

# Deep Learning Lab 1

Spring 2018

## Author:

Moe Almansoori

## Objective:

Being familiar with the basic functions and tools of Python by solving simple text processing problems. Such as indexing a string, converting a string to a list separate words, searching and testing some conditions. So, we went through four different tasks.

The objective for each task:

- 1- Creating a password validation tool: This tool must check whether the password satisfies specific conditions which is very important and popular requirement when selecting a password to ensure it is a strong one and it is hard to compromise. This task gave an opportunity to use if statement and how to test if a string contains one of s group of characters.
- 2- Identifying the middle word and the longest word in a string and reversing a string: This task focuses on how to index a string by converting it to a list of words. Here we are dealing with lists and how to access specific element in the list using its position. Other thing is how to iterate within a string in both straight and reverse ways.
- 3- Finding the zero sum for three elements within a list of numbers. This is more like an algorithms problem as we have to iterate this numbers list almost three times to keep testing combinations of three numbers. This task gives about sequential iteration within in specific pattern i.e. selecting two numbers from the list each time and adding it to a third number from the same list and test whether the result equals zero. This application can be useful in many cases like load balancing problems and monitoring a system if its values accedes specific level or margin.
- 4- Searching for common names in two different names lists: Here the program is looking for the common names in two classes enrolment lists. Such application is popular and important such as the one needed in final exams scheduling or task assigning where we need to avoid time or task conflict. The program is using loops to iterate through the loops and the if statement to test the matching possibility.

## **Features:**

The accomplished programs have been written in a simple programming structure and considering the logic sequence in results processing and proper transitioning between the steps. It is considered that when someone else reads the program to understand its commands line by line and or blocks functionality, by clearly separating the tasks and giving descriptive names for the variables and providing descriptive comments for the main parts of the code.

## **Configuration:**

Similar working configuration had been used for all the Lab1 requirements, they are:

