,-20\*yFrom)

myPen.pendown()

myPen.goto(-20\*xTo,0)

for i in range(0,11):

yFrom=10-i

xTo=i

myPen.penup()

#There is a run-time error below (yFrom is an integer and can’t be indexed)

myPen.goto(0,20\*yFrom[0])

myPen.pendown()

myPen.goto(-20\*xTo,0)

inner\_diamond()

**Working Code:**

import turtle

myPen = turtle.Turtle()

myPen.speed(0)

#def inner\_diamond()

def inner\_diamond():

myPen.goto(0,0)

for i in range(0,11):

yFrom=10-i

xTo=i

myPen.penup()

myPen.goto(0,20\*yFrom)

myPen.pendown()

myPen.goto(20\*xTo,0)

for i in range(0,11):

yFrom=10-i

xTo=i

myPen.penup()

myPen.goto(0,-20\*yFrom)

myPen.pendown()

myPen.goto(20\*xTo,0)

#for i in range(5,16):

for i in range(0,11):

yFrom=10-i

xTo=i

myPen.penup()

myPen.goto(0,-20\*yFrom)

myPen.pendown()

myPen.goto(-20\*xTo,0)

for i in range(0,11):

yFrom=10-i

xTo=i

myPen.penup()

#myPen.goto(0,20\*yFrom[0])

myPen.goto(0,20\*yFrom)

myPen.pendown()

myPen.goto(-20\*xTo,0)

inner\_diamond()