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 #Importing the random function to generate the random numbers
```

In [2]:
```
#Create an empty list
X = []
#Iterate over 0 to 99 (100 times)
for i in range(0,100):
    #Each time generate a random number and append to the list
    X.append(random.randint(-25,25))
```

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 [3]:
```
"""
    Generate n random numbers defined by the parameter number_of_items,
    between start and end parameters
"""
def create_random_list(number_of_items = 100, start=-25,end=25):
    #Create an empty list
    random_list = []
    #Iterate over 0 to 99 (100 times)
    for value in range(0,number_of_items):
        #Each time generate a random number and append to the list
        random_list.append(random.randint(-25,25))
    return random_list
```

In [4]:
```
"""
    This function returns the count of odd and even numbers on a given list (list_of_numbers), and return both va
lues.
"""
def count_odd_and_even(list_of_numbers):
    odds = 0
    even = 0
    #iterate over the values on the list
    for n in list_of_numbers:
        #if the mod of value is 0 the number is even, so increment the variable
        if n % 2 == 0:
            even = even + 1
        #else, increment the variable even
        else:
            odds = odds + 1
    return odds,even
```

In [5]:
```
"""
    Check how much odds and even numbers are on the list. Use recursivity to calculate this value.
    If the quantity is different keep generating a new list until achieves the same number.
"""
def get_equal_list(list_of_numbers,start=-25,end=25):

    #Get the numbers of the odds and even numbers
    odds,even = count_odd_and_even(list_of_numbers)
    print("The list have ", odds, " odds numbers and ", even, " even numbers.")
    #if the list is odd the number of odds and even are diferent

    list_length = len(list_of_numbers)
    if list_length % 2 != 0:
        #Generate a new list with and even number and use recursivity to get in the same function
        print("The list hasn't the same number of odd and even numbers. A new list will be created.")
        return get_equal_list(create_random_list(list_length + 1))

    if odds != even:
        print("The list hasn't the same number of odd and even numbers. A new list will be created.")
        return get_equal_list(create_random_list(list_length))
    else:
        return list_of_numbers
```

In [6]:
```
get_equal_list(X)
```

```
The list have  55  odds numbers and  45  even numbers.
The list hasn't the same number of odd and even numbers. A new list will be created.
The list have  52  odds numbers and  48  even numbers.
The list hasn't the same number of odd and even numbers. A new list will be created.
The list have  43  odds numbers and  57  even numbers.
The list hasn't the same number of odd and even numbers. A new list will be created.
The list have  51  odds numbers and  49  even numbers.
The list hasn't the same number of odd and even numbers. A new list will be created.
The list have  45  odds numbers and  55  even numbers.
The list hasn't the same number of odd and even numbers. A new list will be created.
The list have  50  odds numbers and  50  even numbers.
```

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

```
7,
-22,
22,
16,
-1,
-22,
17,
1,
4,
3,
-1,
8,
-23,
-23,
10,
-8,
-7,
0,
-1,
0,
-25,
-19,
-20,
12,
-14,
-12,
6,
-8,
-9,
-15,
25,
-20,
8,
-25,
-19,
12,
-3,
-6,
2,
-20,
1,
2,
```

```
-12,
24,
-14,
13,
18,
-15,
19,
-3,
-10,
16,
-3,
-19,
11,
-20,
1,
3,
-15]
```

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 [7]:
```
str_5th_number = X[4]
str_100th_number = X[99]
#Converts the values to string
str_5th_number = str(str_5th_number)
str_100th_number = str(str_100th_number)
```

In [8]:
```
#Print the values using replace to remove the "-" character
print("The 5th number is: ",str_5th_number," and the number of digitis is: ", len(str_5th_number.replace("-",""
)))
print("The 100th number is: " ,str_100th_number, " and the number of digitis  is: ", len(str_100th_number.replace
("-","")))
```