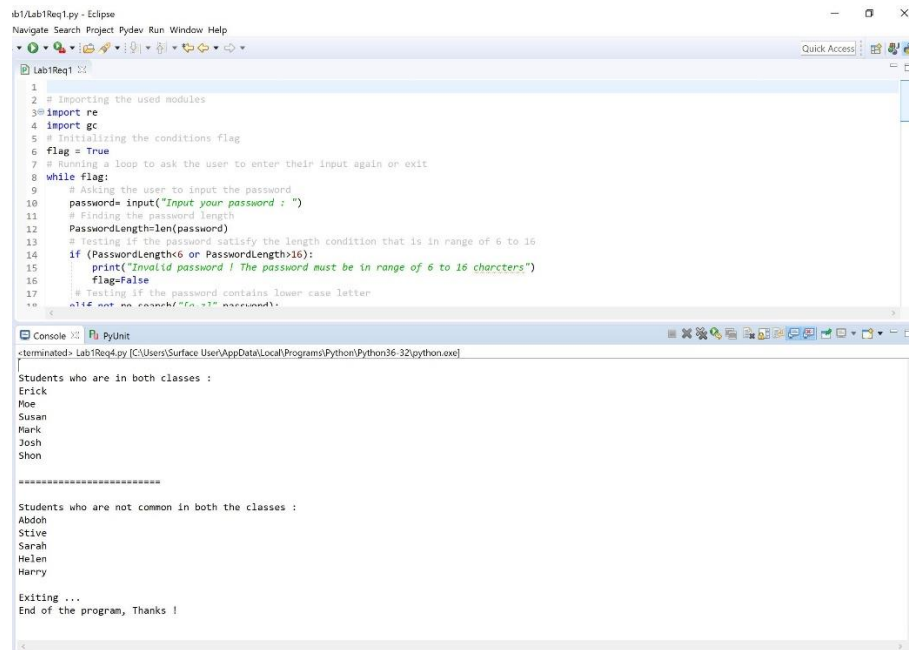
Python 3.6

OS: Windows 10

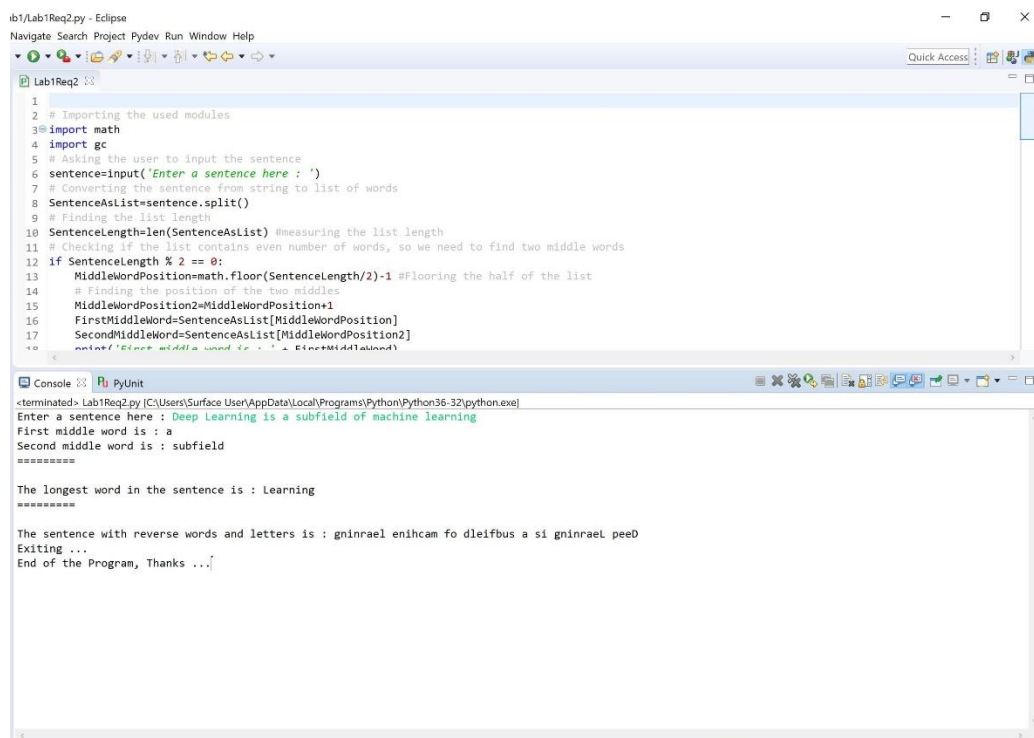
IDE: Eclipse

### Requirement 1:

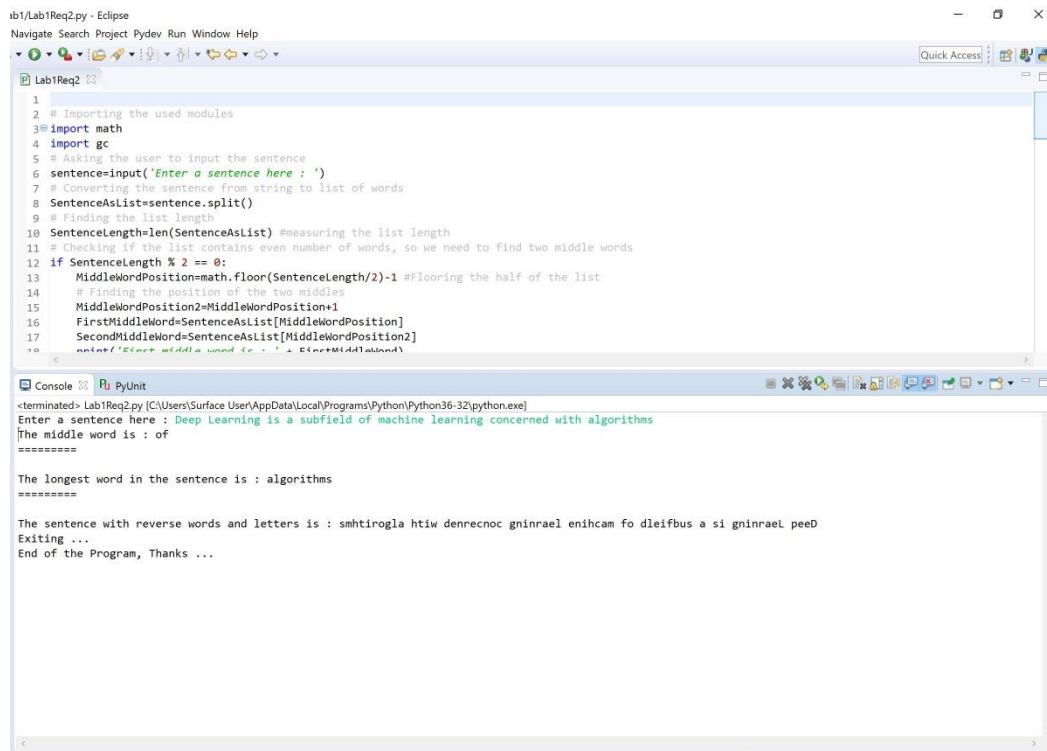
### Requirement 1:



**Equirement 2 (string odd number of words):**



## Requirement 2 (string odd number of words):



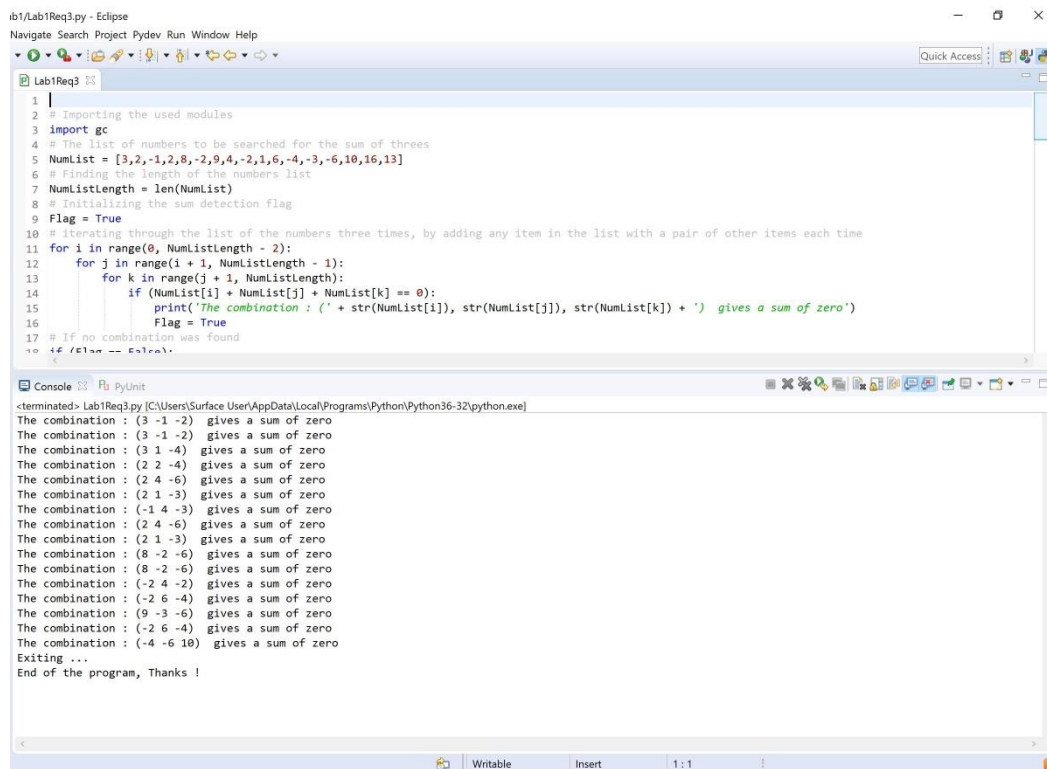
The screenshot shows the Eclipse IDE with a file named 'Lab1Req2.py'. The code in the editor is as follows:

```
1
2 # Importing the used modules
3 import math
4 import gc
5 # Asking the user to input the sentence
6 sentence=input('Enter a sentence here : ')
7 # Converting the sentence from string to list of words
8 SentenceAsList=sentence.split()
9 # Finding the list length
10 SentenceLength=len(SentenceAsList) #measuring the list length
11 # Checking if the list contains even number of words, so we need to find two middle words
12 if SentenceLength % 2 == 0:
13     MiddleWordPosition=math.floor(SentenceLength/2)-1 #Flooring the half of the list
14     # Finding the position of the two middles
15     MiddleWordPosition2=MiddleWordPosition+1
16     FirstMiddleWord=SentenceAsList[MiddleWordPosition]
17     SecondMiddleWord=SentenceAsList[MiddleWordPosition2]
18     enter('Enter middle word is : ' + FirstMiddleWord)
```

