Programming for Data Science 2020

Homework Assigment One

Please Remember: Homework activities aim, not only, at testing your ability to put into practice the concepts you have learned during the Lectures and Labs, but also your ability to explore the Python documentation as a resource. But above all, it is an opportunity for you to challenge yourself and practice. If you are having difficulties with the assignment reach out for support.

The Homework Assigment One is divided into three parts:

1. Explore the core building blocks of programming, focusing in variables, data structures, and their manipulation;
2. Focuses in data loading and analysis using the core elements of Python, without fancy third-party libraries;
3. Focus in functional programming and will require you to define different functions and operate with the map() and filter();

Comment your code properly, which includes naming your variables in a meaningful manner. Badly documented code will be penalized.

Your submission will be graded according to the following guidelines:

1. Execution (does your program does what is asked from the exercise?)
2. Objectivity (are you using the adequate libraries? are you using a libraries not in the scope of exercise?)
3. Readibility of your code (is your code correctly documented? is variable naming adequate?)

Note: When you are asked to answer a question, for example the highest number in a list, you must answer this question using code. If you just look at the list to find the highest number and write it in the cell you will not recieve any points.

This assignment is Individual, students that are caught cheating will obtain a score of 0 points. The Homeworking Assignment One is worth 15% of your final grade.

The submission package should correspond to a .zip archive (.rar files are not acceptable) with the following files:

1. Jupyter Notebook with the output of all the cells;
2. PDF print of your Jupyter Notebook;
3. All text or csv files exported as part of the exercises.

Submission is done through the respective Moodle activity. Deadline is October 2nd at 23:59. A penality of 1 point per day late will be applied to late deliveries.

Part 1 - Variable Declaration and Manipulation

Exercise I - of Lists and Random numbers

Q: Declare a variable X that stores a list of 100 integers randomly sampled between -25 and 25.
Note: You are not allowed to use Numpy or Scipy in this exercise.

In [1]:

```
import random

def generateRandomNumberList(minValue, maxValue): #function that receives a min and max
value for our
    return [random.randint(minValue, maxValue) for number in range(100)] #storing 100 r
andom numbers between the values that we declare

randomNumbersList = generateRandomNumberList(-25, 25) #storing the random values in our
list
print(randomNumbersList)
```

```
[23, 5, -9, -11, 1, 19, 21, -12, 20, 22, 7, -5, 8, -9, -10, -22, 6, -15,
2, 2, 14, 23, -22, -20, -21, 0, -7, 13, -12, 7, 21, 21, -7, -13, 1, -8, -1
3, 17, -25, -5, -8, -22, -21, -7, -11, 19, -20, 3, -21, -18, 21, -5, -20,
15, 4, 25, 7, -8, 19, 24, -1, 6, 16, 11, -9, -14, -21, 1, 2, 16, -24, 7,
1, 25, -10, -1, 0, 7, -16, -25, -16, 21, -19, 0, 13, 1, 2, -25, 21, -5, 2
1, 9, 9, -17, 6, -21, -8, -7, 5, -22]
```

Q: Write a program to tell you how many odd numbers are in the list X.

Check to make sure you have the same number of odd and even numbers, if not discard the list and generate a new one.

Extra: Can you automatize this pipeline to avoid having to run multiple cells multiple times by hand?

In [2]:

```
countOddNumbers = 0
randomListSize = len(randomNumbersList)

for number in randomNumbersList:
    if(number % 2 == 1):
        countOddNumbers += 1

if(countOddNumbers % (randomListSize / 2) == 0): #If our odd numbers are exactly half o
f our list, we have the same number of odd and even numbers
    print('Our list has the same odd and even numbers!')
else:
    randomNumbersList = generateRandomNumberList(-25, 25) #If the number is not the sam
e, were we can generate another list
    print('Odd and even numbers where diferent. Another list was created')
```

```
Odd and even numbers where diferent. Another list was created
```

Q: Print the number of digits that the 5th and 100th element of the list have.
Note: For instance, the number 1 contains one digit, the number 10 contains two digits. Note: we consider that the number -2 contains one digit.

