CSEE 5590/490: Python and Deep Learning Programming (2018 Fall)

PYTHON LAB ASSIGNMENT 1

Group ID: 1

Team member: Alex Acsenvil (ID: 1)

Aihua Zhu (ID 32)

Submitted on: 09/19/2018

Department of Computer Science and Electrical Engineering

COTENTS

AUTHOR

This report is submitted by Alex Acsenvil (ID: 1) and Aihua Zhu (ID: 32). Both are 1st year graduate student majoring in computer science at University of Missouri –Kansas City. This report is a part of lab assignment 1 for CS 5590/490 Python and Deep learning taught by coordinated by Dr. Yungyug Lee, instructed by Saria Coudarzvand with Sravanthi Gogadi and Bhargavi Nadendla as the teaching assistant.

OBJECTIVE

The objective of this lab is to utilize python programming, including its introduction, to step through a series of concepts such as looping through strings and files, with if and conditional statements, nested loops, lists, tuples and dictionaries. Further we will leverage the Python framework to remove commonly used words via the NLP (Natural Language Processing) Stop Removal mechanism in comparing multiple files. We also look into set differences, and displaying elements of those sets. Additionally, we focus on building a management system for a hospital admission system, whereby classes, their inheritance, and relationships are used. And finally this lab culminates into retrieving table data from the web using the beautiful soup library into a file. This lab is segmented in five parts:

- 1, 2, and 3 are using Python Introduction, inclusive of loops, list, tuples, and set differences.
- 4 dives deeper into Object oriented concepts such as classes to construct a management system used within a hospital.
- 6 uses existing python library such as beautiful soup, to parse table data from the web into a file.

FEATURES

Key features involved in this lab are documented below. We use the Python commenting syntax to describe the code. This commenting syntax starts with a # (number sign key) followed by the

comment. We also use triple quotes (“‘ ”’) to denote the multiple line comment e.g. “‘comment”’.

Task 1:

The objective of this task:

a) Input from the user is unlimited in terms of number of characters. This is a sentence that takes any character.
b) The input ends once the user press enter or carriage return, marking the end of the input.
c) Once input is received, we count the occurrence of each character in the sentence (input), then display the very first non-repeated character to the user.

Task 2:

The purpose of using Python for this task is to:

a) Take the contents of two related files using lists and or tuples
b) Extracting the common elements of the two files, in other words file A has contents which are also part of file B. A Venn diagram list the elements common to both files, however;
c) The Stop Removal feature of Python is leveraged to output the contents of file A, having extracted those elements common to file B.

Task 3:

With a list of students taking two different course work we will:

a) Find the list of student enrolled in one class but not the other
b) We use the if function to move through the two lists, and print the selected students who are part of list A (Python) but not list B (Web Application).

Task 4:

In this task, we develop a python program using:

a) Classes, relationships, and inheritance to
b) Further build a hospital management system inclusive of:

c) Various roles such as doctors, nurses, patients, etc.

d) Demonstrating mastery of:

   a. multiple classes

   b. Initializing a class

   c. Creating and making a super call

   d. Using the self

   e. A private data member

   f. Relationships between classes.

Task 6:

In this task:

a) We are downloading table data from the web and output that into a file.

b) We use the beautiful soup library to parse the table content.

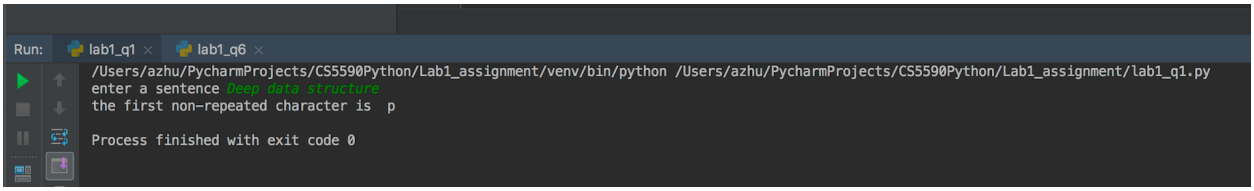
CONFIGURATION

The advanced version of Python 3.6.4 was used for the programming. PYCHARME 2018 (Community Edition) software was used to build the code on macOS.

INPUT/OUTPUT SCREENSHOTS

Task 1:

A sentence “Deep data structure” is taken from the user as input. The first non-repeated character in the string was searched and printed out as shown in the following screenshot.



Task 2:

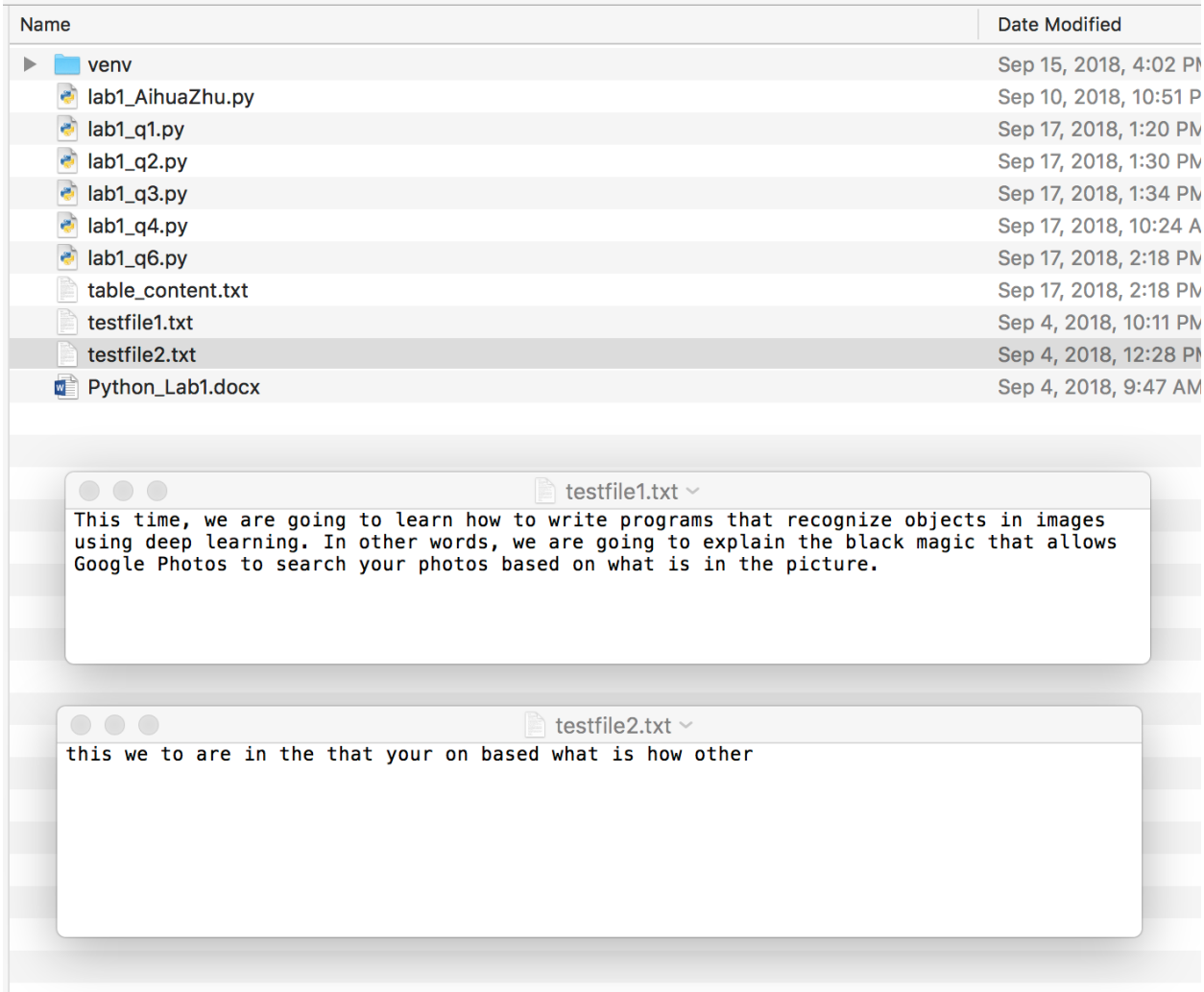
The input is taken from two given text files: testfile1.txt and testfile2.txt. The screenshot shows the content of the given files.

