

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

Python Programming

Lab Assignment 2

Assignment Overview

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

One should be able to use sets and dictionaries in any environment.

One should be able to use classes to build any of the management system. This can be used as back end for any web application created in Python. Also we will learn about numpy package and will see how this can be implemented.

Lab Assignment

1. Consider a shop UMKC with dictionary of all book items with their prices. Write a program to find the books from the dictionary in the range given by user.

Sample Input:

```
{“python”:50,“web”:30,“c”:20,“java”:40}
```

For range 30 to 40

Sample Output:

You can purchase books (web, java)

Results:

The following code was developed to create a phone contact list that can be searched and edited by the user.

```
#Don Baker
#Lab 2, Problem 1

#Consider a shop UMKC with dictionary of all book items with their prices.
# Write a program to find the books from the dictionary in the range given by user.

#Create the dictionary containg the book subjects and the associated prices
books = {'Python is a Piece of Pie': 50, 'The Web is not for Spiders': 30, 'C or
Sea?': 20, 'Java not Lava': 40, 'Pascal is My Pal': 10, 'Fortran Forever': 5}
print("The dictionary containing the information for books is:\n")
print(books)

#User input for price range
lr,hr = input("\nPlease the minimum and maximum price of your range you wish to pay.
Please separate with a comma: ").split(',')
```

```
low_range = int(lr)
high_range = int(hr)

print("\nThe range of prices selected is $",low_range,"to $",high_range,".")

#Determine the books that fall within the selected range.

#Print the books that fall within the price range specified.
print("\nThe books that fall within the specified price range are :\n")

for key, value in books.items():
    if value >= low_range:
        if value <=high_range:
            print(key)
```

The results of this code are shown below.

```
"D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 2\Lab2\venv\Scripts\python.exe"
"D:/Google Drive/UMKC/PhD/Classes/Python/Labs/Lab 2/Lab2/Lab2 Problem 1.py"
```

The dictionary containing the information for books is:

```
{'Python is a Piece of Pie': 50, 'The Web is not for Spiders': 30, 'C or Sea?': 20,
'Java not Lava': 40, 'Pascal is My Pal': 10, 'Fortran Forever': 5}
```

Please the minimum and maximum price of your range you wish to pay. Please separate with a comma: 23,42

The range of prices selected is \$ 23 to \$ 42 .