In [3]:

```
def verifyDigits(value):
    return(len(str(abs(value)))) #We transform our value in absolute number and count t
he number of digits

print('Our 5th element is ' + str(randomNumbersList[4]) + ' and has ' + str(verifyDigit
s(randomNumbersList[4])) + ' digits')
print('Our 100th element is ' + str(randomNumbersList[99]) + ' and has ' + str(verifyDi
gits(randomNumbersList[99])) + ' digits')
```

```
Our 5th element is 25 and has 2 digits
Our 100th element is 10 and has 2 digits
```

Q: Is the total of all the numbers in the list even or odd?

In [4]:

```
if(sum(randomNumbersList) % 2 == 0):
    print('The total of all the numbers in the list is even')
else:
    print('The total of all the numbers in the list is odd')
```

```
The total of all the numbers in the list is odd
```

Q: What is the average of all the numbers in the list? What is the standard deviation?

In [5]:

```
import math

average = sum(randomNumbersList) % len(randomNumbersList)
print('The average of all numbers is ' + str(average))

variance = 0

for number in randomNumbersList:
    variance += (number - average)**2

variance = variance / len(randomNumbersList) - 1

deviation = math.sqrt(variance)
print('The standard deviation is ' + str(round(deviation,4)))
```

```
The average of all numbers is 45
The standard deviation is 47.635
```

Q: Sort list X in descending order and store the result in variable Xsort.
Then replace each value in Xsort with index i as the sum of the values with index i-1 and i.
Note: Per definition Xsort[-1] = 0.

For example the list [1,2,3,4,5] would become [1,3,6,10,15]

In [6]:

```
sortedList = randomNumbersList
sortedList.sort(reverse=True)
print('Sorted list: ' + str(sortedList))

for i in range(len(sortedList)):
    sortedList[i] = sortedList[i] + sortedList[i - 1]

print('Sorted and summed list: ' + str(sortedList))
```

```
Sorted list: [25, 25, 25, 24, 24, 24, 22, 22, 21, 20, 20, 20, 20, 19, 19,
19, 19, 18, 18, 16, 16, 13, 12, 11, 10, 10, 9, 9, 8, 8, 7, 6, 6, 6, 6, 4,
3, 2, 2, 2, 1, 1, 0, 0, -1, -1, -1, -2, -2, -2, -4, -5, -5, -6, -6, -6, -
7, -7, -7, -7, -8, -8, -8, -8, -8, -9, -9, -9, -9, -10, -10, -10, -10, -1
0, -10, -11, -11, -12, -12, -12, -12, -13, -13, -13, -15, -15, -17, -18, -
18, -19, -19, -19, -20, -20, -20, -21, -22, -22, -23, -25]
Sorted and summed list: [0, 25, 50, 74, 98, 122, 144, 166, 187, 207, 227,
247, 267, 286, 305, 324, 343, 361, 379, 395, 411, 424, 436, 447, 457, 467,
476, 485, 493, 501, 508, 514, 520, 526, 532, 536, 539, 541, 543, 545, 546,
547, 547, 547, 546, 545, 544, 542, 540, 538, 534, 529, 524, 518, 512, 506,
499, 492, 485, 478, 470, 462, 454, 446, 438, 429, 420, 411, 402, 392, 382,
372, 362, 352, 342, 331, 320, 308, 296, 284, 272, 259, 246, 233, 218, 203,
186, 168, 150, 131, 112, 93, 73, 53, 33, 12, -10, -32, -55, -80]
```

Exercise II - we have a gamer in the room

Q: Consider the dictionaries purchases and clients that are declared in the cell below.
Create a list with the names of the clients who bought more than one videogame. Print the List.
What is the name of the client that bought the most videogames?

TIP: You will want to check the methods associated with string manipulation. See the links below:
https://python-reference.readthedocs.io/en/latest/docs/unicode/index.html (https://python-reference.readthedocs.io/en/latest/docs/unicode/index.html)
https://www.w3schools.com/python/python_strings.asp
(https://www.w3schools.com/python/python_strings.asp)

In [7]:

```
purchases = {

    "1539":"Red dead redemption II",
    "9843":"GTA V,FarCry 5",
    "8472":"Canis Canem Edit",
    "3874":"Watchdogs II,South Park: The Stick of Truth",
    "5783":"AC: The Ezio Collection",
    "9823":"For Honor,The Forest,South Park: The Fractured but whole"

}
```

