2018 Spring: COMP-SCI 5590/490 - Special Topics

Python Programming

Lab Assignment 1

Assignment Overview

The following assignment focus on to make one familiar with python basic topics.

Lab Assignment

1) For any web application login, the user password need to be validated against database rules. For My UMKC web application following are the criteria for valid password:

a) The password length should be in range 6-16 characters
b) Should have at least one number
c) Should have at least one special character in [$@!*]
d) Should have at least one lowercase and at least one uppercase character

Use loops to write a python program for the above scenario.

The following code was developed to validate a password against the specified criteria.

```
#Lab 1, Question 1
#Program to enter password that meets password criteria

#Rules
#Password should have a length between 6 and 16 characters
#Password should have at least one number.
#Password should have at least one special character
#Password should have at least one lowercase and at least one uppercase character

#Load re module
import re

#Set Check variable for While loop
Check=1

#Print password requirements
print("Your password should be between 6 and 16 characters.\n"\
        "Your password should have at least one number.\n"\
        "Your password should have at least one special character [$@!*].\n"\
        "Your password should have at least one lowercase and one uppercase character.\n")

#Loop until password meets criteria.
while Check==1:
```

```
    #Enter password
    Password=input("Please enter your password.\n")

    #Determine password length
    PasswordLength=len(Password)
    if PasswordLength <6 or PasswordLength > 16:
        print("Password does not contain between 6 and 16 characters.")

    #Determine if password contains a number
    elif not re.search("[0-9]", Password):
        print("Password does not contain a number.\n")

    # Determine if password contains a lowercase letter
    elif not re.search("[a-z]", Password):
        print("Password does not contain a lowercase letter.\n")

    # Determine if password contains an uppercase letter
    elif not re.search("[A-Z]", Password):
        print("Password does not contain an uppercase letter.\n")

    # Determine if password contains a special character
    elif not re.search("[$@!*]", Password):
        print("Password does not contain one of the special characters $, @, !, or *.\n")

    else:
        print("Password meets the security requirements.\n")
        break
```

The following shows the results of the program run.

```
"D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 1\venv\Scripts\python.exe" "D:/Google
Drive/UMKC/PhD/Classes/Python/Labs/Lab 1/Lab1Q1.py"
Your password should be between 6 and 16 characters.
Your password should have at least one number.
Your password should have at least one special character [$@!*].
Your password should have at least one lowercase and one uppercase character.

Please enter your password.
Wat3r!
Password meets the security requirements.


Process finished with exit code 0
```

The screen printouts for this question are illustrated in the following two screenshots. The first screenshot illustrates the run where the password meets the requirements. The second screenshot illustrates the run where the passwords do not meet requirements.

