Arizona State University
W. P. Carey School of Business

CIS 325: Programming for Business Analytics

Assignment 2 (Individual work)

Submission Instructions

- Submit your Python script according to the guidelines in this document as a .py file with the following naming convention: "A2-[ASURITE ID].py", where you will replace the text "[ASURITE ID]" with your ASURITE ID (= your ASU login ID).

  For example, if your ASURITE ID is "abcd12", then your submission file name would be "A2-abcd12.py".

- On the top (header) of your submission .py file, add your name and email address as shown below:

```
"""
@author: [YOUR FULL NAME]
@email: [YOUR EMAIL ADDRESS]
"""
```

- Add Python comment statements (such as "*# Answer to Question 1*") in your submission .py file to separate your answers between questions.

- Not following the above submission instructions will result in a **0.2 point reduction** from your grade.

Question 1 (1 points)

Create a Python for loop script of exactly 2 lines of code that generates a sequence of integer numbers starting from 2 up to (and including) 26 incremented by 2. For each integer generated, print in the Python console the following string – for instance, if you have generated the number four: "Generated number: 4". Ensure that your script generated the following output in the Python console. HINT: look up the range() function's documentation.

Expected output:

```
Generated number: 2
Generated number: 4
Generated number: 6
Generated number: 8
Generated number: 10
Generated number: 12
Generated number: 14
Generated number: 16
Generated number: 18
Generated number: 20
Generated number: 22
Generated number: 24
Generated number: 26
```

Question 2 (1 points)

Create a Python script of exactly three lines of code that uses two for loops with range statements and one print statement to generate the following output in the Python console:

Expected output:

```
3;1001
3;1002
3;1003
3;1004
3;1005
2;1001
2;1002
2;1003
2;1004
2;1005
1;1001
1;1002
1;1003
1;1004
1;1005
```

Question 3 (1 points)

Create a Python script that enables a user to enter an integer number into the Python console and stores such integer number into a variable named input1. Use Python if/elif/else statements to print the following output per the following conditions:

1. If input1 is a negative number, print in console “Input1 is negative”
2. If input1 is zero, print in console “Input1 is zero”
3. If input1 is a positive number less than or equal to 20, print in console “Input1 is positive but less than or equal to 20”
4. If input1 is greater than 20, print in console “Input1 is greater than 20”

Starter code:

```
input1 = int(input('Enter an integer number: '))

### Place your code below this line ###




### Place your code above this line ###
```

Question 4 (1 points)

An experienced Python programmer in your company wants your assistance calculating the sum of all of integer multipliers of 11 between (and including) the number 0 and up to (not including) the number 10. As an example:

$$0 \times 11 + 0 = 0$$
$$1 \times 11 + 0 = 11$$
$$2 \times 11 + 11 = 33$$
$$3 \times 11 + 33 = 66$$
$$\vdots$$
$$7 \times 11 + 231 = 308$$
$$8 \times 11 + 308 = 396$$
$$9 \times 11 + 396 = 495$$

Use the following starter code and complete the while loop:

Starter code:

```
j = 0
sum11 = 0

while j < 10:
    ### Place your code below this line ###




    ### Place your code above this line ###

print('')
print('Total sum11 is:', sum11)
```

Expected output:

```
j:  0  sum11:    0
j:  1  sum11:    11
j:  2  sum11:    33
j:  3  sum11:    66
j:  4  sum11:    110
j:  5  sum11:    165
j:  6  sum11:    231
j:  7  sum11:    308
j:  8  sum11:    396
j:  9  sum11:    495

Total sum11 is:  495
```

Question 5 (1 points)

A machine learning project you are conducting uses a set of ten historical observations stored in a tuple named historical. For each of these historical observations, your machine learning exercise has generated a couple of prediction sets stored respectively in the tuples named predictiona and predictionb.

As a result, you will need to store the historical predictions and the two prediction sets in a paired manner, using the zip() function, in a new paired tuple named topresults. Then, you will need to use a for loop statement to iterate through the zipped tuples to print for each of the historical observations (with prediction pairs) a string in the Python console. For instance, in the first pair, you would have elements historical: 3, predictiona: 1, predictionb: 6. The string that you will need to output in the Python console for each item in this example would be: "historical: 3 prediction a: 1 prediction b: 6". You will need to do the same for the remainder 9 historical observations.

Use the starter code below to get this done and ensure your output complies with the expected output below. Hint: use Python tuple unpacking using a for loop statement.