In [8]:

```
clients = {

    "1539":"Rick Sanchez",
    "9843":"Morty Smith",
    "8472":"Eve Polastri",
    "3874":"Mildred Ratched",
    "5783":"Alex Vause",
    "9823":"Sheldon Cooper"

}
```

In [9]:

```
nameList = []
mostGamesBought = 0
mostGamesBoughtClient = ''

for purchase in purchases:
    gamesBought = len(purchases[purchase].split(","))

    if(gamesBought >= 2):
        nameList.append(clients[purchase])

    if(gamesBought > mostGamesBought):
        mostGamesBought = gamesBought
        mostGamesBoughtClient = clients[purchase]

print('More than two games bought: ' + str(nameList))
print('The client who bought more games was ' + str(mostGamesBoughtClient))
```

```
More than two games bought: ['Morty Smith', 'Mildred Ratched', 'Sheldon Co
oper']
The client who bought more games was Sheldon Cooper
```

Part 2 - Data loading and analysis

Exercise I - Alice what do you have to say?

Important Make sure that the file alice.txt which was included with this homework is in the same folder where you have your notebook.

Q: Load the Alice text file into a variable called Alice.
Note1: You are not allowed to use third-party libraries like Pandas. Example code can be adapted from lab 2 or you can search the internet for help. Note 2: The first two cells make sure that the alice.txt file was downloaded to the same folder as your notebook, now all you need to do is load it :) !

In [10]:

```
import sys
!conda install --yes --prefix {sys.prefix} requests
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.
```

In [11]:

```
import requests
r = requests.get('https://www.dropbox.com/s/67pt7yaymxdw6up/alice.txt?dl=1')
open('alice.txt', 'wb').write(r.content)
```

Out[11]:

```
306
```

In [12]:

```
#write the code to load the dataset here:
aliceData = open("alice.txt", "r").read()
print(aliceData)
```

```
Alice was beginning to get very tired of sitting by her sister on the ban
k, and of having nothing to do: once or twice she had peeped into the book
her sister was reading, but it had no pictures or conversations in it, <<a
nd what is the use of a book>>. thought Alice <<without pictures or conver
sation?>>.
```

# Q: Create a list in which each element is a word from the file Alice. Store that list in a variable called wAlice. Note: You will need to do some text parsing here. In particular to split the sentences into words. It is also a good practice to normalize words so that words "Hello" and "hello" become identical, by making all letters lower case. Tip: It is possible, and likely easier, to just use the inbuilt python methods available for string manipulation to complete this task. For those interrested one advanaced way to this is by following the links below for a discussion on regular expressions. https://docs.python.org/3/library/re.html https://stackoverflow.com/questions/1276764/stripping-everything-but-alphanumeric-chars-from-a-string-in-python

In [13]:

```
import re
aliceDataString = re.sub(r'\W+', ' ', aliceData) #cleaning the data with a regex

wAlice = aliceDataString.split() #splitting our information into list elements

print(wAlice)
```

```
['Alice', 'was', 'beginning', 'to', 'get', 'very', 'tired', 'of', 'sittin
g', 'by', 'her', 'sister', 'on', 'the', 'bank', 'and', 'of', 'having', 'no
thing', 'to', 'do', 'once', 'or', 'twice', 'she', 'had', 'peeped', 'into',
'the', 'book', 'her', 'sister', 'was', 'reading', 'but', 'it', 'had', 'n
o', 'pictures', 'or', 'conversations', 'in', 'it', 'and', 'what', 'is', 't
he', 'use', 'of', 'a', 'book', 'thought', 'Alice', 'without', 'pictures',
'or', 'conversation']
```

Using the list wAlice answer the following questions:
Q: How many words contains the text file Alice.txt?
Q: What is the longest and smallest word in the textfile? (Length here is measured in terms of the number of characters)
Q: Delete all the repeated words from wAlice. How many different words does the text contain? Hint: there is an easy way to do this in one line of code, but you can answer however you like.