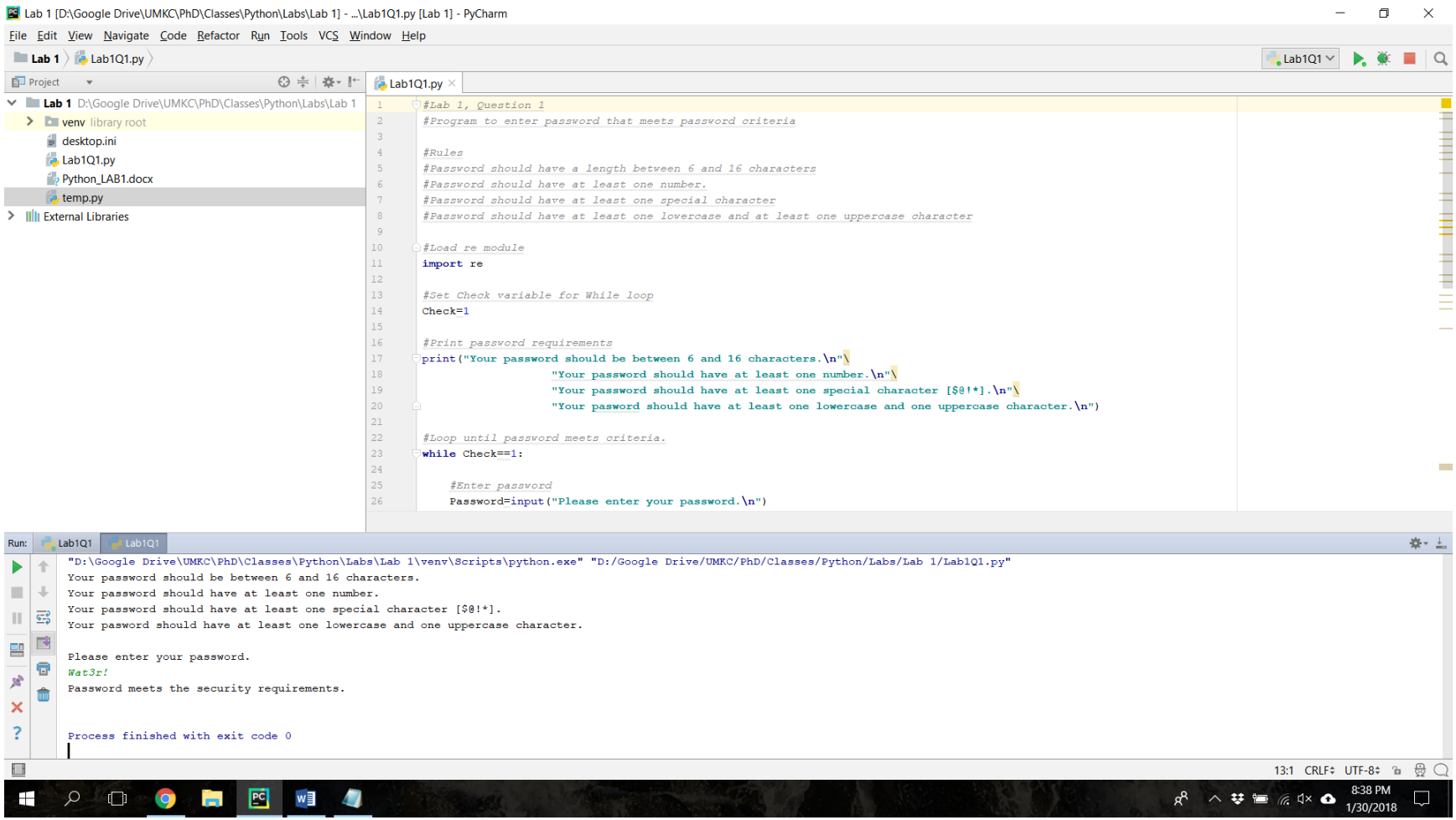

Figure 1 *Screenshot Illustrating Password Meeting Requirements*

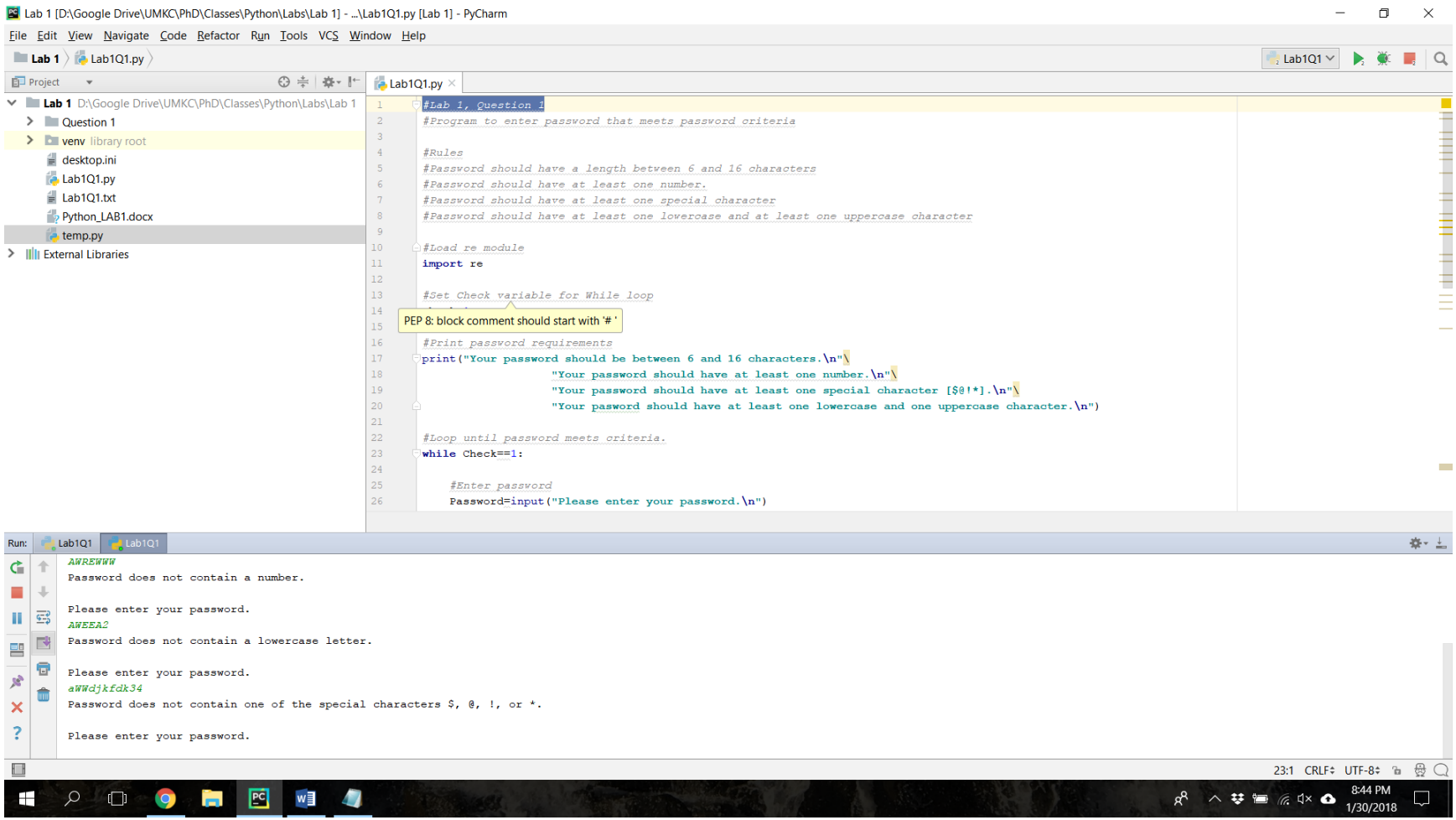

Figure 2 Screenshot Illustrating Password Not Meeting Requirements

2) Write a Python function that accepts a sentence of words from user and display the following:

a) Middle word
b) Longest word in the sentence
c) Reverse all the words in sentence

Sample input:

```
My name is Jacqueline Fernandez Dsouza
```

Sample output:

Middle words are : [is,Jacqueline]

Longest word is : Jacqueline

Sentence with reverse words is: ym eman si enileuqcaj zednanref azuosd

The following code was developed to validate a password against the specified criteria.

```
#Lab 1, Question 2
#Program that accepts sentence entry from user and finds middle word(s), longest word and reverses the sentence.

#Enter Sentence
Sentence=input("Please enter a sentence. Please omit punctuation.\n")

#Split sentence into a word list
Words=Sentence.split()

#Determine the middle word(s)
#Detemine middle position of the list
x=len(Words)

#Middle point of string
MidPosition=x/2

#Determine if number of words is even or odd based on remainder
Mod=int(x%2)

#Print middle words in the sentence based on even or odd number
#Even number of words
if (Mod==0):
   MidPoint=int(MidPosition-1)
   MidPoint1=MidPoint+2
   print("The middle words are: ",Words[MidPoint],", ",Words[MidPoint1],".")

#Odd number of words.
else:
```

```
    MidPoint=int(MidPosition+0.5)
    print("The middle word is ",Words[MidPoint],".")

#Determine longest word in sentence.

#Sort words based on length
SortedWords = sorted(Words, key=len)

#Print the longest word which is the last one in the list.
print("The longest word in the list is: %s.\n" % (SortedWords[-1],))

#Print the sentence with words in reverse order.
print("The sentence with words in reverse order is: ",Sentence[::-1])
```