Starter code:

```
historical = 3, 5, 1, 9, 0, 3, 9, 2, 4, 7
predictiona = 1, 5, 4, 1, 7, 7, 1, 0, 3, 9
predictionb = 6, 0, 4, 3, 4, 4, 8, 4, 3, 7

print('')
print('historical:', historical)
print('predictiona:', predictiona)
print('predictionb:', predictionb)
print('')

### Place your code below this line ###




### Place your code above this line ###
```

Expected output:

```
historical: 3 prediction a: 1 prediction b: 6
historical: 5 prediction a: 5 prediction b: 0
historical: 1 prediction a: 4 prediction b: 4
historical: 9 prediction a: 1 prediction b: 3
historical: 0 prediction a: 7 prediction b: 4
historical: 3 prediction a: 7 prediction b: 4
historical: 9 prediction a: 1 prediction b: 8
historical: 2 prediction a: 0 prediction b: 4
historical: 4 prediction a: 3 prediction b: 3
historical: 7 prediction a: 9 prediction b: 7
```

Question 6 (1 points)

A data science experiment you are conducting has retrieved two historical observations for the price of Bitcoin (BTC) on December 2, 2017 of 11234 and 12475. Create a Python script that stores these two historical observations in a list variable named btcdec1.

Your data science experiment requires four additional data tasks. The first task is to use the list append() method to add the BTC price of 14560 to the list btcdec1. The second task is to create a new empty list named btcdec2 and append the BTC prices of 15630, 12475, and 14972. The third task required you to use the list extend() method to add the contents of btcdec2 into the list btcdec1. The fourth and final task requires you to use the list sort() method of the list btcdec1 to sort the items in the newly extended list, then use the print statement to output the content of list btcdec1 to the Python console.

Starter code:

```
### Place your code below this line ###




### Place your code above this line ###

print(btcdec1)
```

Expected output:

```
[11234, 12475, 12475, 14560, 14972, 15630]
```

Question 7 (1 points)

Write a Python script that counts the number of common items between two lists named list1 and list2. Form the starter code below, check if each item in list1 also appears in list2 and keep track of the number of common items using the variable named cnt. Hint: see in-class exercise 8 from lecture 7 (last slide).

Starter code:

```
list1 = [-4, 2, 7, -6, 3, -5, 8, 10, 4, -10]
list2 = [1, 7, 8, -10, 2, 6, -1, 10, -3, -8]

cnt = 0

for item in list1:
    ### Place your code below this line ###




    ### Place your code above this line ###

print('Number of common items between list1 and list2 is:', cnt)
```

Expected output:

```
Number of common items between list1 and list2 is: 5
```

Question 8 (1 points)

Suppose that you have a collection of *n* numbers: $a_1, a_2, a_3, \cdots, a_{n-1}, a_n$. Formally, the *arithmetic mean* (= what we usually refer to as simply the mean or average) and the *geometric mean* are computed by the following formulas:

- Arithmetic mean: $(a_1 + a_2 + a_3 + \cdots + a_{n-1} + a_n)/n$
- Geometric mean: $a_1 \times a_2 \times a_3 \times \cdots \times a_{n-1} \times a_n)^{\frac{1}{n}}$

The geometric mean is often used in finance to determine the performance results of an investment or portfolio.[1] One example is to compute average growth rates using the geometric mean and is referred to as the *compounded annual growth rate*. For example, consider a stock that grows by 10% in year one, declines by 20% in year two, and then grows by 30% in year three. Then, the geometric mean of the growth rate is calculated as follows:

$$((1 + 0.1) \times (1 - 0.2) \times (1 + 0.3))^{1/3}$$
$$= (1.1 \times 0.8 \times 1.3)^{1/3}$$
$$= 1.144^{1/3}$$
$$= 0.046 \quad \text{(= 4.6\% annually)}$$

Write a Python program that computes the geometric mean of 1.03, 0.9, 1.36, 1.23, 1.08, 1.12, 1.55, 1.06, 1.05 and 0.92, which are stored as a Python list variable named growth_rates. The first step is to multiply all numbers in the list growth_rates and store the result in a variable named mult. You can use a for loop statement to do this step. Alternatively, you can use functions provided by the math package or the NumPy package. The next step is to raise the value of mult to the power of 1/n, where n is the number of items in the list growth_rates. Lastly, round the resulting number to two decimal places and assign the result to a variable named geo_mean.

Starter code:

```
growth_rates = [1.03, 0.9, 1.36, 1.23, 1.08, 1.12, 1.55, 1.06, 1.05, 0.92]

### Place your code below this line ###




### Place your code above this line ###

print('Compounded annual growth rate is:', geo_mean)
```

Expected output:

```
Compounded annual growth rate is: 1.12
```