```
The 5th number is:  -11  and the number of digitis is:  2
The 100th number is:  12  and the number of digitis  is:  2
```

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

In [9]:
```
total_numbers = sum(X)
if total_numbers % 2 == 0:
    print("The sum of the numbers (",total_numbers,") on the list is even")
else:
        print("The sum of the numbers (",total_numbers,") on the list is odd")
```

```
The sum of the numbers ( -129 ) on the list is odd
```

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

In [10]:
```
mean_of_X = total_numbers / len(X)
print("The mean of X is: ", mean_of_X)
```

```
The mean of X is:  -1.29
```

In [11]:
```
#Calculate the variance
variance = sum(map(lambda x : (x - mean_of_X) ** 2,X)) / (len(X) - 1)
#Calculate the std
standard_deviation = variance ** (1/2)
print("The standard deviation of the list X is: ", standard_deviation)
```

```
The standard deviation of the list X is:  14.617199637841894
```

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 [12]:
```
"""
Sum each number on a given list with the previous one.
The value on the index -1 will be 0 per definition
"""
def sum_with_previous(list_of_numbers):
    for i,value in enumerate(list_of_numbers):
        if i > 0:
            list_of_numbers[i] = value + list_of_numbers[i-1]
    return list_of_numbers
```

In [13]:
```
Xsort = sorted(X,reverse=True)
sum_with_previous(X)
```

Out[13]:
```
[2,
 -16,
 -14,
 9,
 -2,
 -16,
 -3,
 14,
 38,
 22,
 13,
 12,
 12,
 1,
 17,
 26,
 28,
 30,
 23,
 2,
 4,
 -19,
 -38,
 -41,
 -41,
 -59,
 -64,
 -80,
 -60,
 -81,
 -80,
 -74,
 -61,
 -76,
 -57,
 -58,
 -56,
 -70,
 -76,
 -82,
 -66,
```

```
-89,
-100,
-115,
-115,
-121,
-127,
-130,
-136,
-161,
-170,
-177,
-202,
-206,
-199,
-188,
-177,
-181,
-174,
-176,
-195,
-174,
-168,
-183,
-162,
-137,
-119,
-136,
-141,
-139,
-161,
-186,
-179,
-155,
-177,
-201,
-197,
-180,
-167,
-163,
-168,
-182,
-192,
```

```
-209,
-200,
-219,
-220,
-196,
-219,
-202,
-181,
-159,
-135,
-110,
-103,
-104,
-108,
-127,
-141,
-129]
```

In [ ]:

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 [14]:
```
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 [15]:
```
clients = {

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

}
```

In [16]:
```
#First I filtered the dict to return only the purchases with more than one videogame. I use the split method to find
#this items
purchases_filtered = dict(filter(lambda value: len(value[1].split(","))>1,purchases.items()))
purchases_filtered
```

Out[16]:
```
{'9843': 'GTA V,FarCry 5',
 '3874': 'Watchdogs II,South Park: The Stick of Truth',
 '9823': 'For Honor,The Forest,South Park: The Fractured but whole'}
```