The content of testfile1.txt is:

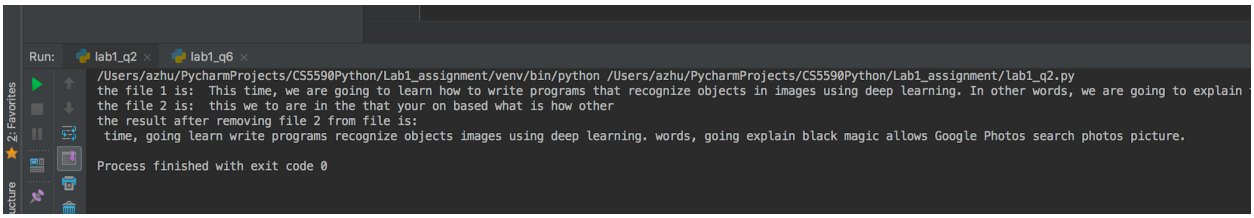
“This time, we are going to learn how to write programs that recognize objects in images using deep learning. In other words, we are going to explain the black magic that allows Google Photos to search your photos based on what is in the picture”

The content in testfile2.txt is:

“this we to are in the that your on based what is how other”



Every word that is in the testfile1.txt is searched and if it is not in the testfile2.txt, it will be put into the output list difstr. The results after removing everything in the testfile2.txt from testfile1.txt is “time, going learn write programs recognize objects images using deep learning. words, going explain black magic allows Google Photos search photos picture” as shown in the following screenshot.



Task 3:

Two lists are given in the code as the input. Plist is the list of students who are attending “python” class. WAlist is the list of students who are attending “Web Application” class.

```
# Q3. Find the list of students who are attending “python” classes but not “Web Application”

# list of student in "python class"
Plist=['Anna','Adam','Ben','Chloe','Dan','Ella','Fin','Greg','Henry','Ian','Jake','Micheal','Monica','Nancy','Peter','Steve','Ted','Zach']
# list of student in "Web Application" class
WAlist=['Adam','Ben','Blake','Dan','Ella','Fin','Greg','Ian','Jake','Luke','Micheal','Monica','Paul','Peter','Steve','Zoe']
print("student in python class: \n",Plist)
print("student in Web Application class: \n",WAlist)
inPnotWAlist=[]
for i in Plist:
    if i not in WAlist:
        inPnotWAlist.append(i)
print('the list of students who are attending python not web application are: \n',inPnotWAlist)
```

Every item in Plist was checked. If that item satisfies the condition “not in the WAlist”, it will be put into the list for the students who are in “Python” class but not “Web Application” class as shown in the following screenshot.

Task 4:

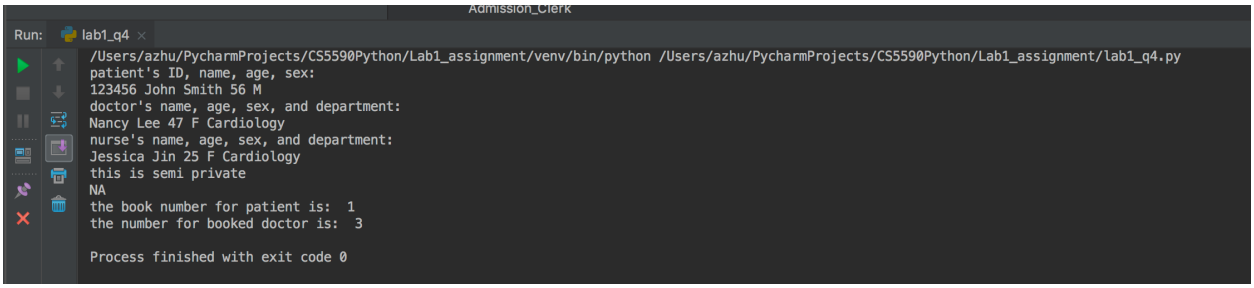
Different instances were given to test functions defined in 7 different classes, which are Patient, Doctor, Nurse, Admission_Clerk, Book, BookedPatient, BookedDoctor as shown in the following.

```
xp=Patient(123456,"John Smith",56,"M")
xp.get_patient_information()

yd=Doctor("Nancy Lee",47,"F","Cardiology")
yd.get_doctor_information()

an=Nurse("Jessica Jin",25,"F","Cardiology")
an.get_nurse_information()

nc=Admission_Clerk()
#print(nc.__superprivate_admission)     # this is to test if we can access the super private data from outside
print(nc._semiprivate_admission)

cb=Book("Nancy Lee","Cardiology","NA")
print(cb.availability)

zp=BookedPatient(123456,"John Smith",56,"M",0)
zp.get_patient_book_number()

sd=BookedDoctor("Eric Johnson",45,"M","ICU",3)
sd.get_doctor_book_number()
```