In [14]:

```
print('The file contains ' + str(len(wAlice)) + ' words')

wAliceSorted = wAlice
wAliceSorted.sort(key=len)

print('The smallest word is: ' + str(wAliceSorted[0]))
print('The largest word is: ' + str(wAliceSorted[-1]))

wAliceSet = set(wAliceSorted)

print('Sorted, the file contains ' + str(len(wAliceSet)) + ' word and the list is ' + s
tr(wAliceSet))
```

```
The file contains 57 words
The smallest word is: a
The largest word is: conversations
Sorted, the file contains 41 word and the list is {'or', 'a', 'it', 'get',
'reading', 'nothing', 'to', 'sister', 'do', 'thought', 'having', 'withou
t', 'conversations', 'into', 'conversation', 'and', 'on', 'but', 'the', 'o
nce', 'book', 'twice', 'pictures', 'peeped', 'tired', 'Alice', 'use', 'wa
s', 'bank', 'no', 'she', 'very', 'by', 'what', 'of', 'in', 'beginning', 's
itting', 'is', 'had', 'her'}
```

Exercise II - I Love Economics

Consider the list countries in the cell below.
It consists of a list of the 3-digit ISO codes of a set of countries of interest.

In [15]:

```
countries = ['PAN','ARG','NGA','KOR','SRB','BIH']
```

Consider the list country_data in the cell below. It is a list of tuples where each tuple has the following variables:

The first value is the 3-digit ISO code of a country.
The second value is the name of the country.
The third value is the continent in which the country is localized.
The fourth value is the population size of the country (in millions).

In [16]:

```
country_data = [
 ('arg', 'Argentina', 'SouthAmerica', 41.2238883972168),
 ('geo', 'Georgia', 'Asia', 4.231660842895509),
 ('kor', 'South Korea', 'Asia', 49.5528564453125),
 ('swz', 'Swaziland', 'Africa', 1.2028429508209229),
 ('cog', 'Republic of the Congo', 'Africa', 4.386693000793457),
 ('srb', 'Serbia', 'Europe', 7.291436195373535),
 ('pan', 'Panama', 'NorthAmerica', 3.643222093582153),
 ('ita', 'Italy', 'Europe', 59.72980880737305),
 ('dma', 'Dominica', 'NorthAmerica', 0.07143999636173247),
 ('nga', 'Nigeria', 'Africa', 158.57826232910156),
 ('bih', 'Bosnia and Herzegovina', 'Europe', 3.722084045410156)]
```

Q: Create a dictionary that allows us to map the 3-digit ISO code of a country to it's name.

In [17]:

```
countryDict = {}

for country in country_data:
    countryDict[country[0]] = country[1]

print(countryDict)
```

```
{'arg': 'Argentina', 'geo': 'Georgia', 'kor': 'South Korea', 'swz': 'Swazi
land', 'cog': 'Republic of the Congo', 'srb': 'Serbia', 'pan': 'Panama',
'ita': 'Italy', 'dma': 'Dominica', 'nga': 'Nigeria', 'bih': 'Bosnia and He
rzegovina'}
```

Q: Using country_data, identify what is the most common Continent among the Nations in the list countries.

In [18]:

```
#This exercise could been done using Counter from collections

setContinents = {}

for continent in country_data:
    if continent[2] not in setContinents:
        setContinents[continent[2]] = 1
    else:
        setContinents[continent[2]] += 1

mostCommonContinent = [max(setContinents, key=setContinents.get)]

for continent in setContinents:
    if continent != mostCommonContinent[0] and setContinents[continent] == setContinent
s[mostCommonContinent[0]]:
        mostCommonContinent.append(continent)

if len(mostCommonContinent) == 1:
    print(mostCommonContinent[0] + ' is the most common continent')
else:
    print('The most common continents are:')
    for continent in mostCommonContinent:
        print('- ' + continent)
```

```
The most common continents are:
- Africa
- Europe
```

Q: Using country_data, identify the most populated nation among the countries list.

In [19]:

```
populationDict = {}

for country in country_data:
    populationDict[country[1]] = country[-1]

largestPopulation = max(populationDict, key=populationDict.get)

print(largestPopulation + ' with ' + str(populationDict[largestPopulation]))
```

```
Nigeria with 158.57826232910156
```

Part 3 - Functions hurt nobody

Exercise I - I hate math

Consider the following equation:

$$y = 6x^2 + 3x + 2$$

Q: write a function called f that takes one argument, x, and returns y according to the equation above. Call the function for x = 2 and print the answer.