The results of the run for this program are shown below.

```
"D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 1\venv\Scripts\python.exe" "D:/Google
Drive/UMKC/PhD/Classes/Python/Labs/Lab 1/Lab1Q2.py"
Please enter a sentence. Please omit punctuation.
My name is Jacqueline Fernandez Dsouza
The middle words are:  is ,  Fernandez .
The longest word in the list is: Jacqueline.

The sentence with words in reverse order is:  azuosD zednanreF enileuqcaJ si eman yM

Process finished with exit code 0
```

The screen printouts for this question are illustrated in the following two screenshots. The first screenshot illustrates the run where the sentence as an even number of words. The second screenshot illustrates the run where the sentence has an odd number of words.

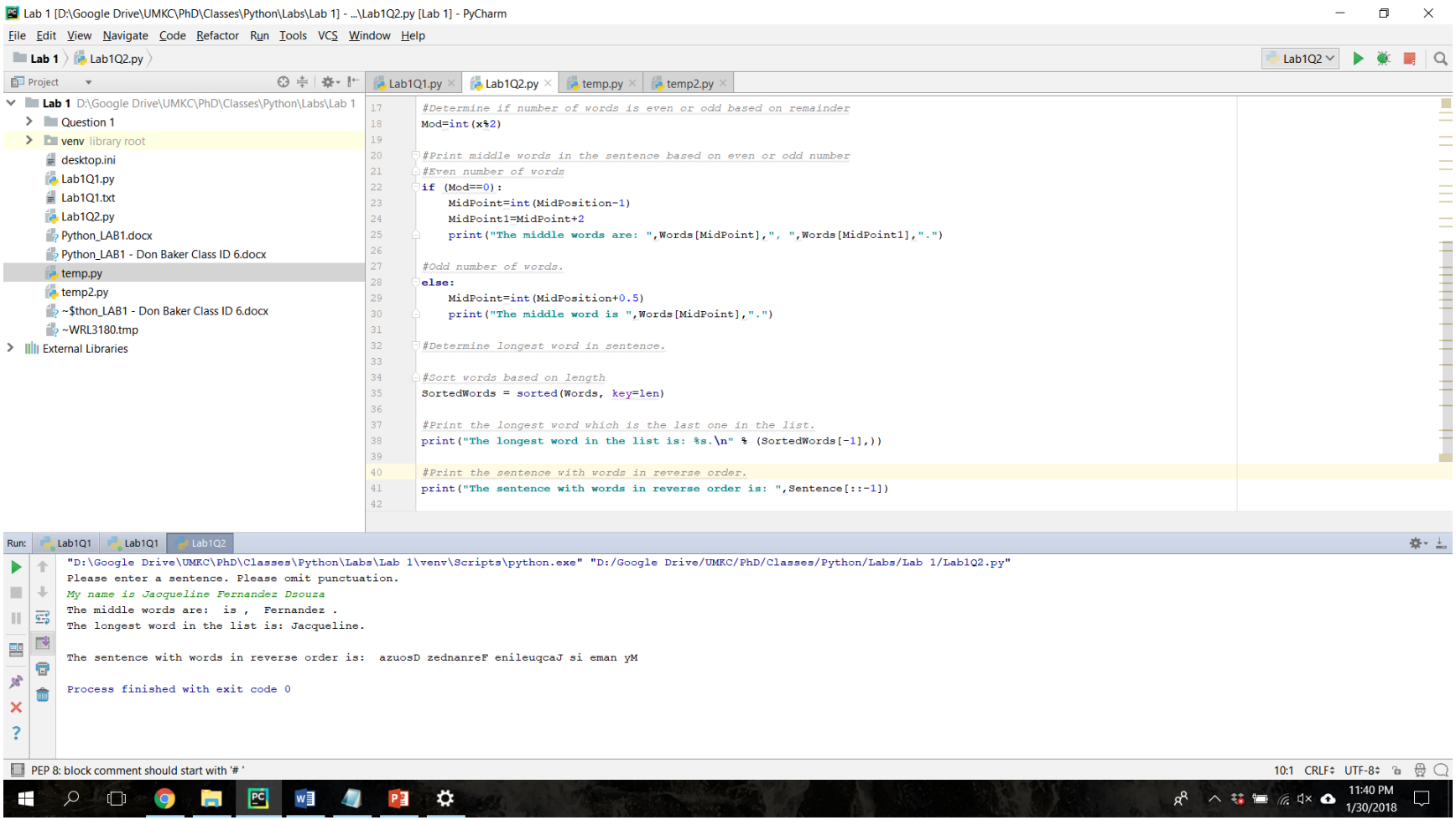

Figure 3 Screenshot for Sentence with Even Number of Words

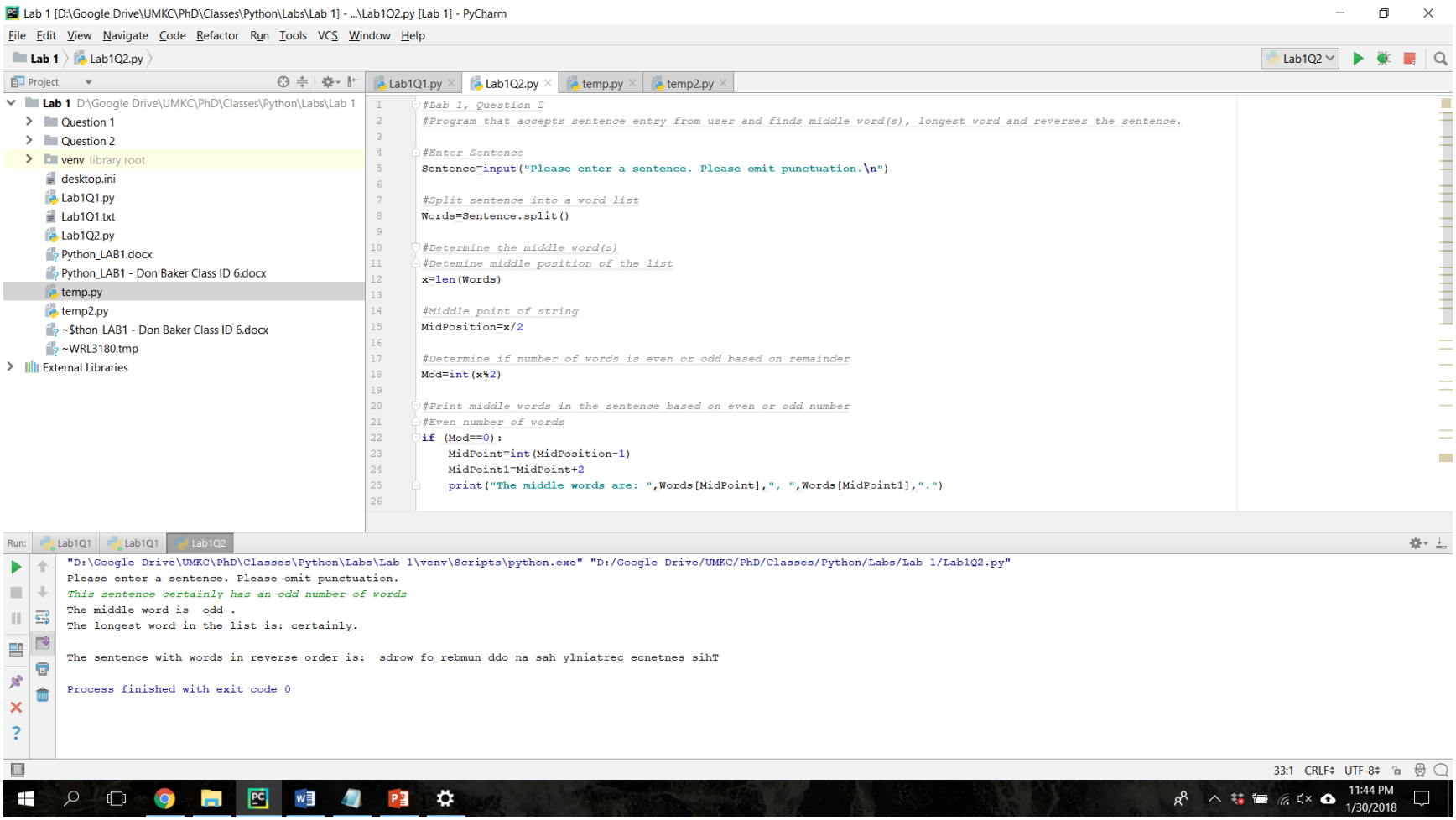

Figure 4 Screenshot of Sentence with Odd Number of Words

3) Given a list of n number, write a Python program to find triplets in the list which gives the sum of zero.

Sample input:

```
[1, 3, 6, 2, -1, 2, 8, -2, 9 ]
```

Sample output:

[(3,-1,-2)]

The code to provide the triples that add to zero from a given list of integers is shown below.

```
#Lab 1, Question 3
#Program that finds triples that add to zero from a given list of integers.

#Define list.
List=[1, 3, 6, 2, -1, 2, 8, -2, 9]

#Find Triples that add to zero.

#Length of list.
Length=int(len(List))

#Loop for first set of numbers
for x in range(0,Length-3):

    #Loop for second set of numbers
    for y in range(x+1,Length-1):

        #Loop for third set of numbes
        for w in range(y+1,Length-1):
            z1=List[x]
            z2=List[y]
            z3=List[w]

            #Check to see if triples add to zero
            if (z1+z2+z3)==0:

                #Print triples
                print("[",z1,",",z2,",",z3,"]")
```

The results from this program are shown below.

```
"D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 1\venv\Scripts\python.exe" "D:/Google
Drive/UMKC/PhD/Classes/Python/Labs/Lab 1/Lab1Q3.py"
[ 3 , -1 , -2 ]
Process finished with exit code 0
```

The screenshot from the run of this program is shown in the next figure.