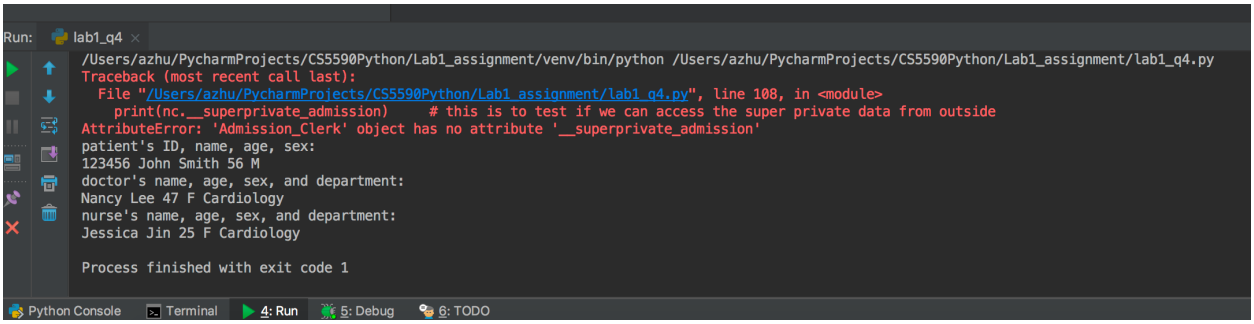
In order to test the accessibility of private data from outside, one semi private data (._semiprivate_admission) and one super private data (.__superprivate_admission) were defined in the Admission_Clerk class.

```
class Admission_Clerk:
    """A class to represent Admission Clerk"""
    def __init__(self):
# this is a semi private data with 1 leading under-score in its name
        self._semiprivate_admission = "this is semi private"
# this a super private data with 2 leading under-scores in its name, which can't be seen and accessed from outside
        self.__superprivate_admission = "private, not for public"
```

The following output showed that semi private data (._semiprivate_admission) can be seen from the outside.

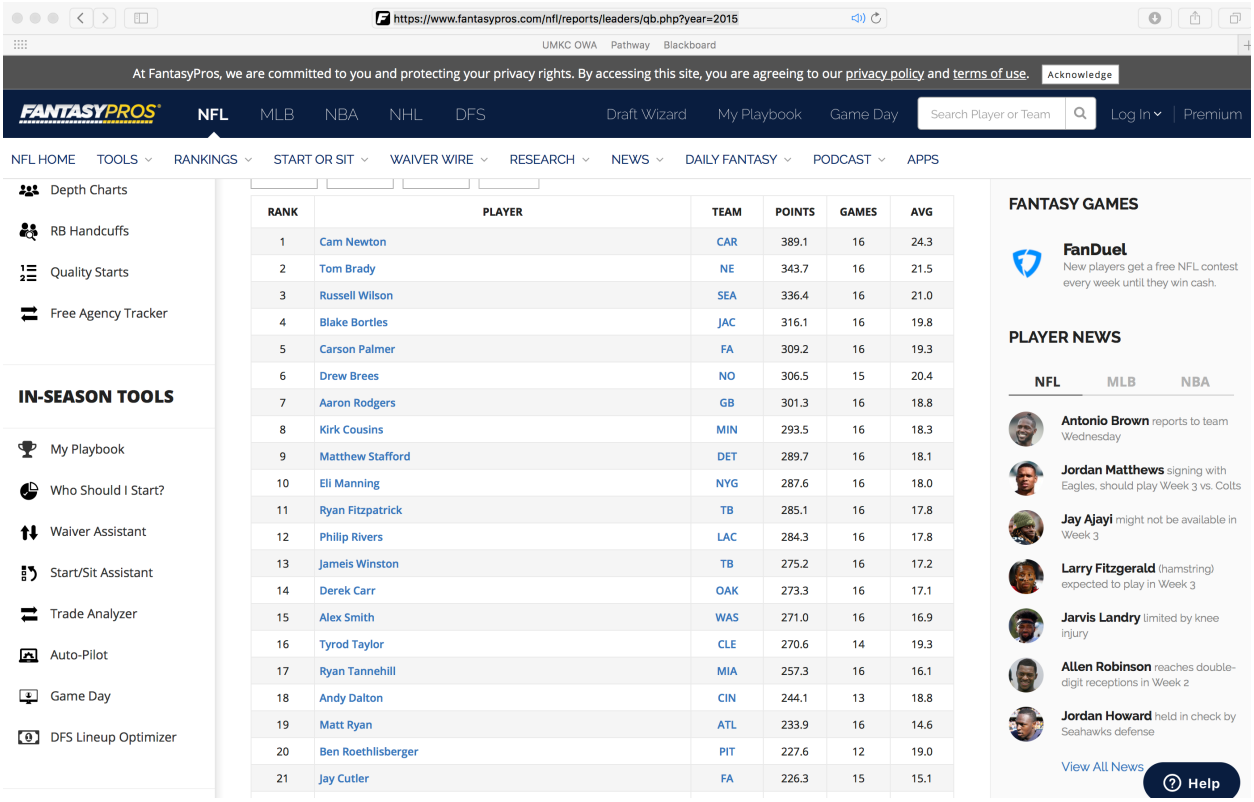


If we access the super private data (.__superprivate_admission) from the outside, attribute error message will show as following because super private data cannot be accessed from outside.