In [20]:

```
def exerciseOneFunc(value):
    return (6 * value ** 2) + (3 * value) + 2

print(exerciseOneFunc(2))
```

```
32
```

Consider the following sequence of numbers:

$$x_0 = 0;$$
$$x_n = x_{n-1} - 1; \quad if \;\; x_{n-1} - n > 0$$
$$x_n = x_{n-1} + 1; \quad otherwise$$

Q: Write a function that returns the nth digit of the above defined sequence.
Note: the above sequence is also known as the Recamán's sequence, and it was invnted by Bernardo Recamán Santos (Bogotá, Colombia)

In [21]:

```
#recaman sequence adds or subtracts n a nd not 1
def recamanSequence(n):
    if(n == 0):
        return 0
    else:
        if(recamanSequence(n - 1) - n > 0):
            return recamanSequence(n - 1) - n
        else:
            return recamanSequence(n - 1) + n
```

In [22]:

```
print(recamanSequence(5))
```

```
7
```

Exercise II - Role Playing with Python

Q: Create a function (called purchase_calculator) that will be used by a videogames store manager. This function recieves a list of number of codes of videogames, and a dictionary that has all the video games codes as keys and their cost as value.
This function should use the list and dictionary to return the total cost of the purchase.

In [23]:

```
def purchase_calculator(gamesCodeList, gamesDict):
    totalAmount = 0

    for game in gamesCodeList:
        totalAmount += gamesDict[game]

    return totalAmount

codeList = {
    "1539":15.99,
    "9843":69.99,
    "8472":10,
    "3874":5.99,
    "5783":122.4,
    "9823":24.5
}

print('Total amount is ' + str(purchase_calculator(["1539", "9843", "8472", "3874", "57
83", "9823"], codeList)) + '€')
```

```
Total amount is 248.87€
```

Q: Create a function named random_nr that takes a number (e.g., m) as an argument and returns a random number from 1 to m.

In [24]:

```
def random_nr(value):
    return random.randint(1, value)

print(random_nr(10))
```

```
2
```

Q: Create a function named key_returner that takes a number (n) and a dictionary (s) as arguments and returns n random keys from s.

In [25]:

```python
def key_returner(number, dictValues):
    return [random.choice(list(dictValues.items())) for x in range(number)]

returnedList = key_returner(number, clients)

for row in returnedList:
    print(row[0])
```

```
5783
9823
3874
5783
5783
1539
9823
5783
3874
9823
```

Q: Create a function named bool_gen that takes a number (p) as an argument and returns True with probability p, else it returns False.

NOTE: Your function needs to consider weather p is a number from 0 to 1 (and hence is the probability itself) or if the number is between 0 to 100 (and hence is in percentages and needs to be changed to a probability). If the value inserted is inferior to 0 or bigger than 100 the function should assume p to be 0.5

To clarify If p is 0.45 we will have a 45% probability to return True. If p is 55, it needs to be converted to 0.55 (having the same meaning as previously stated : we have a 55% probability of returning true

In [26]:

```python
def bool_gen(prob):
    if(0 < prob > 100):
        prob = 50

    if(random.randint(0, 100) <= prob):
        return True
    else:
        return False

bool_gen(80)
```

Out[26]:

```
True
```

Q: Create a function named apply_discount that receives a number (v) and a boolean (b) as arguments. If b is true then the function should apply a discount of 15% to value, and return it. Otherwise it should return the value as is.

In [27]:

```
def apply_discount(price, discount):
    if(discount):
        return price * 0.85
    return price

apply_discount(20, True)
```

Out[27]:

```
17.0
```

Q: Create a function called customerSim. This function takes as an input two dictionaries (stock and prices). Then it should perform the following steps:

1. Call random_nr to generate a number between 1 and the maximum number of videogames available in the store's stock (this number will represent the number of videogames that will be purchased);
2. Call key_returner, passing the number of videogames that will be purchased obtained from 1. and the stock dictionary, in order to determine what are the codes of the videogames to be purchased;
3. Call apply_discount to each value in the customer order and combined with the boolGen, decide whether you apply a discount or not to each item;
4. Print the following message "Dear customer, your order of Y items has a total cost of X euros, and a 15% discount was applied to W items.", replacing X,Y, W with the respective values.
5. Return a tuple with the number of items ordered, total cost, and the total discount applied.

Run the function customerSim to simulate the orders of 10 clients.