In [17]:
```
#Then I used previous dictionary to find this new buyers with an intersection (after typecast to a filter)
best_buyers = dict(filter(lambda x : set(x).intersection(set(purchases_filtered)), clients.items()))
print("The clients who bought more than one videogame are : ",list(best_buyers.values()))
```

```
The clients who bought more than one videogame are :  ['Morty Smith', 'Mildred Ratched', '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 [18]:
```
import sys
!conda install --yes --prefix {sys.prefix} requests
```

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

EnvironmentNotWritableError: The current user does not have write permissions to the target environment.
  environment location: C:\ProgramData\Anaconda3
```

```
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\ProgramData\Anaconda3

  added / updated specs:
    - requests


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-4.8.5                |           py37_0         2.9 MB
    ------------------------------------------------------------
                                           Total:         2.9 MB

The following packages will be UPDATED:

  conda                                     4.8.2-py37_0 --> 4.8.5-py37_0



Downloading and Extracting Packages

conda-4.8.5          | 2.9 MB    |            |   0%
conda-4.8.5          | 2.9 MB    |            |   1%
conda-4.8.5          | 2.9 MB    | 2          |   3%
conda-4.8.5          | 2.9 MB    | #          |  10%
conda-4.8.5          | 2.9 MB    | ##4        |  24%
conda-4.8.5          | 2.9 MB    | ###        |  30%
conda-4.8.5          | 2.9 MB    | ###6       |  36%
conda-4.8.5          | 2.9 MB    | ####2      |  42%
conda-4.8.5          | 2.9 MB    | #####      |  50%
conda-4.8.5          | 2.9 MB    | #####6     |  56%
conda-4.8.5          | 2.9 MB    | ######3    |  63%
conda-4.8.5          | 2.9 MB    | ######9    |  69%
conda-4.8.5          | 2.9 MB    | #######5   |  76%
conda-4.8.5          | 2.9 MB    | ########3  |  83%
conda-4.8.5          | 2.9 MB    | #########  |  91%
```

```
conda-4.8.5          | 2.9 MB    | #########8 |  99%
conda-4.8.5          | 2.9 MB    | ########## | 100%
Preparing transaction: ...working... done
Verifying transaction: ...working... failed
```

In [19]:
```
import requests
r = requests.get('https://www.dropbox.com/s/67pt7yaymxdw6up/alice.txt?dl=1')
open('alice.txt', 'wb').write(r.content)
```

Out[19]:
```
306
```

In [20]:
```
#write the code to load the dataset here:
with open('alice.txt', 'r') as file:
    data = file.read()
```

In [21]:
```
data
```

Out[21]:
```
'Alice was beginning to get very tired of sitting by her sister on the bank, 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, <<and w
hat is the use of a book>>. thought Alice <<without pictures or conversation?>>.'
```

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://docs.python.org/3/library/re.html)

https://stackoverflow.com/questions/1276764/stripping-everything-but-alphanumeric-chars-from-a-string-in-python (https://stackoverflow.com/questions/1276764/stripping-everything-but-alphanumeric-chars-from-a-string-in-python)

In [22]:
```
#Making all the words lowercase
data = data.lower()
#Remove all  punctuatiosn
data = data.replace("<","").replace(">","").replace(",","").replace(".","").replace(":","")
#Creating a list with each word of the string
wAlice = data.split()
wAlice
```

Out[22]:
```
['alice',
 'was',
 'beginning',
 'to',
 'get',
 'very',
 'tired',
 'of',
 'sitting',
 'by',
 'her',
 'sister',
 'on',
 'the',
 'bank',
 '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',
 'and',
 'what',
 'is',
 'the',
 '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?

In [23]:
```
print("The text file Alice contains ", len(wAlice), " words")
```

```
The text file Alice contains  57  words
```

Q: What is the longest and smallest word in the textfile? (Length here is measured in terms of the number of characters)

In [24]:
```
#We could have more than one word with the same length. So I will show all the smallest and all the longest words
#First I'll find the length of the smallest and the longest word
length_smallest = len(min(wAlice))
length_longest = len(max(wAlice))
```

In [25]:
```
#Then I'll get all the smallest and all the longes words
list_smallest = [w for w in wAlice if len(w) == length_smallest]
list_longest = [w for w in wAlice if len(w) == length_longest]
```

In [26]:
```
print("The smallest words are ", list_smallest)
```

```
The smallest words are  ['a']
```

In [27]:
```
print("The longest words are ", list_longest)
```

```
The longest words are  ['sitting', 'nothing', 'reading', 'thought', 'without']
```

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 [28]:
```
#Sets allows onli unique items. So I typecasted the wAlice list to a Set
wAlice = list(set(wAlice))
print("The text file contains ", len(wAlice), " different words.")
```