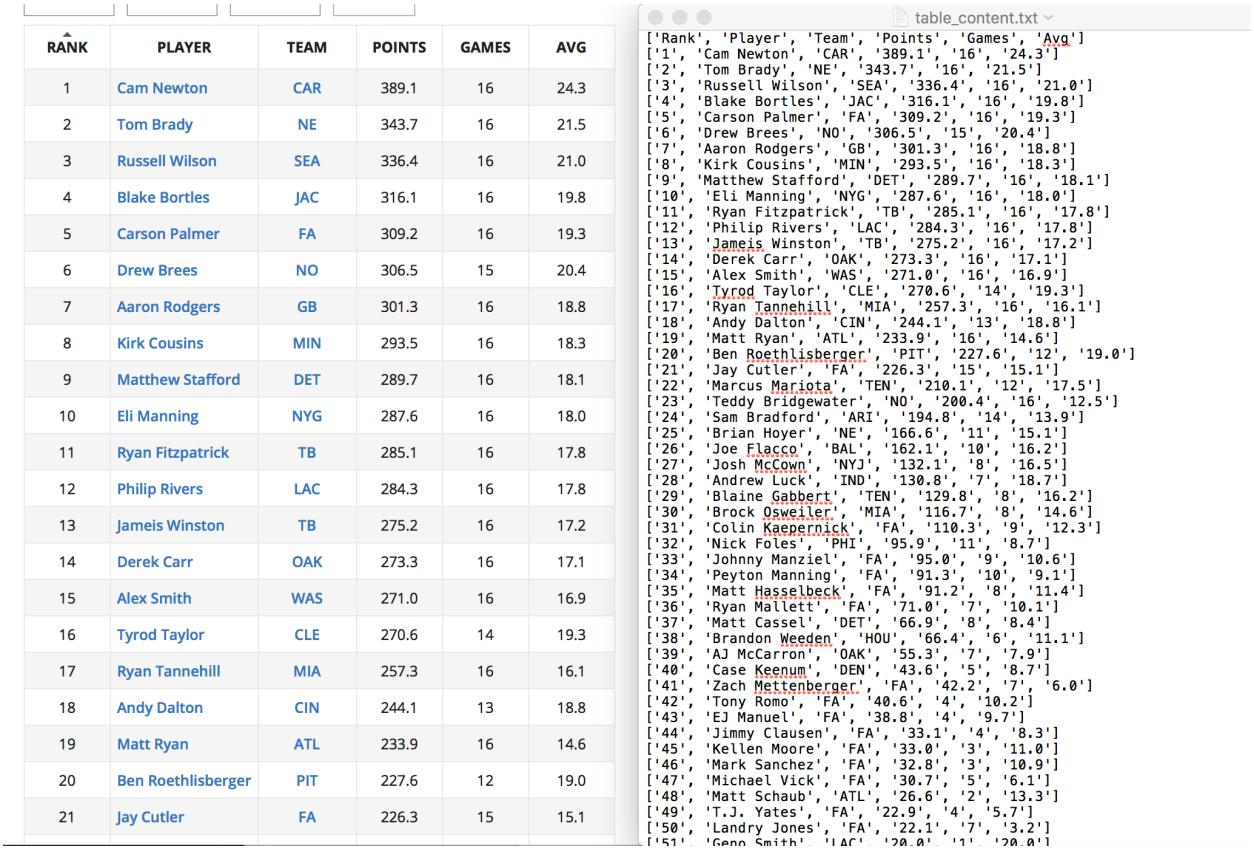


Task 6:

The website https://www.fantasypros.com/nfl/reports/leaders/qb.php?year=2015 is given as the input url.



BeautifulSoup library is imported and employed to parse the page. The object with a div Tag and class: “mobile-table” was returned to the list result_list using the .findAll( ) method. The table header and table content were extracted and written into the output file table_content.txt and printed out the same time as shown in the following.



RANK	PLAYER	TEAM	POINTS	GAMES	AVG
1	Cam Newton	CAR	389.1	16	24.3
2	Tom Brady	NE	343.7	16	21.5
3	Russell Wilson	SEA	336.4	16	21.0
4	Blake Bortles	JAC	316.1	16	19.8
5	Carson Palmer	FA	309.2	16	19.3
6	Drew Brees	NO	306.5	15	20.4
7	Aaron Rodgers	GB	301.3	16	18.8
8	Kirk Cousins	MIN	293.5	16	18.3
9	Matthew Stafford	DET	289.7	16	18.1
10	Eli Manning	NYG	287.6	16	18.0
11	Ryan Fitzpatrick	TB	285.1	16	17.8
12	Philip Rivers	LAC	284.3	16	17.8
13	Jameis Winston	TB	275.2	16	17.2
14	Derek Carr	OAK	273.3	16	17.1
15	Alex Smith	WAS	271.0	16	16.9
16	Tyrod Taylor	CLE	270.6	14	19.3
17	Ryan Tannehill	MIA	257.3	16	16.1
18	Andy Dalton	CIN	244.1	13	18.8
19	Matt Ryan	ATL	233.9	16	14.6
20	Ben Roethlisberger	PIT	227.6	12	19.0
21	Jay Cutler	FA	226.3	15	15.1

table_content.txt

```
['Rank', 'Player', 'Team', 'Points', 'Games', 'Avg']
['1', 'Cam Newton', 'CAR', '389.1', '16', '24.3']
['2', 'Tom Brady', 'NE', '343.7', '16', '21.5']
['3', 'Russell Wilson', 'SEA', '336.4', '16', '21.0']
['4', 'Blake Bortles', 'JAC', '316.1', '16', '19.8']
['5', 'Carson Palmer', 'FA', '309.2', '16', '19.3']
['6', 'Drew Brees', 'NO', '306.5', '15', '20.4']
['7', 'Aaron Rodgers', 'GB', '301.3', '16', '18.8']
['8', 'Kirk Cousins', 'MIN', '293.5', '16', '18.3']
['9', 'Matthew Stafford', 'DET', '289.7', '16', '18.1']
['10', 'Eli Manning', 'NYG', '287.6', '16', '18.0']
['11', 'Ryan Fitzpatrick', 'TB', '285.1', '16', '17.8']
['12', 'Philip Rivers', 'LAC', '284.3', '16', '17.8']
['13', 'Jameis Winston', 'TB', '275.2', '16', '17.2']
['14', 'Derek Carr', 'OAK', '273.3', '16', '17.1']
['15', 'Alex Smith', 'WAS', '271.0', '16', '16.9']
['16', 'Tyrod Taylor', 'CLE', '270.6', '14', '19.3']
['17', 'Ryan Tannehill', 'MIA', '257.3', '16', '16.1']
['18', 'Andy Dalton', 'CIN', '244.1', '13', '18.8']
['19', 'Matt Ryan', 'ATL', '233.9', '16', '14.6']
['20', 'Ben Roethlisberger', 'PIT', '227.6', '12', '19.0']
['21', 'Jay Cutler', 'FA', '226.3', '15', '15.1']
['22', 'Marcus Mariota', 'TEN', '210.1', '12', '17.5']
['23', 'Teddy Bridgewater', 'NO', '200.4', '16', '12.5']
['24', 'Sam Bradford', 'ARI', '194.8', '14', '13.9']
['25', 'Brian Hoyer', 'NE', '166.6', '11', '15.1']
['26', 'Joe Flacco', 'BAL', '162.1', '10', '16.2']
['27', 'Josh McCown', 'NYJ', '132.1', '8', '16.5']
['28', 'Andrew Luck', 'IND', '130.8', '7', '18.7']
['29', 'Blaine Gabbert', 'TEN', '129.8', '8', '16.2']
['30', 'Brock Osweiler', 'MIA', '116.7', '8', '14.6']
['31', 'Colin Kaepernick', 'FA', '110.3', '9', '12.3']
['32', 'Nick Foles', 'PHI', '95.9', '11', '8.7']
['33', 'Johnny Manziel', 'FA', '95.0', '9', '10.6']
['34', 'Peyton Manning', 'FA', '91.3', '10', '9.1']
['35', 'Matt Hasselbeck', 'FA', '91.2', '8', '11.4']
['36', 'Ryan Mallett', 'FA', '71.0', '7', '10.1']
['37', 'Matt Cassel', 'DET', '66.9', '8', '8.4']
['38', 'Brandon Weeden', 'HOU', '66.4', '6', '11.1']
['39', 'AJ McCarron', 'OAK', '55.3', '7', '7.9']
['40', 'Case Keenum', 'DEN', '43.6', '5', '8.7']
['41', 'Zach Mettenberger', 'FA', '42.2', '7', '6.0']
['42', 'Tony Romo', 'FA', '40.6', '4', '10.2']
['43', 'EJ Manuel', 'FA', '38.8', '4', '9.7']
['44', 'Jimmy Clausen', 'FA', '33.1', '4', '8.3']
['45', 'Kellen Moore', 'FA', '33.0', '3', '11.0']
['46', 'Mark Sanchez', 'FA', '32.8', '3', '10.9']
['47', 'Michael Vick', 'FA', '30.7', '5', '6.1']
['48', 'Matt Schaub', 'ATL', '26.6', '2', '13.3']
['49', 'T.J. Yates', 'FA', '22.9', '4', '5.7']
['50', 'Landry Jones', 'FA', '22.1', '7', '3.2']
['51', 'Geno Smith', 'LAC', '20.0', '1', '20.0']
```