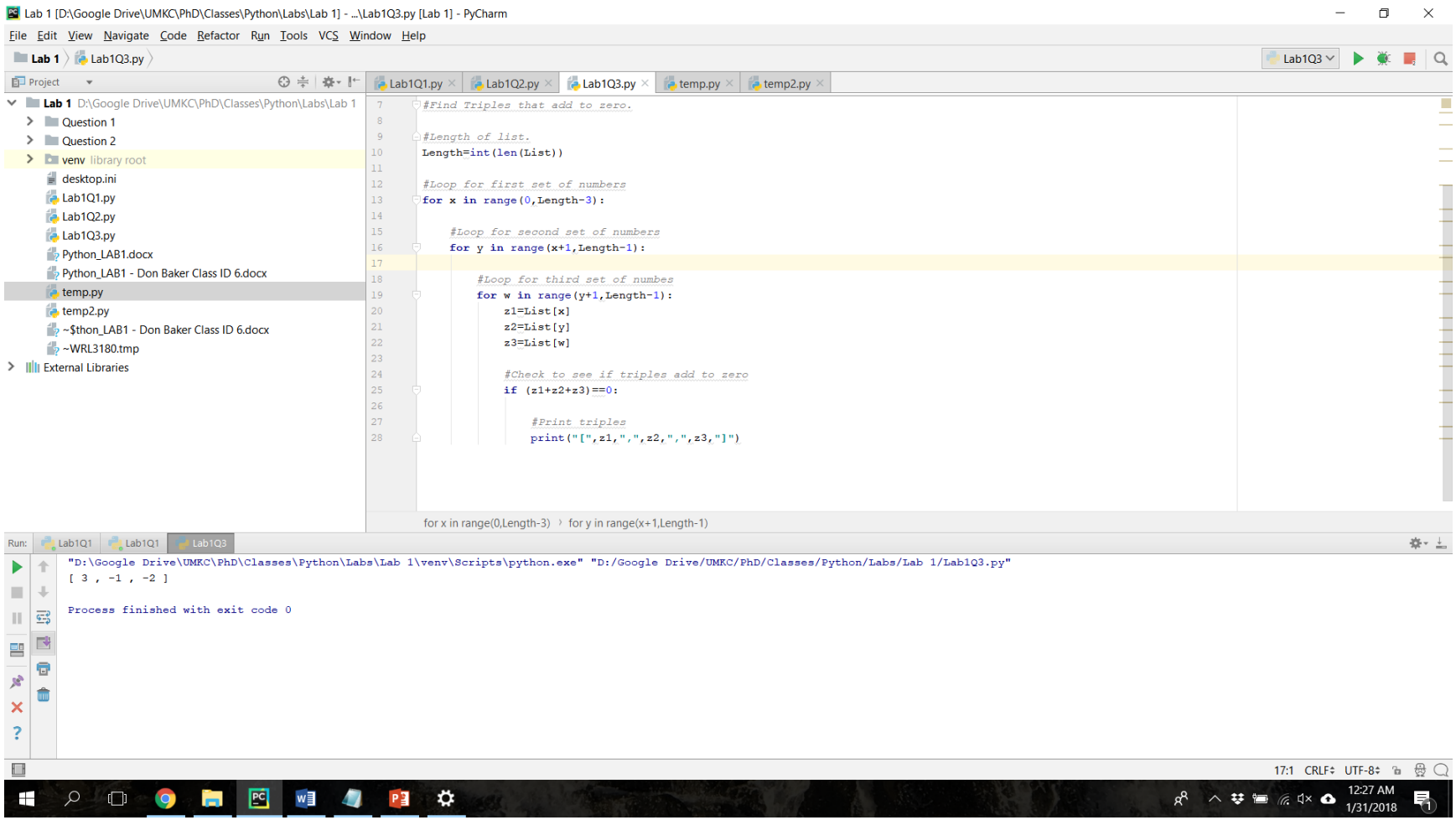

Figure 5 Screenshot of the Program to Find Triples that Add to Zero for a Given List of Integers

4) Consider the following scenario. You have a list of students who are attending class “Python” and another list of students who are attending class “Web Application”.

Find the list of students who are attending both the classes. Also find the list of students who are not common in both the classes. Print it.

The code to compare two lists of students in shown below.

```
#Program to compare the students in two lists

#Students in Python list
Python=["Donald","Kelli","Jack","Roger","Kim","Shelly","Paul"]

#Students in Web Application list
WebApplication=["Jon","Paul","Lisa","Pam","Roger","Kelli","Jane","Bob"]

#Print list of students in Python class.
print("The students in the Python class are:")
print(Python)
print()

#Print list of students in Web Application class.
print("The students in the Web Application class are:")
print(WebApplication)
print()

#Length of lists
PythonLength=int(len(Python))
WebAppLength=int(len(WebApplication))

#Define common and uncommon student lists
Common = []
Uncommon = []

#Define counter
z=0

#Define temporary lists
PythonTemp=[]
WebAppTemp=[]

#Loop to compare lists
for x in range (0,PythonLength):