The console output shows the program's execution:

```
<terminated> Lab1Req2.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]
Enter a sentence here : Deep Learning is a subfield of machine learning concerned with algorithms
The middle word is : of
=====
The longest word in the sentence is : algorithms
=====
The sentence with reverse words and letters is : smhtirogla htiw denrecnoc gnirrael enihcam fo dleifbus a si gnirrael peeD
Exiting ...
End of the Program, Thanks ...
```

## Requirement 3:



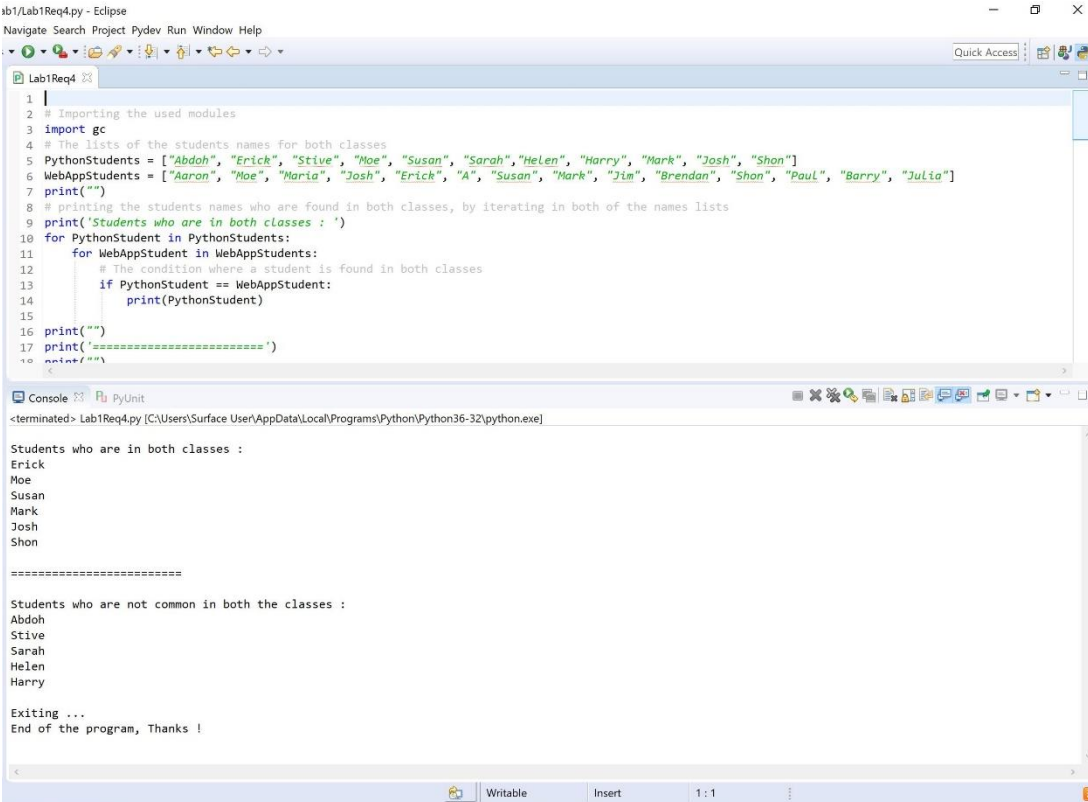
The screenshot shows the Eclipse IDE with a file named 'Lab1Req3.py'. The code in the editor is as follows:

```
1
2 # Importing the used modules
3 import gc
4 # The list of numbers to be searched for the sum of threes
5 NumList = [3,2,-1,2,8,-2,9,4,-2,1,6,-4,-3,-6,10,16,13]
6 # Finding the length of the numbers list
7 NumListLength = len(NumList)
8 # Initializing the sum detection flag
9 Flag = True
10 # Iterating through the list of the numbers three times, by adding any item in the list with a pair of other items each time
11 for i in range(0, NumListLength - 2):
12     for j in range(i + 1, NumListLength - 1):
13         for k in range(j + 1, NumListLength):
14             if (NumList[i] + NumList[j] + NumList[k] == 0):
15                 print('The combination : (' + str(NumList[i]), str(NumList[j]), str(NumList[k]) + ') gives a sum of zero')
16                 Flag = True
17 # If no combination was found
18 if (Flag == False):
```