IMPLEMENTATINO & SOURCE CODE SNIPPET

Task 1:

A sentence is taken from the user as an input string. A function FirstNonRepeat(string) was defined to find the first non-repeated character in the string. A for loop was used for iterating over the input to get each character in the string. The occurrence of each character in the string is calculated using .count() method, which is an inbuilt function in Python that returns the number of occurrences of a substring (each character in this task) in the given string. The occurrence of 1 indicates that the character appears only once in that string. At the first time with occurrence ==1, the character was printed out and the for loop was terminated by break statement. Thus, the first non-repeated character in the string was found.

```
# Q1. search in a string and find the first non-repeated characters in that string

# define a function to find the first non-repeated character
def FirstNonRepeat(string):
    for c in string:
 # count the occurences of the current character presents and find the 1st one it only shows once
        if string.count(c)==1:
            print('the first non-repeated character is ',c)
            break

sentence=input('enter a sentence ')
sentence=sentence.lower()              # ignore the difference due to capitalization of each word
sentence=sentence.replace(" ","")      # remove all space inside the sentence
FirstNonRepeat(sentence)
```

Task 2 :

To open the given files (testfile1.txt and textfile2.txt) in Python, a built-in function open() was used. The method readline() reads one entire line from the file and assigned it to the strings (line1 and line2), respectively. The .split(“ ”) method uses space (“ ”) to split the given string into a list item, which was assigned to listline1 and listline2.

A for loop was used for iterating over the listline1 to get every list item. A conditional statement was applied to check if the list item was also in the listline2 or not. To ignore the case-based difference, the .lower() method was applied to return a copy of the string in which all cased-based characters have been lowercased). Both original word and it’s lowercased form were used in the conditional statement. Every item (including its lowercased form) that is in listline1 but not in listline2 was added the list difstr.

The results were then printed out as a string in which the string elements have been joined by a string separator (sepa=” ”, space is used as the separator in this task).

```
# Q2. remove everything in the file 1 that is inside file 2

filename1 = 'testfile1.txt'             # file 1
filename2 = 'testfile2.txt'             # file 2
infile1=open(filename1)                 # open file 1
infile2=open(filename2)                 # open file 2
line1=infile1.readline()                # read line from file 1
line2=infile2.readline()                # read line from file 2
listline1=line1.split(" ")              # return a list of string from file 1 separated by space
listline2=line2.split(" ")              # return a list of string from file 2 separated by space
sepa=" "
difstr=[]

print("the file 1 is: ",line1)
print("the file 2 is: ",line2)
for i in listline1:
    if (i and i.lower()) not in listline2:      # ignore the capitalization and filter out the strings in
        difstr.append(i)
# join the elements of sequence using separator " " and print it out
print("the result after removing file 2 from file is: \n",sepa.join(difstr))
```

Task 3 :

Two given lists indicated the list of students who are attending “python” class (Plist) and the list of students who are attending “Web Application” class (WAlist). A for loop was used for iterating over every item in the Plist, then a nested for loop was used for iterating over every item in WAlist. Inside the inner for loop, a conditional statement was used to check if the item is in the Plist but not WAlist, and items satisfying the condition were added into the list inPnotWAlist. The result was printed out on the screen.

```
# Q3. Find the list of students who are attending “python” classes but not “Web Application”

# list of student in "python class"
Plist=['Anna','Adam','Ben','Chloe','Dan','Ella','Fin','Greg','Henry','Ian','Jake','Micheal','Monica','Nancy','Peter','Steve','Ted','Zach']
# list of student in "Web Application" class
WAlist=['Adam','Ben','Blake','Dan','Ella','Fin','Greg','Ian','Jake','Luke','Micheal','Monica','Paul','Peter','Steve','Zoe']
print("student in python class: \n",Plist)
print("student in Web Application class: \n",WAlist)
inPnotWAlist=[]
for i in Plist:
    if i not in WAlist:
        inPnotWAlist.append(i)
print('the list of students who are attending python not web application are: \n',inPnotWAlist)
```

Task 4 :

Five classes (Patient, Doctor, Nurse, Admission_Clerk, Book) were created in the code to represent the hospital admission system.

```python
class Patient:
    """A class to represent all Patient"""
    def __init__(self, pid, pn, pa, ps):
        self.patient_ID = pid
        self.patient_name = pn
        self.patient_age = pa
        self.patient_sex = ps
        self.__privateinfo = "not for public"   # this a super private data member with 2 leading under-scores in its name

    def get_patient_information(self):
        print("patient's ID, name, age, sex: ")
        print(self.patient_ID, self.patient_name, self.patient_age, self.patient_sex)
        return self.patient_ID, self.patient_name, self.patient_age, self.patient_sex

class Doctor:
    """A class to represent all Doctor"""
    def __init__(self, dn, da, ds, ddept):
        self.doctor_name = dn
        self.doctor_age = da
        self.doctor_sex = ds
        self.doctor_department = ddept

    def get_doctor_information(self):
        print("doctor's name, age, sex, and department: ")
        print(self.doctor_name, self.doctor_age, self.doctor_sex, self.doctor_department)
        return self.doctor_name, self.doctor_age, self.doctor_sex, self.doctor_department

class Nurse:
    """A class to represent Nurse"""
    def __init__(self, nn, na, ns, ndept):
        self.nurse_name = nn
        self.nurse_age = na
        self.nurse_sex = ns
        self.nurse_department = ndept

    def get_nurse_information(self):
        print("nurse's name, age, sex, and department: ")
        print(self.nurse_name, self.nurse_age, self.nurse_sex, self.nurse_department)
        return self.nurse_name, self.nurse_age, self.nurse_sex, self.nurse_department

class Admission_Clerk:
    """A class to represent Admission Clerk"""
    def __init__(self):
 # this is a semi private data with 1 leading under-score in its name
        self._semiprivate_admission = "this is semi private"
 # this a super private data with 2 leading under-scores in its name, which can't be seen and accessed from outside
        self.__superprivate_admission = "private, not for public"

class Book:
    """A class to represent doctor's avalaibility"""
    def __init__(self,bdn,bdept,bda):
        self.doctor_name = bdn
        self.department = bdept
```

Two inheritance classes (BookedPatient(Patient) and BookedDoctor(Doctor)) were created. One super call was applied to the BookedDoctor(Doctor) class. All defined classes had the __init__ constructor. The semi-private data and super private data were assigned in the class Adimission_Clerk() to test the accessibility from outside as shown in the Input/Output section.

```
class BookedPatient(Patient):
    """A class to represent patient booked appointment with Patient as parent class """
    def __init__(self, pid, pn, pa, ps, bnum):
        Patient.__init__(self, pid, pn, pa, ps)
        self.book_number = bnum

    def get_patient_book_number(self):
        self.book_number += 1
        print("the book number for patient is: ", self.book_number)
        return self.book_number

class BookedDoctor(Doctor):
    """A class to represent doctor booked appointment with Doctor as parent class """
    def __init__(self, dn, da, ds, ddept,bdn):
        super().__init__(dn, da, ds, ddept)
        self.booked_doctor_number = bdn

    def get_doctor_book_number(self):
        print("the number for booked doctor is: ", self.booked_doctor_number)
        return self.booked_doctor_number
```