```
The text file contains  41  different words.
```

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 [29]:
```
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 [30]:
```
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 [31]:
```
"""For me it's not clear if we have to do this with the first list or with the second. I'm assuming that I have t
o use
the first list. Then I use this  dict comprehension to create our final dict only with the countries of the first
list.
"""
country_dict = {country[0].upper() : country[1] for country in country_data}
country_dict = {country : country_dict[country] for country in countries}
```

In [32]:
```
country_dict
```

Out[32]:
```
{'PAN': 'Panama',
 'ARG': 'Argentina',
 'NGA': 'Nigeria',
 'KOR': 'South Korea',
 'SRB': 'Serbia',
 'BIH': 'Bosnia and Herzegovina'}
```

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

In [49]:
```
#Get only the coninents
continents = [data[2] for data in country_data]
#Then use the parameter key of the max function to compare the count of the values.
most_common = max(set(continents),key=continents.count)
#Find the number of occurences of this continent
max_ocurrences = len(list(filter(lambda x: x == most_common, continents)))

#Then I will count the items per continent. I typecasted the list on a Set so I'll not have duplicates
continent_counts = {continent: continents.count(continent) for continent in set(continents)}
#Then I'll find on the count list all continents with the highest number of ocurrences
most_common_continents = [key for key,value in continent_counts.items() if value ==  max_ocurrences]

print("The most common Continent is ",most_common_continents)
```

```
The most common Continent is  ['Europe', 'Africa']
```

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

In [50]:
```
#Using the max aproach again. This time comparing the value (method get) of the dict
population_data = {data[2]: data[3] for data in country_data}
print("The country with highest poupulation is ", max(population_data,key=population_data.get))
```

```
The country with highest poupulation is  Africa
```

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 [51]:
```
def hate_math_function(x):
    return 6 * (x ** 2) + (3 * x) + 2
```

In [52]:
```
result = hate_math_function(2)
```

In [53]:
```
print(result)
```

```
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 [54]:
```
"""
    Return the nth digit of the Racaman's sequence.
"""
def racaman(n_digit):
    #Empty list where I will add the digitis of the sequence
    result_list = []
    #Use range to iterate from 0 until the required digit
    for n in range(n_digit):
        #If n = 0 then add 0 to the sequence
        if n == 0:
            result_list.append(n)
        #Else, check if the previous digit minus n is more the 0 and if the value is not in the sequence yet
        elif (result_list[n -1] - n) > 0:
            new_digit = (result_list[n -1] - n)
            if not any(filter(lambda x : x == new_digit, result_list)):
                result_list.append(new_digit)
            else:
                #Add the previous digit plus n
                result_list.append((result_list[n -1] + n))
        else:
            result_list.append((result_list[n -1] + n))

    #Return the nth digit of the sequence. For instance: the list begins in 0, so the index is n-1
    return result_list[n_digit -1]
```

In [55]:
```
racaman(5)
```

Out[55]: 2

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 [56]:
```
def purchase_calculator(purchase, videogames):
    #Iterate over the items to find only the value for each videogame and the use the sum function to calculate t
he total
    values = [videogames[item] for item in purchase]
    return sum(values)
```

In [57]:
```
videogames = {1:59.8, 2: 18.0, 3:29.99, 4: 5.35,5:64.5,6:87}
purchase = [1,3,2,2,4]
purchase_calculator(purchase,videogames)
```

Out[57]: 131.14

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 [60]:
```
def random_nr(m):
    return random.randint(1,m)
```

In [61]:
```
random_nr(5)
```

Out[61]: 4

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 [62]:
```
s = {"1" : "01", "2": "02", "3": "03", "4": "04", "5":"05"}
```

In [63]:
```
#Use the choice function from random to return a new key each time
def key_returner(n,s):
    list_keys = list(s.keys())
    random_keys = [random.choice(list_keys) for i in range(0,n)]
    return random_keys
```

In [65]:
```
key_returner(3,s)
```

Out[65]: ['1', '5', '5']