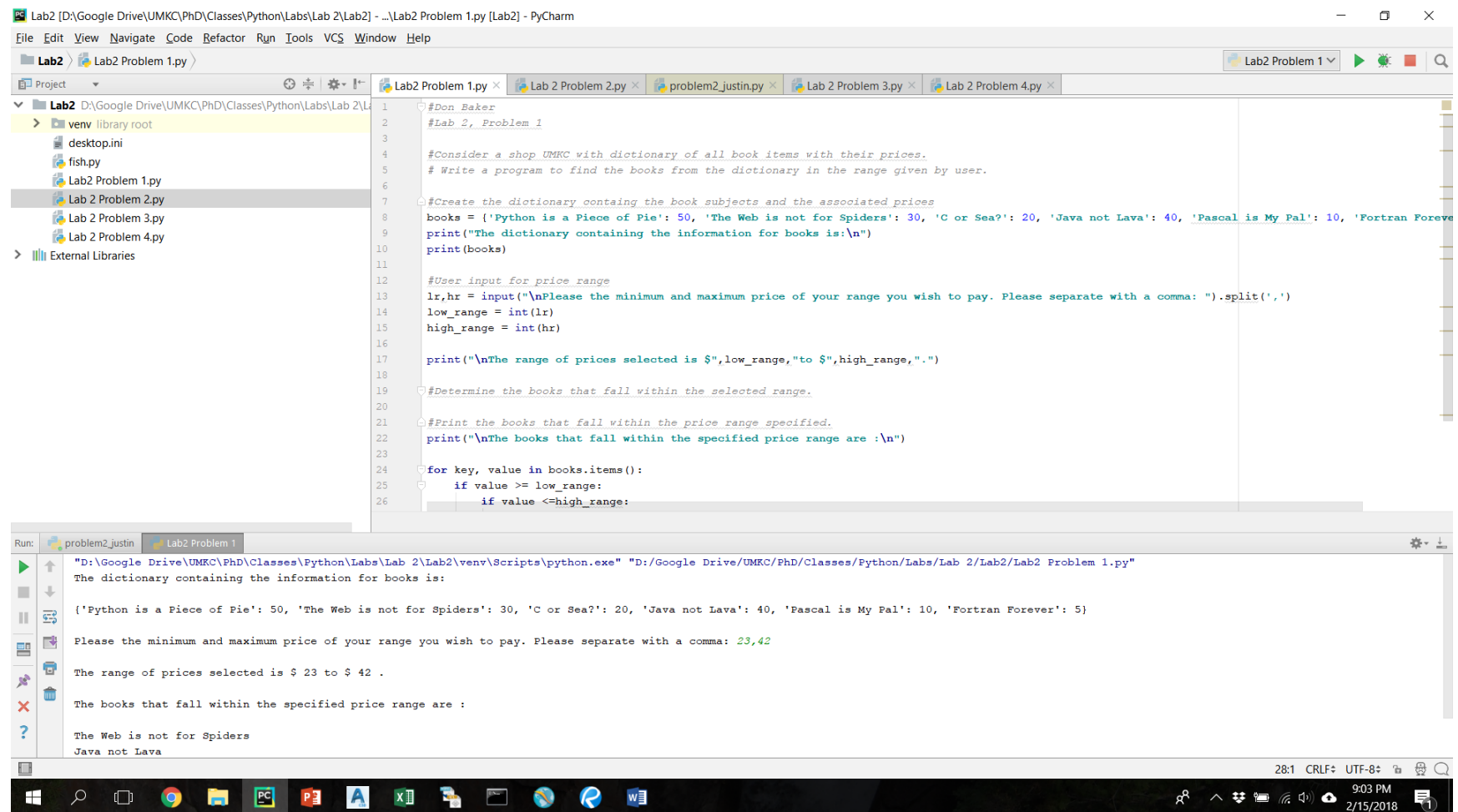
The books that fall within the specified price range are :

The Web is not for Spiders

Java not Lava

Process finished with exit code 0

Figure 1 illustrates the screenshot of the program for this problem.



```
#Don Baker
#Lab 2, Problem 1

#Consider a shop UMKC with dictionary of all book items with their prices.
# Write a program to find the books from the dictionary in the range given by user.

#Create the dictionary containing the book subjects and the associated prices
books = {'Python is a Piece of Pie': 50, 'The Web is not for Spiders': 30, 'C or Sea?': 20, 'Java not Lava': 40, 'Pascal is My Pal': 10, 'Fortran Forever': 5}
print("The dictionary containing the information for books is:\n")
print(books)

#User input for price range
lr,hr = input("\nPlease the minimum and maximum price of your range you wish to pay. Please separate with a comma: ").split(',')
low_range = int(lr)
high_range = int(hr)

print("\nThe range of prices selected is $",low_range,"to $",high_range, ".")

#Determine the books that fall within the selected range.

#Print the books that fall within the price range specified.
print("\nThe books that fall within the specified price range are :\n")

for key, value in books.items():
    if value >= low_range:
        if value <= high_range:
```

Run: problem2_justin Lab2 Problem 1

"D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 2\Lab2\venv\Scripts\python.exe" "D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 2\Lab2\Lab2 Problem 1.py"

The dictionary containing the information for books is:

```
{'Python is a Piece of Pie': 50, 'The Web is not for Spiders': 30, 'C or Sea?': 20, 'Java not Lava': 40, 'Pascal is My Pal': 10, 'Fortran Forever': 5}
```

Please the minimum and maximum price of your range you wish to pay. Please separate with a comma: 23,42

The range of prices selected is \$ 23 to \$ 42 .