Task 6 :

The urlopen() function was imported from urllib module, which can accept URLs (Universal Resource Locators) and open URLs for reading (html). The SSL (Secure Sockets Layers) module was imported to create a secure connection between server and client. A customer defined ssl_context was created in the context parameter in case urllib raises an exception when it can't verify the server certificate. To extract table from the given website, the BeautifulSoup library was imported to parse the webpage. An object (soup) was created by passing through html to the BeautifulSoup constructor. A list (result_list) with the tag name "div" and "class: mobile-table" was returned using the .findAll() method. For each element in the result_list, the objects with tag name "table" was searched and returned. The table header was extracted using tag name "thead" and .getText() method. Similarly, the rows of the table were found using tag name "tr" and .findAll() method. Every row was searched using the for loop through all rows of the table. For each row, every column was extracted using the for loop through the row. The extracted table contents were added to the list (data). To write the table content into a file, a new file named "table_content.txt" was created. Both the header and the table content were written into the file and printed out.

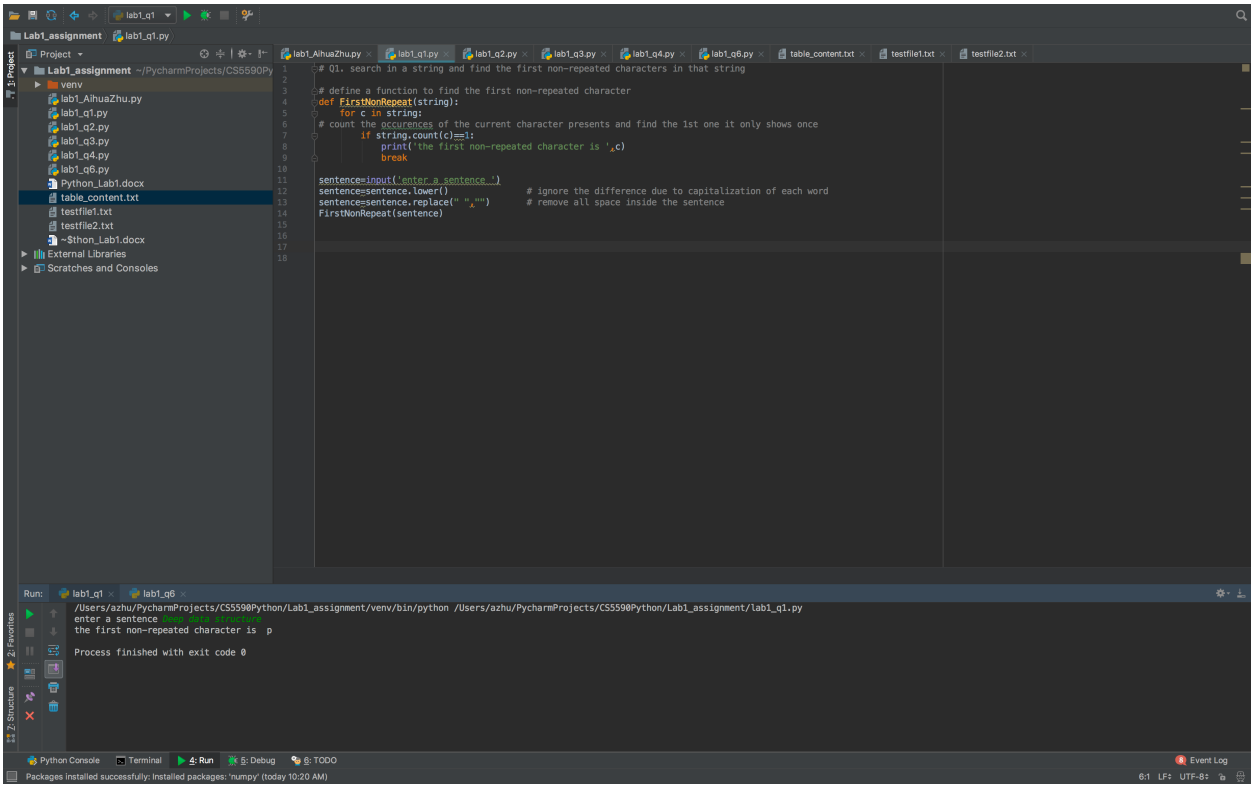
```
# You should save all the information of the table in a file. Sample input: https://www.fantasypros.com/nfl/reports/leaders/qb.php?year=2015
# Sample output: Save the table in this link into a file
"""
from bs4 import BeautifulSoup
from urllib.request import urlopen
import ssl

url = "https://www.fantasypros.com/nfl/reports/leaders/qb.php?year=2015"    # url that is scraped
# create a custom ssl context in the context parameter in case urllib will raise an exception when it can't verify the server certificate
context = ssl._create_unverified_context()
html = urlopen(url,context=context)     # the html from given url
# create an BeautifulSoup object by passing through html to the BeautifulSoup constructor
soup = BeautifulSoup(html, "html.parser")
# returns a list of BeautifulSoup Tag div objects that has the class: mobile-table
result_list = soup.findAll('div', {'class': "mobile-table"})
file = open('table_content.txt','w')       # open the file to write output
data=[]
header=[]

for div in result_list:
    result_table = div.find('table')
# to extract the column header from the table
    result_header = div.find('thead')
    table_header = result_header.findAll('th')
    for th in table_header:
        column_header = th.getText()
        header.append(column_header)
# to extract the cell element from the table for each row
    table_body = result_table.find('tbody')
    rows = table_body.find_all('tr')
    for row in rows:
        columns = row.findAll('td')
        item = [col.getText() for col in columns]
        data.append(item)

file.write(str(header))         # write the table header into the file
file.write("\n")
print(str(header))
for i in data:
    file.write(str(i))          # write the table content into the file'
    file.write("\n")
    print(str(i))

file.close()
```

DEPLOYMENT

All the code for this lab assignment was deployed on PYCHARM software on macOS. The program files were saved in .py format and other Input/Output files were in text (.txt) format. The open source package conda package was added to the PYCHARM project interpreter. The console window was used to enter input and demonstrate the output from the code.

