**CS5590 APS - Python Programming**

**Lab 1.**

Fall 2018

UMKC

Kim-Ndor Djimadoumngar

1. **Introduction**

This is a report by Kim-Ndor Djimadoumngar for Assignment 1 of the Special Topic in Applied Programming Learning (APL) series: CS5590-0001 python and Deep Learning. My student ID is 6; my Lab ID is 7. The course is taught by Dr. Yugyung Lee and instructed by Saria Goudarzvand.

1. **Objectives**

The aim of this lab is to familiarize students with string operations in lists, tuples, sets, dictionaries, and web scrabing as well as class creations and object oriented concepts. There are a total of five exercises on which specific operations are performed

1. **Approaches/Methods**

Python files were created in Pycharm Community Edition 2018 to implement the exercises. We also created txt files in notepad for required tasks such string creations.

1. **Results/Discussion: Input/output**

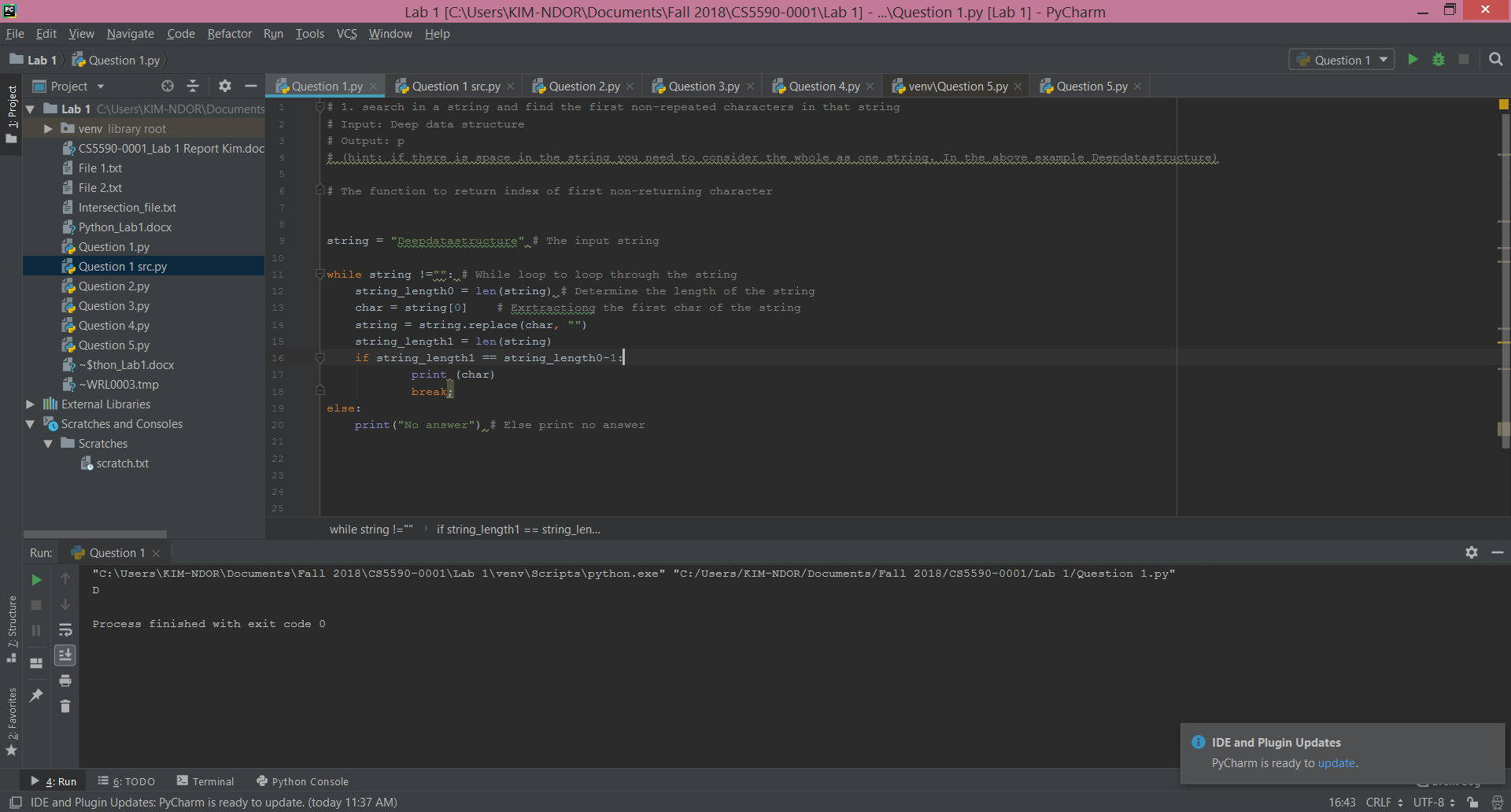
**Question 1**

I used while loop to loop through the string and compute its length. The length () was used to find the length of the string, then the char position was located. Each character is compared to the next; if the character is the same as the previous character, print “no answer”; if the character is different, print the character. The codes as well as the comments are shown below:

# 1. search in a string and find the first non-repeated characters in that string  
# Input: Deep data structure  
# Output: p  
# (hint: if there is space in the string you need to consider the whole as one string. In the above example Deepdatastructure)  
  
# The function to return index of first non-returning character  
  
  
string = "Deepdatastructure" # The input string  
  
while string !="": # While loop to loop through the string  
 string\_length0 = len(string) # Determine the length of the string  
 char = string[0] # Exrtracting the first char of the string  
 string = string.replace(char, "")  
 string\_length1 = len(string)  
 if string\_length1 == string\_length0-1:  
 print (char)  
 break;  
else:  
 print("No answer") # Else print no answer

**Result**

The output the above code gives the below screenshot:



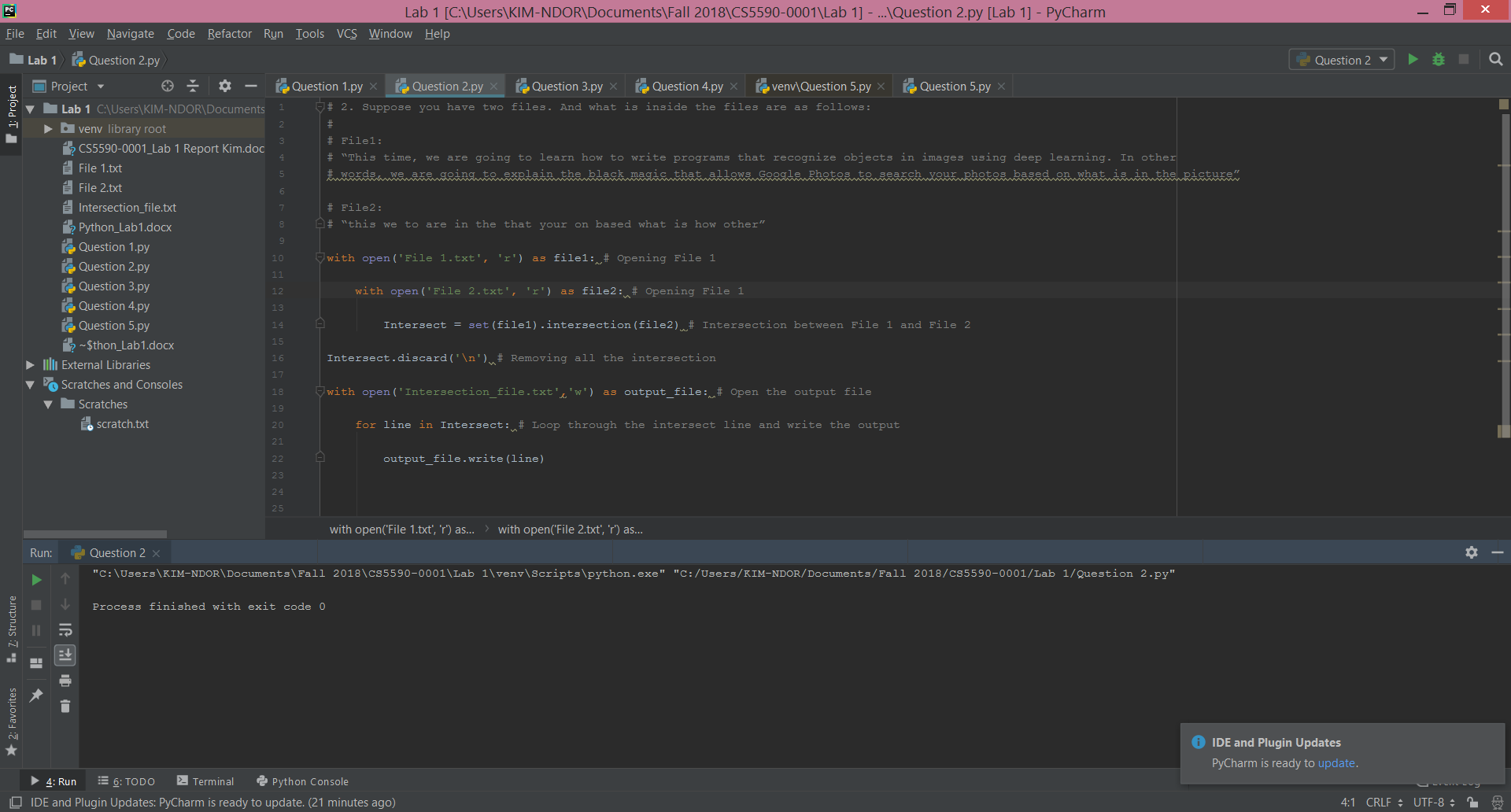
**Question 2**

We used with open() to open the given files File 1 and File 2. We intersected the two files using .intersect function. We remove all the elements that are in the intersection and output those that are only in File 2. We then save the output file in intersection\_fil.txt.

# 2. Suppose you have two files. And what is inside the files are as follows:  
#  
# File1:  
# “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”  
  
# File2:  
# “this we to are in the that your on based what is how other”  
  
with open('File 1.txt', 'r') as file1: # Opening File 1  
  
 with open('File 2.txt', 'r') as file2: # Opening File 1  
  
 Intersect = set(file1).intersection(file2) # Intersection between File 1 and File 2  
  
Intersect.discard('\n') # Removing all the intersection  
  
with open('Intersection\_file.txt','w') as output\_file: # Open the output file  
  
 for line in Intersect: # Loop through the intersect line and write the output  
  
 output\_file.write(line)

**Result**

The screenshot below show the output result of the above codes.