The console output shows the program's execution:

```
<terminated> Lab1Req3.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]
The combination : (3 -1 -2) gives a sum of zero
The combination : (3 -1 -2) gives a sum of zero
The combination : (3 1 -4) gives a sum of zero
The combination : (2 2 -4) gives a sum of zero
The combination : (2 4 -6) gives a sum of zero
The combination : (2 1 -3) gives a sum of zero
The combination : (-1 4 -3) gives a sum of zero
The combination : (2 4 -6) gives a sum of zero
The combination : (2 1 -3) gives a sum of zero
The combination : (8 -2 -6) gives a sum of zero
The combination : (8 -2 -6) gives a sum of zero
The combination : (-2 4 -2) gives a sum of zero
The combination : (-2 6 -4) gives a sum of zero
The combination : (9 -3 -6) gives a sum of zero
The combination : (-2 6 -4) gives a sum of zero
The combination : (-4 -6 10) gives a sum of zero
Exiting ...
End of the program, Thanks !
```

## Requirement 4:



The screenshot shows the Eclipse IDE with a file named 'Lab1Req4.py'. The code in the editor identifies common students between two lists and prints them. The console output shows the results of the program execution.

```
1 |
2 | # Importing the used modules
3 | import gc
4 | # The lists of the students names for both classes
5 | PythonStudents = ["Abdoh", "Erick", "Stive", "Moe", "Susan", "Sarah", "Helen", "Harry", "Mark", "Josh", "Shon"]
6 | WebAppStudents = ["Aaron", "Moe", "Maria", "Josh", "Erick", "A", "Susan", "Mark", "Jim", "Brendan", "Shon", "Paul", "Barry", "Julia"]
7 | print("")
8 | # printing the students names who are found in both classes, by iterating in both of the names lists
9 | print('Students who are in both classes : ')
10 | for PythonStudent in PythonStudents:
11 |     for WebAppStudent in WebAppStudents:
12 |         # The condition where a student is found in both classes
13 |         if PythonStudent == WebAppStudent:
14 |             print(PythonStudent)
15 |
16 | print("")
17 | print('=====')
18 | print("")
```

Console Output:

```
<terminated> Lab1Req4.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]

Students who are in both classes :
Erick
Moe
Susan
Mark
Josh
Shon

=====

Students who are not common in both the classes :
Abdoh
Stive
Sarah
Helen
Harry

Exiting ...
End of the program, Thanks !
```

## The implementation including code snippet

### Requirement 1:

The program asks the user to enter their password to test if it contains lower case letters, upper case letters, numbers, and special characters. Also, the program tests whether the password length is in the range of 6 to 16 characters.

The program is running an interactive while loop which keeps asking the user to exit after finishing the test or testing another password.