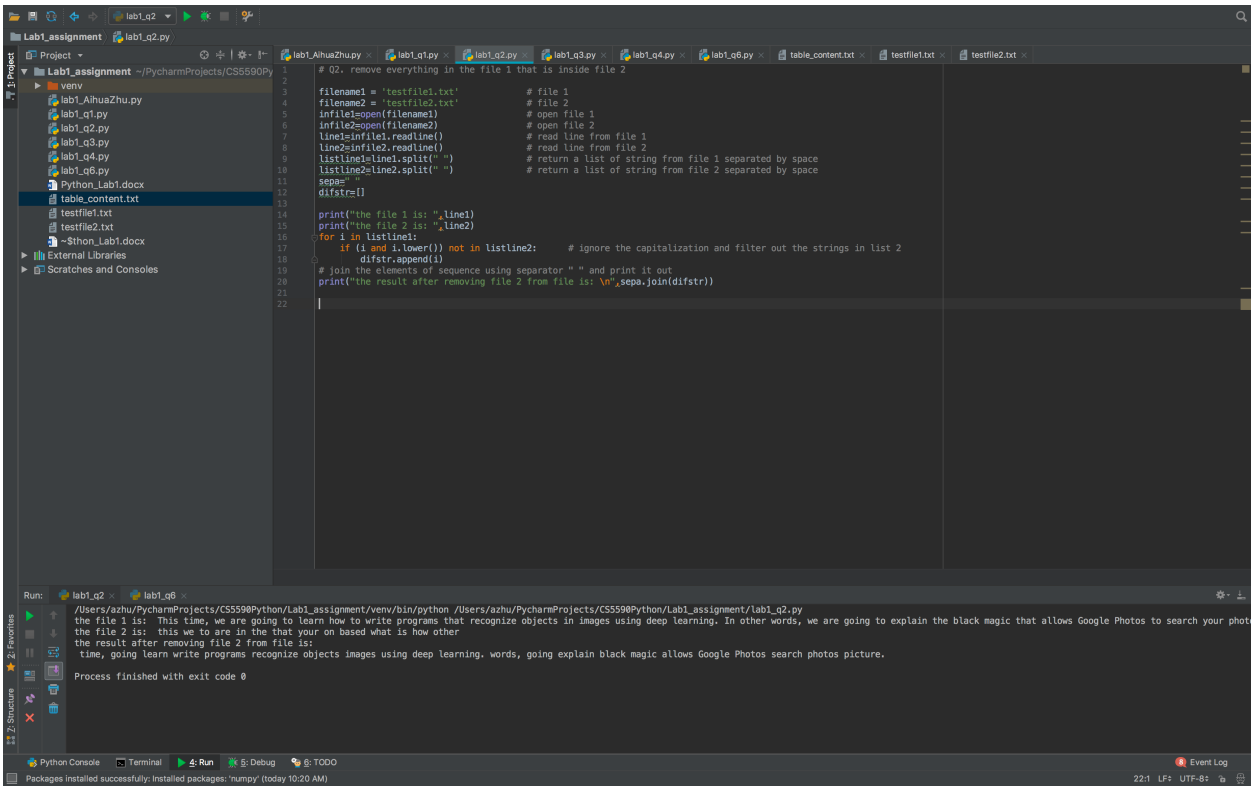
Task 1:

Entering a string “Deep data structure” as the input and results were printed out in the console window as shown in the following.



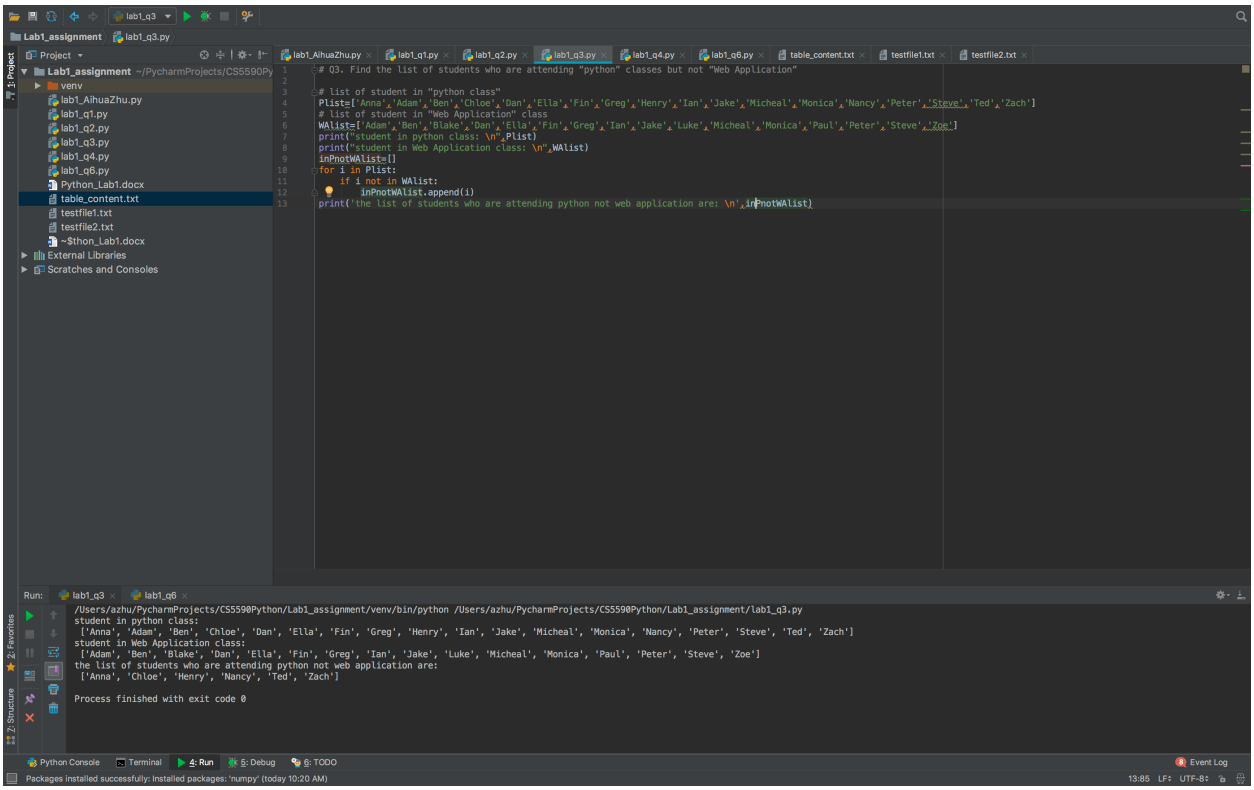
Task 2:

Two given files were open as the input and results were printed out in the console window as shown in the following.