The books that fall within the specified price range are :

```
The Web is not for Spiders
Java not Lava
```

Figure 1 Screenshot Illustrating Books Available in the Selected Range

2. With any given number n, in any mobile, there is contact list. Create a list of contacts and then prompt the user to do the

following:

- a) Display contact by name
- b) Display contact by number
- c) Edit contact by name
- d) Exit

Based on the above scenario, write a single program to perform the above the operations. Each time an operation is performed on the list, the contact list should be displayed

Sample input:

```
Contact_list=[{"name":"Rashmi","number":"8797989821","email":"rr@gmail.com"},{  
"name":"Saria","number":"9897989821","email":"ss@gmail.com"}]
```

Suppose user select to edit contact "Rashmi"

Edit the number to 9999999999 as given by user

Sample output:

```
Contact_list=[{"name":"Rashmi","number":"9999999999","email":"rr@gmail.com"},{  
"name":"Saria","number":"9897989821","email":"ss@gmail.com"}]
```

Results:

The following code was developed to create a phone contact list that can be searched and edited by the user.

```
#Don Baker  
#Lab 2, Problem 2  
  
#With any given mobile phone number n, there is contact list.  
#Create a list of contacts and then prompt the user to do the following:  
#a) Display contact by name  
#b) Display contact by number  
#c) Edit contact by name  
#d) Exit  
  
#Define the Print Instructions routine for the user  
def PrintInstructions():
```

```

    print("Please choose one of the following choices: ")
    print("1 - Display contact by name")
    print("2 - Display contact by number")
    print("3 - Edit phone number by contact name")
    print("4 - Exit\n")

#Main program
def main():

    #Create contact list
    contact_list = [{"name": "Rashmi", "number": "8797989821", "email":
"rr@gmail.com"},
                    {"name": "Saria", "number": "9897989821", "email":
"ss@gmail.com"}]

    #print the instructions for the user
    PrintInstructions()

    #Input the users choice
    choice = int(input("Enter choice: "))

    while choice != 4:

#Retrieve contact by name
        if choice == 1:
            name = input("Please enter contact name to be retrieved: ")
            print()
            for dic in contact_list:
                if ("name", name) in dic.items():
                    for k, v in dic.items():
                        print(k, ":", v)

#Retrieve contact by number
            if choice == 2:
                number = input("Please enter contact number to be retrieved: ")
                print()
                for dic in contact_list:
                    if ("number", number) in dic.items():
                        for k, v in dic.items():
                            print(k, ":", v)

#Change phone number by entering name of contact
            if choice == 3:
                name = input("Enter contact name to change: ")
                newNumber = input("Enter new number: ")
                print()

                for dic in contact_list:
                    if ("name", name) in dic.items():
                        dic["number"] = newNumber
                        for k, v in dic.items():
                            print(k, ":", v)

            choice = int(input("\nEnter choice: "))

if __name__ == "__main__":
    main()

```

The results of this code are shown below. Each of the options is chosen.

```
"D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 2\Lab2\venv\Scripts\python.exe"
```

```
"D:/Google Drive/UMKC/PhD/Classes/Python/Labs/Lab 2/Lab2/Lab 2 Problem 2.py"
```

Please choose one of the following choices:

- 1 - Display contact by name
- 2 - Display contact by number
- 3 - Edit phone number by contact name
- 4 - Exit

Enter choice: 1

Please enter contact name to be retrieved: Saria

name : Saria

number : 9897989821

email : ss@gmail.com

Enter choice: 2

Please enter contact number to be retrieved: 8797989821

name : Rashmi

number : 8797989821

email : rr@gmail.com

Enter choice: 3

Enter contact name to change: Rashmi

Enter new number: 9999999999

name : Rashmi

number : 9999999999

email : rr@gmail.com

Enter choice: 4

Process finished with exit code 0

Figure 2 illustrates the screenshot of the program for this problem.