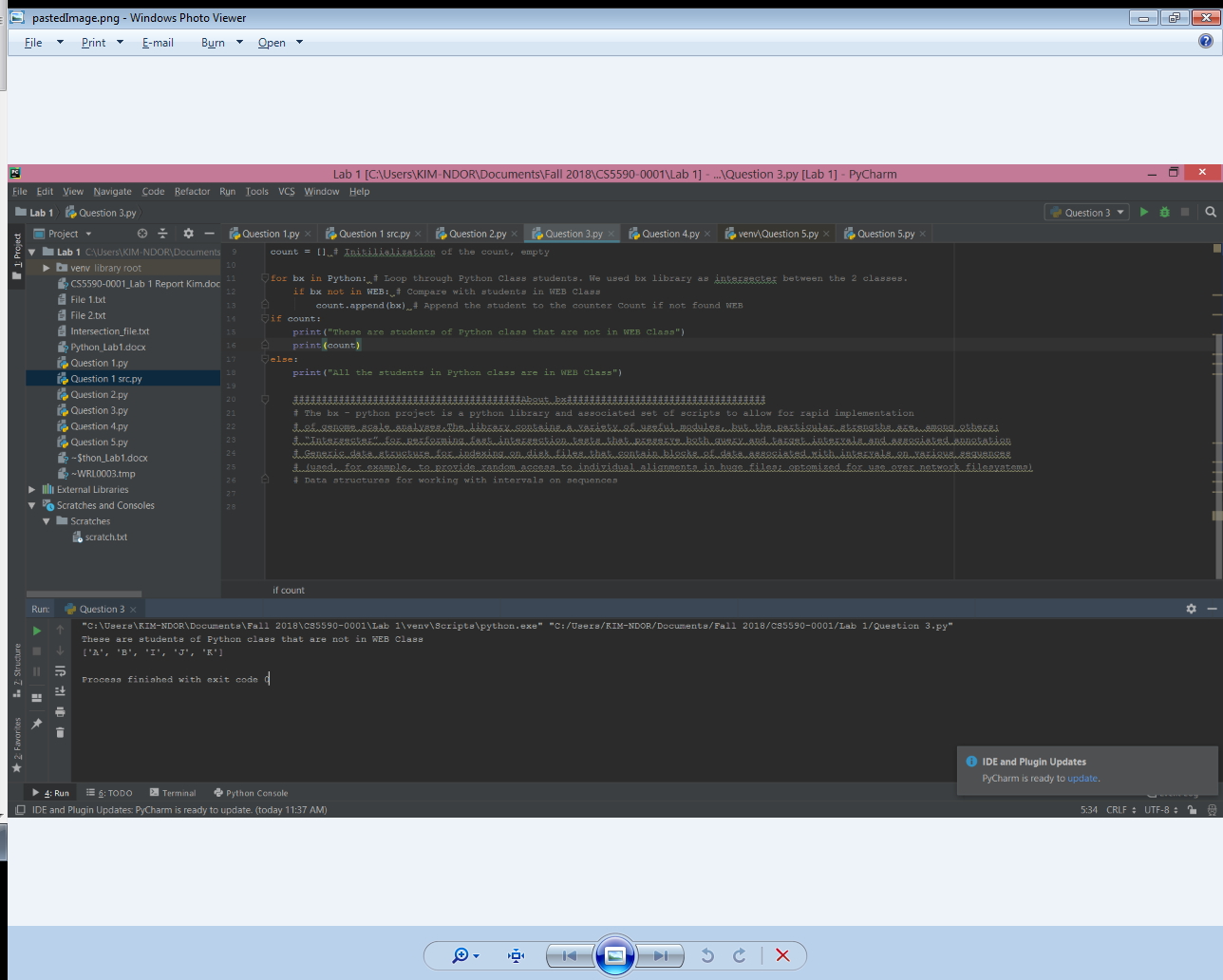
**Question 3**

I created the lists of students in the two classes. Python = [A", "B","C","D","E","F","G","H","I","J","K] and WEB = ["C","D","E","F","G","H"]. we created a counter Count to store the students that are not in WEB class. We loop through Python class using the bx-python library. The bx-python permits, among other tasks, data indexing on files and sequences as well as intersecting files. If the string is not in the WEB class, append it in the counter. In the counter, the print () function is used to display the list of students that are not in the WEB class. Otherwise, print all the students in Python class are also in WEB class.

# 3. 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 “python” classes but not “Web Application”  
  
Python =["A", "B","C","D","E","F","G","H","I","J","K"] # List of students in Python class  
  
WEB =["C","D","E","F","G","H"] # List of students in WEB class  
  
count = [] # Initialization of the count, empty  
  
for bx in Python: # Loop through Python Class students. We used bx library as intersecter between the 2 classes.  
 if bx not in WEB: # Compare with students in WEB Class  
 count.append(bx) # Append the student to the counter Count if not found WEB  
if count:  
 print("These are students of Python class that are not in WEB Class")  
 print(count)  
else:  
 print("All the students in Python class are in WEB Class")

**Result**

Running the above codes provides the students “A”, “B”, “I”, “J”, and “K”, as shown the below screenshot.