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 [110]:
```
def bool_gen(p):
    #Is greater than 100 or less than 0 return 0.5
    if p > 100 or p < 0:
        p = 0.5
    #1 or 100 = 100%,return true imediately
    if p == 1 or p == 100:
        return True;
    #0 is 0% return false imediately
    elif p == 0:
        return False;
    #Less than 1 multiply by 100 to get the values on the same scale
    if p < 1:
        p = p * 100

    #Create a list from 0 to 100
    list_int = [i for i in range(0,101)]
    #For each value on the list that lower or equal the probability map TRUE, otherwhise FALSE
    bool_list = list(map(lambda x: x <= p,list_int))
    #Random select a value on that list
    return bool_list[random.randint(0,100)]
```

In [119]:
```
bool_gen(0.5)
```

Out[119]:
```
False
```

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 [74]:
```
def apply_discount(v,b):
    if b:
        return v - (v * 15 / 100)
    else:
        return v
```

In [75]:
```
apply_discount(80,True)
```

Out[75]: 68.0

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 [92]:
```
def customerSim(stock, prices):
    #Generate the number of videogames and getting them using the key_returner function
    n_videogames = random_nr(len(stock))
    v_codes = key_returner(n_videogames,stock)

    #Iterate over the videogames that will be buyed an create a list of tuples with the price, and if a discount
 will
    #applied or not. Using a fixed probability of 50%
    games_purchaused_prices = [(prices[code],bool_gen(0.5)) for code in v_codes]

    #For each item on the previous list apply the discount
    purchause_with_discount = [apply_discount(game[0],game[1]) for game in games_purchaused_prices]
    #Calculate the total price with and without discount
    total_cost_with_discount = sum(purchause_with_discount)
    total_cost = sum([item[0] for item in games_purchaused_prices])

    #Calculate the number of items with discount
    number_of_discounted_items = len([item for item in games_purchaused_prices if item[1]])

    print( "Dear customer, your order of ", n_videogames ," items has a total cost of ",total_cost_with_discount,
" euros, and a 15% discount was applied to ",number_of_discounted_items," items.")
    return (n_videogames,total_cost_with_discount,total_cost)
```

In [93]:
```
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 [120]:
```
for i in range(0,10):
    customerSim(stock, prices)
```

```
Dear customer, your order of  7  items has a total cost of  166.786  euros, and a 15% discount was applied to  4
items.
Dear customer, your order of  3  items has a total cost of  62.833  euros, and a 15% discount was applied to  2
items.
Dear customer, your order of  10  items has a total cost of  301.163  euros, and a 15% discount was applied to  3
items.
Dear customer, your order of  12  items has a total cost of  295.139  euros, and a 15% discount was applied to  6
items.
Dear customer, your order of  12  items has a total cost of  268.1225  euros, and a 15% discount was applied to
6  items.
Dear customer, your order of  8  items has a total cost of  175.2875  euros, and a 15% discount was applied to  6
items.
Dear customer, your order of  7  items has a total cost of  102.39449999999998  euros, and a 15% discount was app
lied to  3  items.
Dear customer, your order of  1  items has a total cost of  16.9915  euros, and a 15% discount was applied to  1
items.
Dear customer, your order of  8  items has a total cost of  215.6595  euros, and a 15% discount was applied to  4
items.
Dear customer, your order of  11  items has a total cost of  245.887  euros, and a 15% discount was applied to  9
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 [84]:
```
A = [460,3347,3044,490,699,1258,1804,973,2223,3416,2879,1058,2915,2422,351,1543,1020,208,643,795,3337,2585,471,26
23,1077]
```

In [85]:
```
A = list(map(lambda x : x + 1 if x % 2 == 0 else x - 1,A))
```

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 [86]:
```
B = list(map(lambda x : True if  x > 700 else False,A))
```

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 [87]:
```
import math
L = list(map(lambda x : math.log(x,10),A))
```

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

In [88]:
```
print("There are ", len(list(filter(lambda x : x > 1000, A))), " numbers greaterthan 1000 on the list" )
```

```
There are  16  numbers greaterthan 1000 on the list
```

Congratulations, it is done!
Don't forget to post any questions in the Forum or Microsoft Teams.

In [ ]:
```

```