    for y in range (0,WebAppLength):
        if Python[x]==WebApplication[y]:
            Common.append([Python[x]])


#Create list of uncommon students
#Compare lists using list comprehension
UncommonPython=[w for w in Python if w not in WebApplication]
UncommonWebApp=[z for z in WebApplication if z not in Python]

Uncommon = UncommonPython + UncommonWebApp

print("The common students between both classes are:")
print(Common)
print()
```

```
print("The students that are uncommon between the two classes are:")
print(Uncommon)
```

The results form the program are shown below.

"D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 1\venv\Scripts\python.exe" "D:/Google Drive/UMKC/PhD/Classes/Python/Labs/Lab 1/Lab1Q4.py"

The students in the Python class are:

['Donald', 'Kelli', 'Jack', 'Roger', 'Kim', 'Shelly', 'Paul']

The students in the Web Application class are:

['Jon', 'Paul', 'Lisa', 'Pam', 'Roger', 'Kelli', 'Jane', 'Bob']

The common students between both classes are:

[['Kelli'], ['Roger'], ['Paul']]

The students that are uncommon between the two classes are:

['Donald', 'Jack', 'Kim', 'Shelly', 'Jon', 'Lisa', 'Pam', 'Jane', 'Bob']

Process finished with exit code 0

The screenshot for the program run is shown in the next figure.