Finally, the program frees the memory that has been used

Below is the code described in more detail:

```
# Importing the used modules
import re
import gc
# Initializing the conditions flag
flag = True
# Running a loop to ask the user to enter their input again or exit
while flag:
    # Asking the user to input the password
    password= input("Input your password : ")
    # Finding the password length
    PasswordLength=len(password)
    # Testing if the password satisfy the length condition that is in range of 6 to
16
    if (PasswordLength<6 or PasswordLength>16):
        print("Invalid password ! The password must be in range of 6 to 16
characters")
        flag=False
    # Testing if the password contains lower case letter
    elif not re.search("[a-z]",password):
        print("Invalid password ! The password must contain a Lower case Letter")
        flag=False
    # Testing if the password contains upper case letter
    elif not re.search("[A-Z]",password):
        print("Invalid password ! The password must contain an upper case Letter")
        flag=False
    # Testing if the password contains numbers
    elif not re.search("[0-9]",password):
        print("Invalid password ! The password must contain a number")
        flag=False
    # Testing if the password contains special character
    elif not re.search("[$@!*]",password):
        print("Invalid password ! The password must contain one special character")
        flag=False
    # If none of the above tests had met then we have to set the flag value to True
    which means the password met all the requirements
    elif flag==True:
        print("Great ! it is a valid password")
```

```

    # Loop control to see if the user wants to check another password or just
    quitting the program
    print('Do you want to check another password? y = YES, n NO and exit')
    decision=input('Your answer: ')
    if decision=='n':
        flag=False
        print('Thank you for using Password Checker! Exiting ...')
    else:
        flag=True
        #break
# Freeing up the used resources
del password
gc.collect()
print("Password Checker is closed")

```

## **Requirement 2:**

The program asks the user to enter their string and then it converts this string to a list of words using split() function. To find the middle word in this word list the program counts the words by finding the list length and divide it by 2. When the list length is odd we will ceil/round (ignoring the division reminder) the result and this result will be the position of the middle word. However, if the length of the list is a even number then we will be having two middles; upper and lower. The lower is the word of the position of the half of the list length and the upper middle is the one if the next position to the lower.

To find the longest word in this list we use max() function. Finally, to reverse the string, we iterate it from the end to the beginning be using the negative index. Lastly, the program frees the memory that has been used

Below is the code described in more detail:

```

# Importing the used modules
import math
import gc
# Asking the user to input the sentence
sentence=input('Enter a sentence here : ')
# Converting the sentence from string to list of words
SentenceAsList=sentence.split()
# Finding the list length
SentenceLength=len(SentenceAsList) #measuring the list length
# Checking if the list contains even number of words, so we need to find two middle
words
if SentenceLength % 2 == 0:
    MiddleWordPosition=math.floor(SentenceLength/2)-1 #Flooring the half of the list
    # Finding the position of the two middles
    MiddleWordPosition2=MiddleWordPosition+1
    FirstMiddleWord=SentenceAsList[MiddleWordPosition]
    SecondMiddleWord=SentenceAsList[MiddleWordPosition2]
    print('First middle word is : ' + FirstMiddleWord)

```

```

    print('Second middle word is : ' + SecondMiddleWord)
    print('=====')
    print('')
else:
    # If the number of the words in the sentence list is odd, then there will be only
    one middle word
    MiddleWordPosition=SentenceLength/2
    MiddleWord=SentenceAsList[math.floor(MiddleWordPosition)]
    print('The middle word is : ' + MiddleWord)
    print('=====')
    print('')

# Finding the word of the maximum length
print('The longest word in the sentence is : ' + max(sentence.split(), key=len))
print('=====')
print('')
# Printing the sentence in reverse order
print('The sentence with reverse words and letters is : ' + sentence[::-1])
print('Exiting ...')
# Freeing up the used resources
del sentence
del SentenceAsList
del SentenceLength
gc.collect()
print('End of the Program, Thanks ...')

```

### **Requirement 3:**

Here the program is searching for any three numbers in a list of number in which their sum is zero. The program uses three loops to iterate through the list by selecting two numbers each time and add them to all of the list numbers and test whether their sum is equal to zero.