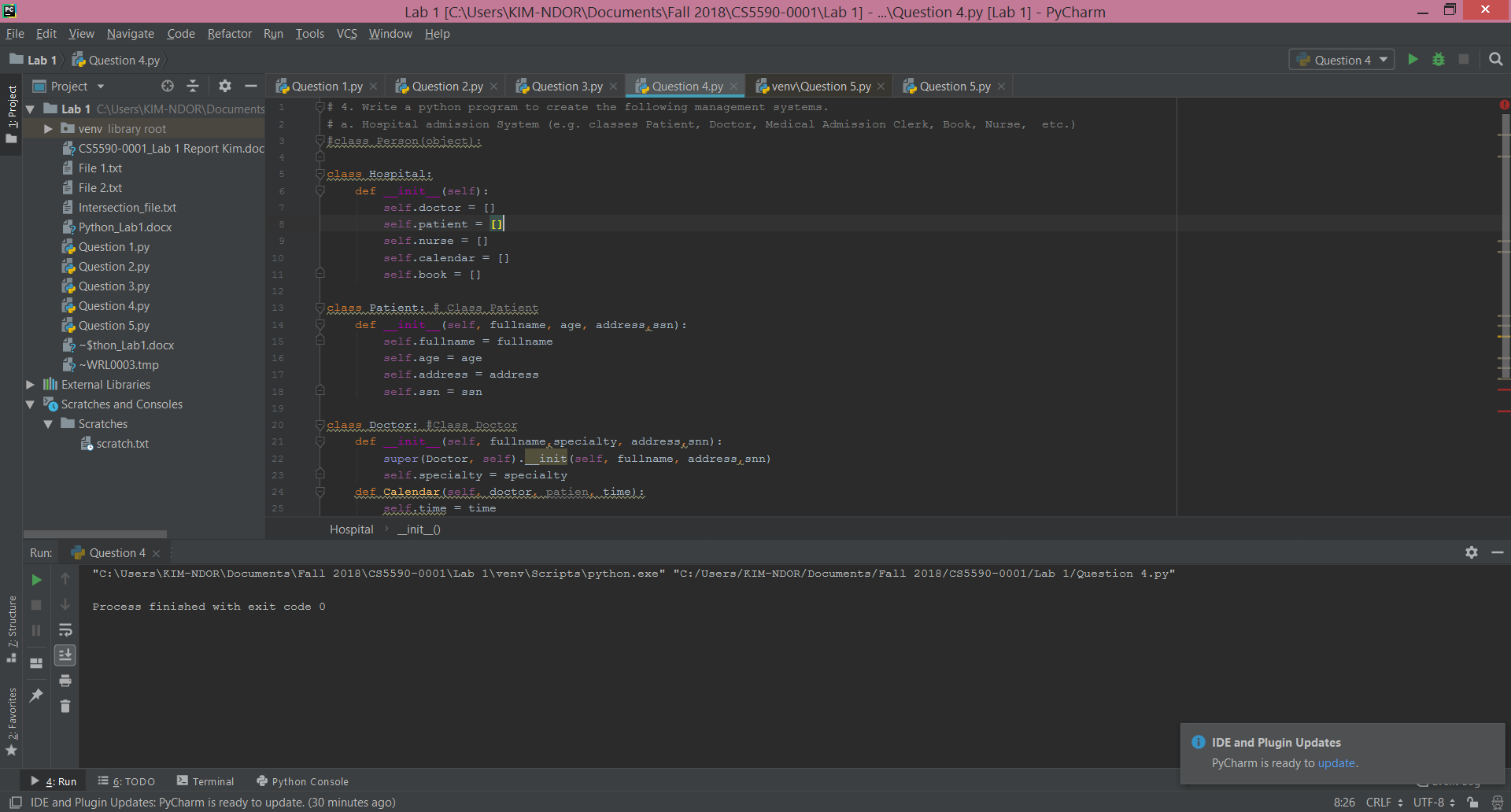
**Question 4**

We used class() function create the respective classes. We also used \_\_init\_\_ constructors in the classes to define the class attributes. We creates a superclass Doctor with Calendar as a subclass. Calendar inherits attributes from Doctor and Patient.

# 4. Write a python program to create the following management systems.  
# a. Hospital admission System (e.g. classes Patient, Doctor, Medical Admission Clerk, Book, Nurse, etc.)  
#class Person(object):  
  
class Hospital:  
 def \_\_init\_\_(self):  
 self.doctor = []  
 self.patient = []  
 self.nurse = []  
 self.calendar = []  
 self.book = []  
  
class Patient: # Class Patient  
 def \_\_init\_\_(self, fullname, age, address,ssn):  
 self.fullname = fullname  
 self.age = age  
 self.address = address  
 self.ssn = ssn  
  
class Doctor: #Class Doctor  
 def \_\_init\_\_(self, fullname,specialty, address,snn):  
 super(Doctor, self).\_\_init(self, fullname, address,snn)  
 self.specialty = specialty  
 def Calendar(self, doctor, patient, time):  
 self.time = time  
 self.doctor = doctor  
 self.patient = patient  
 if self.doctor.is\_available(time) and self.patient.is\_available(time):  
 self.patient.update\_calendar(patient,time)  
 self.doctor.update\_calendar(doctor, time)  
 print("Schedule")  
 return True

**Result**

The output screenshot is as follows:



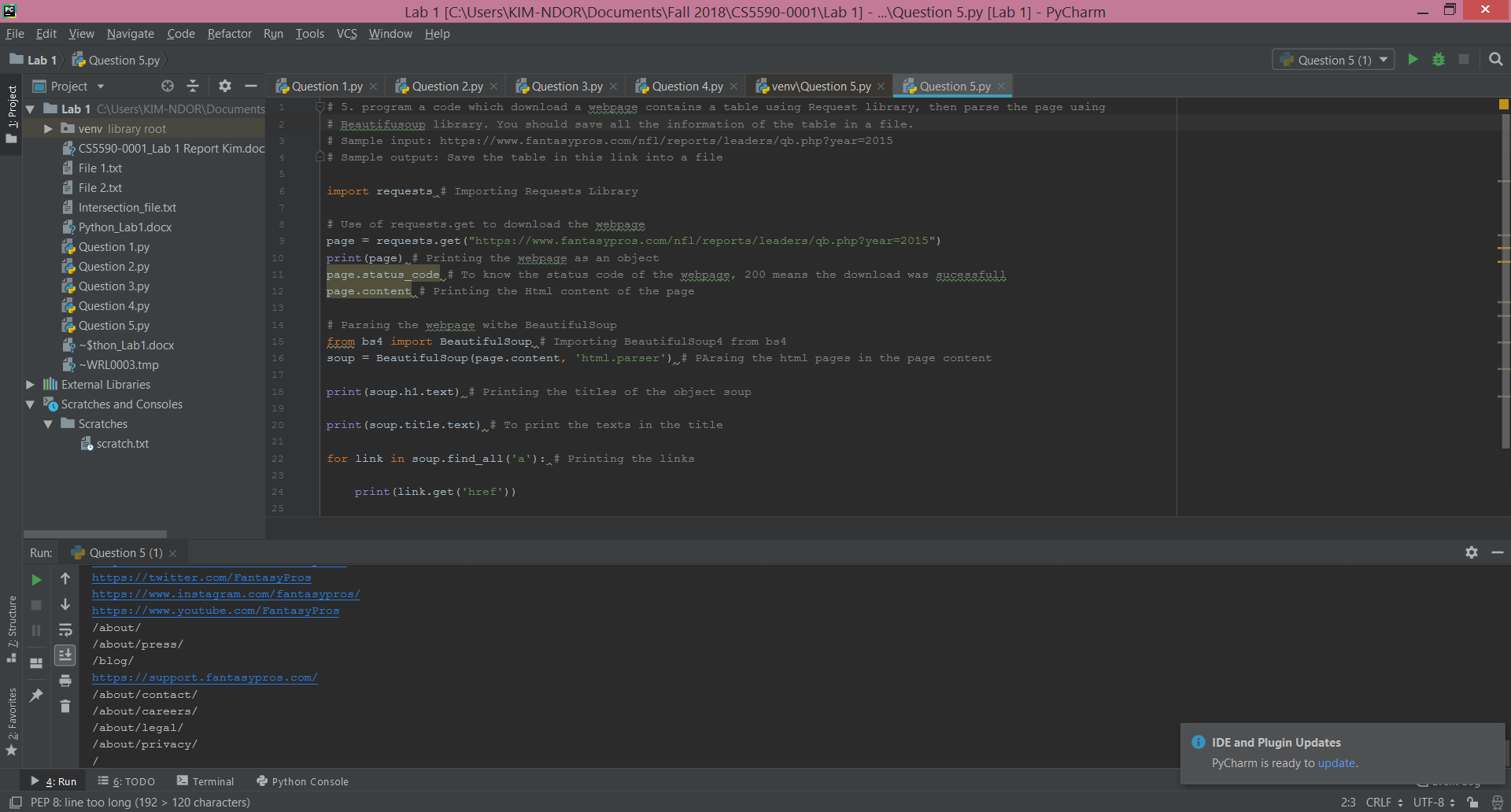
**Question 5**

Import request library was used for the importation. We used request.get() to get the webpage as assigned it to an object called page. The object page is printed using print() function. The code and the content of the webpage are provided using page.status.code and page.content function. BeautifulSoup is imported from bs4 to parse the html pages contained in the webpage. The paragraphs and titles in the content are printed using print() function. We loop through the page to find the links and print them.

# 5. program a code which download a webpage contains a table using Request library, then parse the page using  
# Beautifusoup library. 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  
  
import requests # Importing Requests Library  
  
# Use of requests.get to download the webpage  
page = requests.get("https://www.fantasypros.com/nfl/reports/leaders/qb.php?year=2015")  
print(page) # Printing the webpage as an object  
page.status\_code # To know the status code of the webpage, 200 means the download was sucessfull  
page.content # Printing the Html content of the page  
  
# Parsing the webpage withe BeautifulSoup  
from bs4 import BeautifulSoup # Importing BeautifulSoup4 from bs4  
soup = BeautifulSoup(page.content, 'html.parser') # PArsing the html pages in the page content  
  
print(soup.h1.text) # Printing the titles of the object soup  
  
print(soup.title.text) # To print the texts in the title  
  
for link in soup.find\_all('a'): # Printing the links  
  
 print(link.get('href'))

**Result**

The printed links are shown in the screenshot.



**References**

About bx-python in <https://bx-python.readthedocs.io/en/latest/> (accessed on Sept. 17, 2018).

Multiple Inheritance in <https://www.python-course.eu/python3_multiple_inheritance.php> (accessed on Sept. 18, 2018).

Introduction to BeautifulSoup in <https://sixfeetup.com/blog/an-introduction-to-beautifulsoup> (accessed on Sept. 19, 2018).