In [28]:

```
#Write your function here. You have the stock and prices dictionaries below :)
def customerSim(stock, prices):
    gamesToBuyList = key_returner(random_nr(len(stock)), stock)

    gamesWithDiscount = 0
    totalCost = 0

    for game in gamesToBuyList:
        discountProb = bool_gen(random_nr(100))

        if(discountProb):
            gamesWithDiscount += 1
            totalCost += apply_discount(prices[game[0]], discountProb)
        else:
            totalCost += prices[game[0]]

    return (round(totalCost,2), len(gamesToBuyList), gamesWithDiscount)
```

In [29]:

```
stock = {

    "234":"God of War",
    "956":"Call of Duty: Modern Warfare 2",
    "365":"Final Fantasy IX",
    "827":"BioShock Infinite",
    "106":"World of Goo",
    "465":"Metal Gear Solid V: The Phantom Pain",
    "622":"Portal 2",
    "266":"The Last of Us",
    "534":"The Legend of Zelda: Majora's Mask",
    "884":"Halo 2",
    "472":"Red Dead Redemption",
    "123":"Grand Theft Auto: San Andreas"

}

prices = {

    "234":49.99,
    "956":35.0,
    "365":68.99,
    "827":20.99,
    "106":2.50,
    "465":49.99,
    "622":19.99,
    "266":19.99,
    "534":5.99,
    "884":9.99,
    "472":19.99,
    "123":24.99

}
```

In [30]:

```
#Call your function here :)
receipt = customerSim(stock, prices)

print('Dear customer, your order of ' + str(receipt[1]) + ' items has a total cost of '
+ str(receipt[0]) + ' euros, and a 15% discount was applied to ' + str(receipt[2]) + '
 items.')
```

```
Dear customer, your order of 11 items has a total cost of 340.75 euros, an
d a 15% discount was applied to 5 items.
```

Exercise II

Q: Consider the list A, below. Use the function map() to change the values in A by adding 1 to each value if it is even and subtracting 1 to it otherwise.

In [31]:

```
A = [460,3347,3044,490,699,1258,1804,973,2223,3416,2879,1058,2915,2422,351,1543,1020,20
8,643,795,3337,2585,471,2623,1077]
```

In [32]:

```
def numberValidator(number):
    if(number % 2 == 0):
        return number + 1
    else:
        return number - 1

AFinal = map(numberValidator, A)

print(list(AFinal))
```

```
[461, 3346, 3045, 491, 698, 1259, 1805, 972, 2222, 3417, 2878, 1059, 2914,
2423, 350, 1542, 1021, 209, 642, 794, 3336, 2584, 470, 2622, 1076]
```

Q: Create a list B with the same size as A, and where each element is True if the associated value in list A is greater than 700, else is False.

In [33]:

```
def numberValidatorBoolean(number):
    if(number > 700):
        return True
    return False

B = map(numberValidatorBoolean, A)

print(list(B))
```

```
[False, True, True, False, False, True, True, True, True, True, True, Tru
e, True, True, False, True, True, False, False, True, True, True, False, T
rue, True]
```

Q: Create a list L that contains the Logarithm of base 10 of each value in A.
Note: You should use the module math to compute the logarithm.

In [34]:

```
L = map(math.log10, A)

print(list(L))
```

```
[2.662757831681574, 3.5246557123577773, 3.4834446480985353, 2.690196080028
514, 2.8444771757456815, 3.09968064110925, 3.256236533205923, 2.9881128402
68352, 3.3469394626989906, 3.5335178620169674, 3.459241664878082, 3.024485
667699167, 3.464638559095033, 3.3841741388070337, 2.545307116465824, 3.188
365926063148, 3.0086001717619175, 2.3180633349627615, 2.808210972924222,
2.9003671286564705, 3.5233562066547925, 3.412460547429961, 2.6730209071288
96, 3.4187982905903533, 3.0322157032979815]
```

Q: How many numbers in A are greater than 1000?
Note: You should use the function filter()

In [35]:

```
greaterThan = len(list(filter(lambda x: x > 1000, A)))

if greaterThan != 0:
    print('There are ' + str(greaterThan) + ' numbers greater than 1000')
else:
    print('It doesn\'t exist any number greater than 1000')
```

```
There are 16 numbers greater than 1000
```

Congratulations, it is done!

Don't forget to post any questions in the Forum or Microsoft Teams.