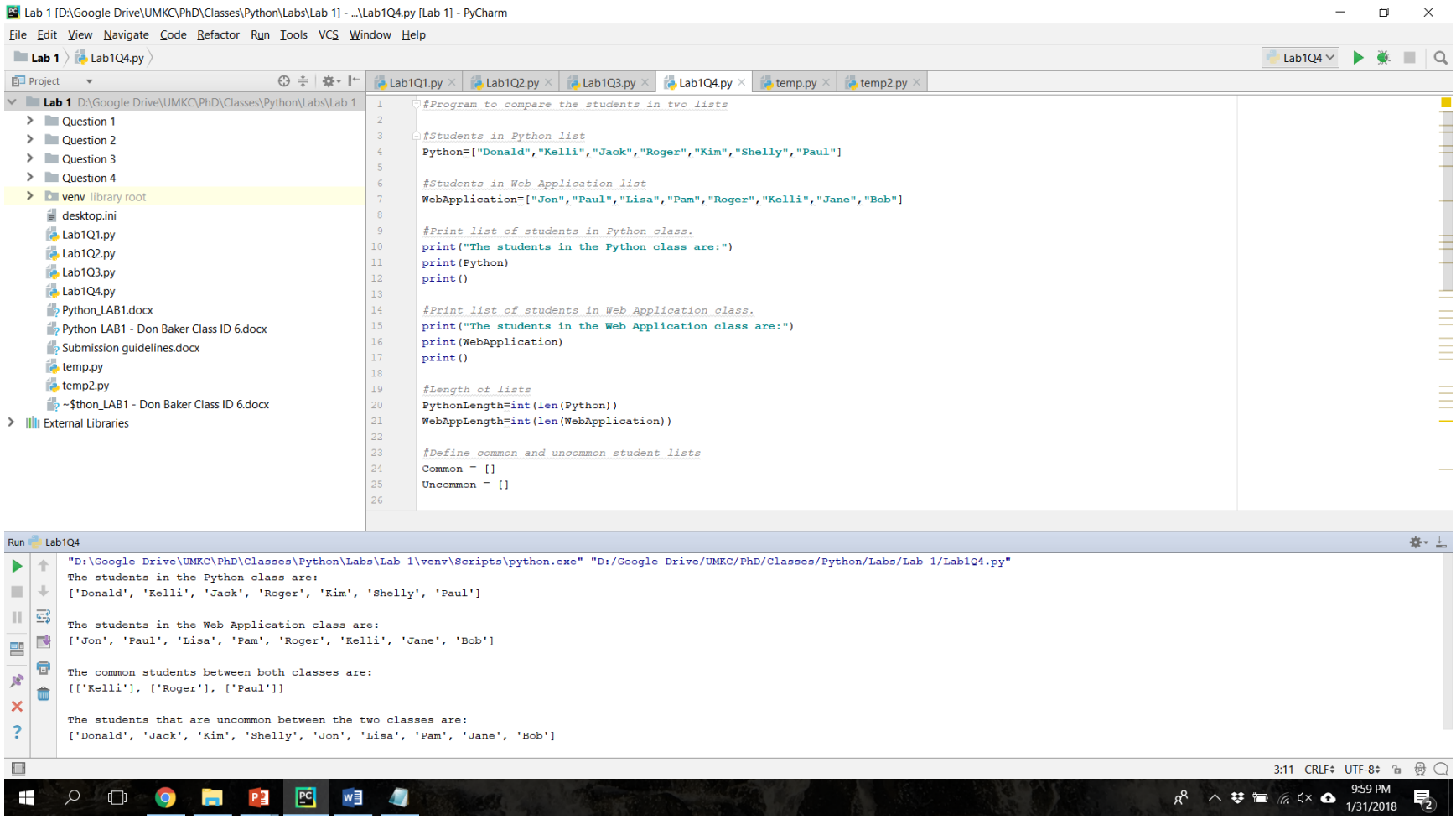

Figure 6 Screenshot for the Program Comparing the Names of the Students in Two Classes