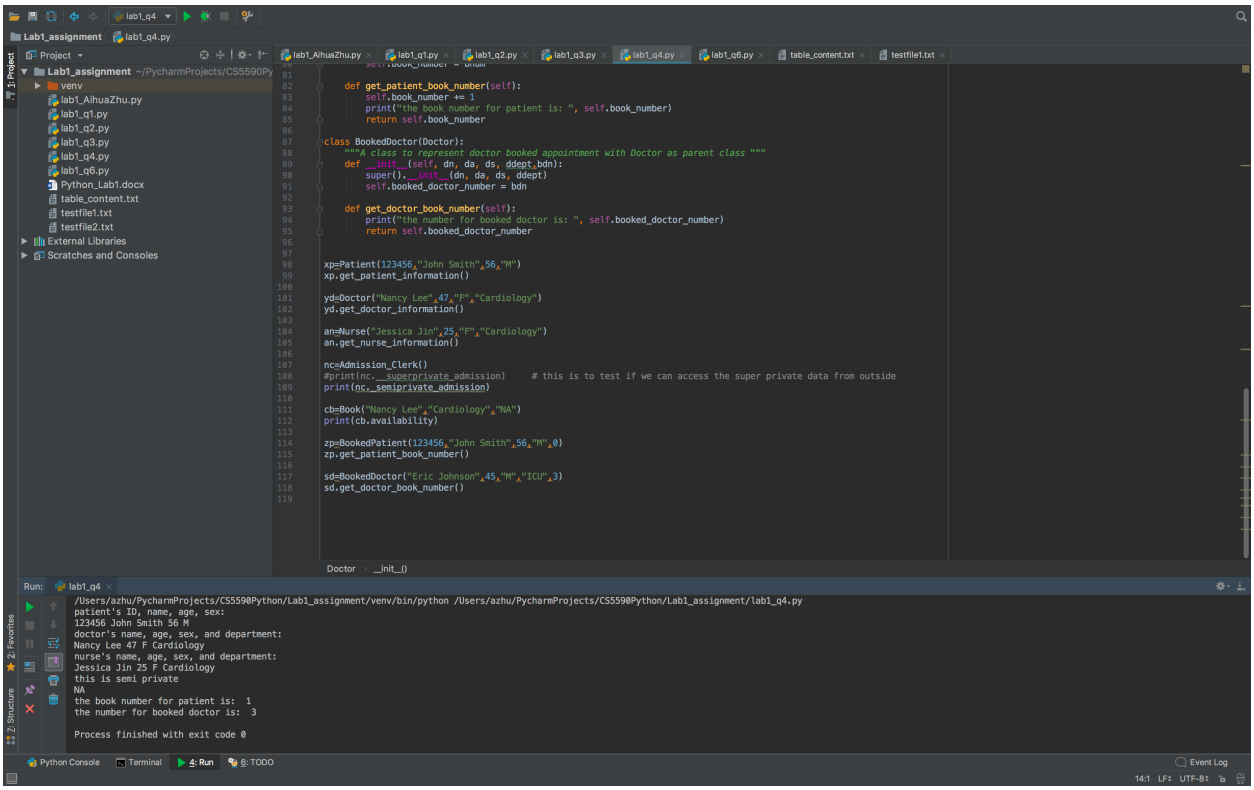
Task 3:

Two lists were given in the code as the input and results were printed out in the console window as shown in the following.



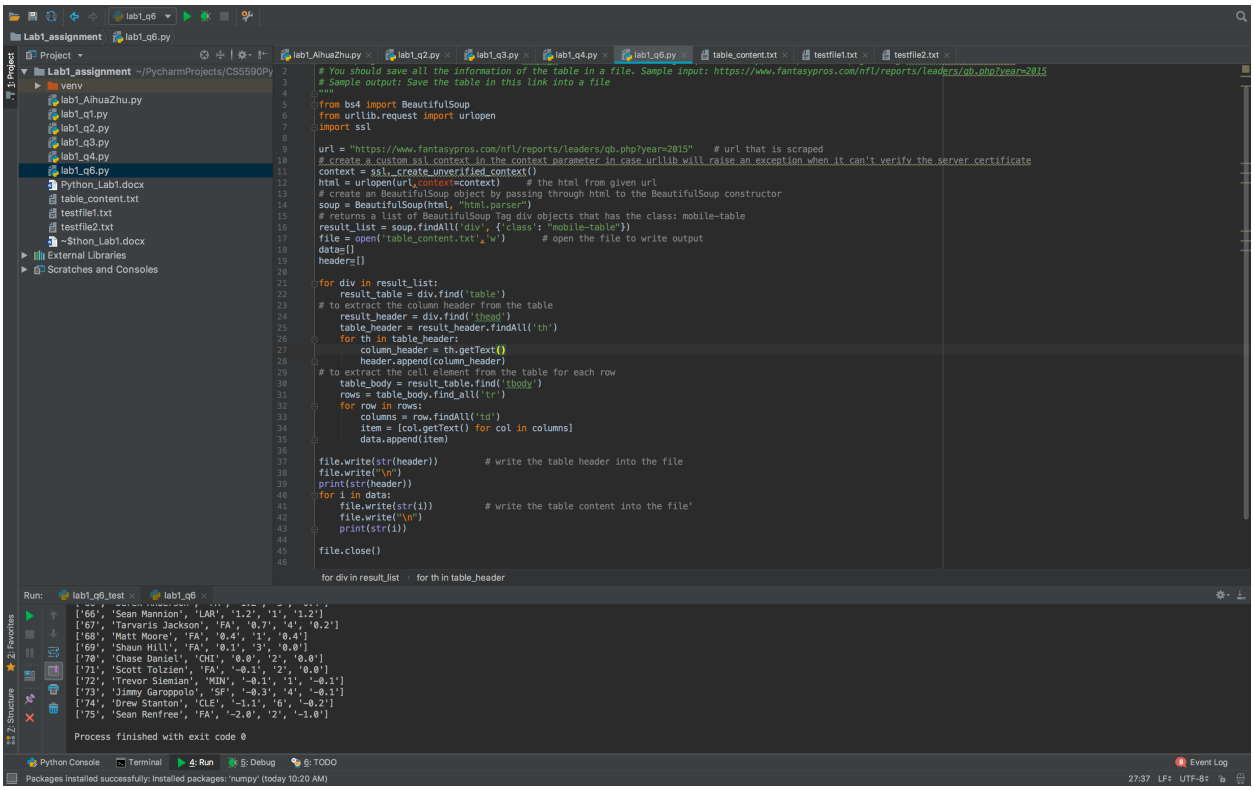
Task 4:

Different instances were given in the code to demonstrate each class and results were printed out in the console window as shown in the following.



Task 6:

The url link was given in the code as the input and the results were saved into a text file and also printed out in the console window.



LIMITATIONS

All python codes accomplished the objective of the task and met all the requirements given. A few limitations of the codes are listed below and can be improved in the future projects with further learning of python programming.

- In task 3, two input lists were given inside the code, it can be improved to let user enter the name list or read a name file.
- In task 4, the relationship among different classes were not well presented

REFERENCES

https://www.pythonforbeginners.com/dictionary/python-split

http://savvastjortjoglou.com/nba-draft-part01-scraping.html

https://docs.python.org/2/library/urllib.html

https://www.dataquest.io/blog/web-scraping-tutorial-python/

https://www.crummy.com/software/BeautifulSoup/bs3/documentation.html#The%20basic%20find%20method:%20findAll(name,%20attrs,%20recursive,%20text,%20limit,%20**kwargs)