**Module 2 Graded Assessment**

**1.Complete the function by filling in the missing parts. The color\_translator function receives the name of a color, then prints its hexadecimal value. Currently, it only supports the three additive primary colors (red, green, blue), so it returns "unknown" for all other colors.**

def color\_translator(color):

if color == "red":

hex\_color = "#ff0000"

elif color == "green":

hex\_color = "#00ff00"

elif color == "blue":

hex\_color = "#0000ff"

else:

hex\_color = "unknown"

return hex\_color

print(color\_translator("blue")) # Should be #0000ff

print(color\_translator("yellow")) # Should be unknown

print(color\_translator("red")) # Should be #ff0000

print(color\_translator("black")) # Should be unknown

print(color\_translator("green")) # Should be #00ff00

print(color\_translator("")) # Should be unknown

**2.What's the value of this Python expression: "big" > "small"**

False

**3.What is the elif keyword used for?**

To handle more than two comparison cases

**4.Students in a class receive their grades as Pass/Fail. Scores of 60 or more (out of 100) mean that the grade is "Pass". For lower scores, the grade is "Fail". In addition, scores above 95 (not included) are graded as "Top Score". Fill in this function so that it returns the proper grade.**

def exam\_grade(score):

if score > 95:

grade = "Top Score"

elif score >= 60:

grade = "Pass"

else:

grade = "Fail"

return grade

print(exam\_grade(65)) # Should be Pass

print(exam\_grade(55)) # Should be Fail

print(exam\_grade(60)) # Should be Pass

print(exam\_grade(95)) # Should be Pass

print(exam\_grade(100)) # Should be Top Score

print(exam\_grade(0)) # Should be Fail

**5.What's the value of this Python expression: 11 % 5?**

1

**6.Complete the body of the format\_name function. This function receives the first\_name and last\_name parameters and then returns a properly formatted string.**

Specifically:

If both the last\_name and the first\_name parameters are supplied, the function should return like so:

print(format\_name("Ella", "Fitzgerald"))

Name: Fitzgerald, Ella

If only one name parameter is supplied (either the first name or the last name) , the function should return like so:

print(format\_name("Adele", ""))

Name: Adele

or

print(format\_name("", "Einstein"))

Name: Einstein

Finally, if both names are blank, the function should return the empty string:

print(format\_name("", ""))

Implement below:

def format\_name(first\_name, last\_name):

if first\_name !="" and last\_name !="":

string = "Name: " + last\_name + ", " + first\_name

elif first\_name !="" and last\_name == "":

string = "Name: " + first\_name

elif first\_name == "" and last\_name != "":

string = "Name: " + last\_name

else:

string = ""

print(format\_name("Ernest", "Hemingway"))

# Should return the string "Name: Hemingway, Ernest"

print(format\_name("", "Madonna"))

# Should return the string "Name: Madonna"

print(format\_name("Voltaire", ""))

# Should return the string "Name: Voltaire"

print(format\_name("", ""))

# Should return an empty string

**7.The longest\_word function is used to compare 3 words. It should return the word with the most number of characters (and the first in the list when they have the same length). Fill in the blank to make this happen.**

def longest\_word(word1, word2, word3):

if len(word1) >= len(word2) and len(word1) >= len(word3):

word = word1

elif len(word2) > len(word1) and len(word2) > len(word3):

word = word2

else:

word = word3

return(word)

print(longest\_word("chair", "couch", "table"))

print(longest\_word("bed", "bath", "beyond"))

print(longest\_word("laptop", "notebook", "desktop"))

**8.What’s the output of this code?**

def sum(x, y):

return(x+y)

print(sum(sum(1,2), sum(3,4)))

10

**9.What's the value of this Python expression? ((10 >= 5*2) and (10 <= 5*2))**

True

**10.The fractional\_part function divides the numerator by the denominator, and returns just the fractional part (a number between 0 and 1). Complete the body of the function so that it returns the right number. Note: Since division by 0 produces an error, if the denominator is 0, the function should return 0 instead of attempting the division.**

def fractional\_part(numerator, denominator):

# Operate with numerator and denominator to

# keep just the fractional part of the quotient

if denominator == 0:

return 0

return (numerator / denominator) % 1

print(fractional\_part(5, 5)) # Should be 0

print(fractional\_part(5, 4)) # Should be 0.25

print(fractional\_part(5, 3)) # Should be 0.66...

print(fractional\_part(5, 2)) # Should be 0.5

print(fractional\_part(5, 0)) # Should be 0

print(fractional\_part(0, 5)) # Should be 0