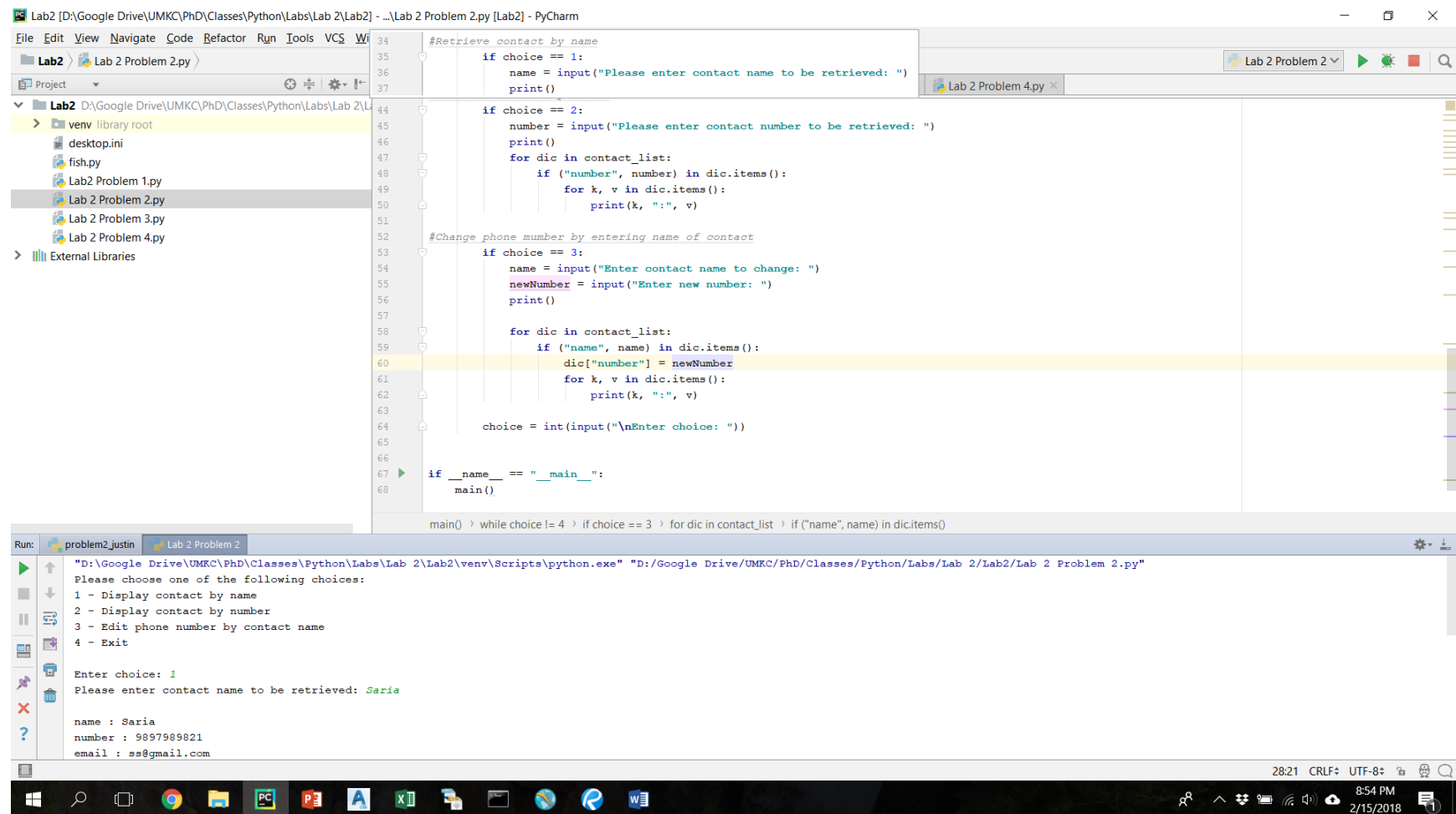


Figure 2 Screenshot Illustrating the Phone List Program

3. Write a python program to create any one of the following management systems. You can also pick one of your own.

- a. Library Management System (should have classes for Person, Student, Librarian, Book etc.)
- b. Airline Booking Reservation System (classes for Flight, Person, Employee, Passenger etc.)
- c. Hotel Reservation System (classes for Room, Occupants, Employee etc.)
- d. Student Enrollment System (classes for Student, System, Grades etc.)
- e. Expense Tracker System (classes for Expense, Transaction Category etc.)

Prerequisites:

Your code should have at least five classes.

Your code should have `_init_` constructor in all the classes

Your code should show inheritance atleast once

Your code should have one super call

Use of self is required

Use at least one private data member in your code.

Use multiple Inheritance atleast once

Create instances of all classes and show the relationship between them.

Your submission code should point out where all these things are present.