Below is the code described in more detail:

```

# Importing the used modules
import gc
# The list of numbers to be searched for the sum of threes
NumList = [3,2,-1,2,8,-2,9,4,-2,1,6,-4,-3,-6,10,16,13]
# Finding the length of the numbers list
NumListLength = len(NumList)
# Initializing the sum detection flag
Flag = True
# iterating through the list of the numbers three times, by adding any item in the
list with a pair of other items each time
for i in range(0, NumListLength - 2):

```



```

    for j in range(i + 1, NumListLength - 1):
        for k in range(j + 1, NumListLength):
            if (NumList[i] + NumList[j] + NumList[k] == 0):
                print('The combination : (' + str(NumList[i]), str(NumList[j]),
str(NumList[k]) + ') gives a sum of zero')
                Flag = True
# If no combination was found
if (Flag == False):
    print(" No combination can give a sum of zero")
print('Exiting ...')
# Freeing up the used resources
del NumList
del NumListLength
del i
del j
del k
gc.collect()
print('End of the program, Thanks !')
```

#### **Requirement 4:**

Here the program is testing two groups of student's names that are enrolled in two different classes. It searches for the common students who have been enrolled for both classes. Also, it searches for the student that are not taking both classes.

The program is using tow loops to compare each student name in the first group with all the names of the second group.

Below is the code described in more detail:

```

# Importing the used modules
import gc
# The lists of the students names for both classes
PythonStudents = ["Abdoh", "Erick", "Stive", "Moe", "Susan", "Sarah", "HeLen",
"Harry", "Mark", "Josh", "Shon"]
WebAppStudents = ["Aaron", "Moe", "Maria", "Josh", "Erick", "A", "Susan", "Mark",
"Jim", "Brendan", "Shon", "Paul", "Barry", "Julia"]
print("")
# printing the students names who are found in both classes, by iterating in both of
the names lists
```

```

print('Students who are in both classes : ')
for PythonStudent in PythonStudents:
    for WebAppStudent in WebAppStudents:
        # The condition where a student is found in both classes
        if PythonStudent == WebAppStudent:
            print(PythonStudent)

print("")
print('=====')
print("")
print('Students who are not common in both the classes : ')
# Initializing the detection indicator value
Indicator=0
for PythonStudent in PythonStudents:
    for WebAppStudent in WebAppStudents:
        if PythonStudent == WebAppStudent:
            # If each name in the first students group has a match in the second
            # students group
            # then we set the indicator to 1 to not include this name in the not
            # common list of names
            Indicator=1
            # If a match discovered (indicator=1) then we will not print this name
            # and we will reset the indicator back to its original value (0) to compare
            # the next name from the first group with all the names of the second group
            if Indicator==1:
                Indicator=0
            # If no match found then this student is not common in both classes, so we print
            # his/her name
            elif Indicator==0:
                print(PythonStudent)

print("")
print('Exiting ...')
# Freeing up the used resources
del PythonStudents
del WebAppStudents
gc.collect()
print('End of the program, Thanks !')

```

## Explain about the deployment

It started with understanding the problem and figuring out what a good way to provide the required results. Then writing down a pseudo code and what is the input type and format. I was trying to avoid using any unnecessary programming structures and seek the simplicity. The startup program can be containing unnecessary loops or variables, but with several testing and polishing the code, the programs became more effective and easy to follow and understand.

## Limitation

The programs that have been developed for this lab purpose still need more improvements to serve in real life environment. For example, the password program needs to check if a space has been used and give a warning for that. Other thing is these programs do not save the results permanently such as in a file.

## References

- 1- <https://docs.python.org/3/reference/import.html>
- 2- <https://docs.python.org/3/library/functions.html#max>
- 3- <https://pymotw.com/2/gc/>
- 4- <https://docs.python.org/3/library/gc.html>
- 5- <https://github.com/tensorflow/tensorflow/issues/1727>