Results:

The following code was developed to create a levee inspection system.

```
#Don Baker
#Lab 2, Problem 3

#Write a python program to create any one of the following management systems.
#You can also pick one of your own.
#a. Library Management System (should have classes for Person, Student, Librarian,
Book etc.)
#b. Airline Booking Reservation System (classes for Flight, Person, Employee, Passenger
etc.)
#c. Hotel Reservation System (classes for Room, Occupants, Employee etc.)
#d. Student Enrollment System (classes for Student, System, Grades etc.)
#e. Expense Tracker System (classes for Expense, Transaction Category etc.)

#Prerequisites:
#Your code should have at least five classes.
#Your code should have _init_ constructor in all the classes
#Your code should show inheritance atleast once
#Your code should have one super call
#Use of self is required
#Use at least one private data member in your code.
#Use multiple Inheritance atleast once
#Create instances of all classes and show the relationship between them.
#Your submission code should point out where all these things are present.

#Create the Python code for a levee inspection system

#Create class for Levee
class Levee:
```



```
#Create constructor for Levee class
def __init__(self,name,location):
    self.levee_name = name
    self.levee_location=location

#Create class for levee systems
class Levee_System:

    #Create constructor for Levee class
    def __init__(self, syst_name,syst_location):
        self.system_name = syst_name
        self.system_location=syst_location

#Create class for levee owners
class Owner(Levee):
    pass

    #Create constructor for Levee owner
    def __init__(self,owner,name,location):
        self.system_owner=owner
        self.levee_name = name
        self.levee_location = location

#Create class for levee construction era
class Construction_Era(Levee_System):
    pass

    # Create constructor for construction era
    def __init__(self,era,syst_name,syst_location):
        self.const_era=era
        self.system_name = syst_name
        self.system_location = syst_location

#Create class for levee condition
class Condition(Levee):
    pass

    # Create constructor for levee condition
    def __init__(self, condition,name,location):
        super().__init__(name,location) # Super call Levee
        self.levee_condition = condition
        self.levee_name = name
        self.levee_location = location

#Create class for levee repair cost range
class Repair_Cost(Levee, Levee_System): #Multiple inheritance
    pass

    #Create constructor for repair cost of levee
    def __init__(self, cost,name,location,syst_name,syst_location):
        self.levee_cost=cost
        self.levee_name = name
        self.levee_location = location
        self.system_name = syst_name
        self.system_location = syst_location

#Define the levees for class Levee
Levee1=Levee("MR361","Holt County")
Levee2=Levee("ML256","Chase County")
Levee3=Levee("ML457","Gray County")
```

```

#Define levee system
LeveeSys1=Levee_System("Missouri River Right Bank","Nebraska")
LeveeSys2=Levee_System("Missouri River Left Bank","Missouri")

#Define Owners
Own1=Owner("USACE","MR361","Holt County")
Own2=Owner("ML Levee District","ML256","Chase County")
Own3=Owner("Levee District 1","ML457","Gray County")

#Define Eras
Era1=Construction_Era("1950","Missouri River Right Bank","Nebraska")
Era2=Construction_Era("1960","Missouri River Left Bank","Missouri")

#Define Condition
Cond1=Condition("Good","MR361","Holt County")
Cond2=Condition("Fair","ML256","Chase County")
Cond3=Condition("Poor","ML457","Gray County")

#Define Costs
Cost1=Repair_Cost("$100,000","MR361","Holt County","Missouri River Right Bank","Nebraska")
Cost2=Repair_Cost("$200,000","ML256","Chase County","Missouri River Left Bank","Missouri")
Cost3=Repair_Cost("$300,000","ML457","Gray County","Missouri River Left Bank","Missouri")

#print(Levee1)

print("The levee name for Levee 1 is:",Levee1.levée_name)
print("The location for levee system 1 is:",LeveeSys1.system_location)
print("The owner for levee system 1 is",Own1.system_owner)
print("Levee system 2 was constructed in:",Era2.const_era)
print("The condition of levee 2 is:",Cond2.levée_condition)
print("The repair cost for levee 3 is:",Cost3.levée_cost)

```

The results of this code are shown below. Each of the options is chosen.

```

"D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 2\Lab2\venv\Scripts\python.exe"
"D:/Google Drive/UMKC/PhD/Classes/Python/Labs/Lab 2/Lab2/Lab 2 Problem 3.py"

```

The levee name for Levee 1 is: MR361

The location for levee system 1 is: Nebraska

The owner for levee system 1 is USACE

Levee system 2 was constructed in: 1960

The condition of levee 2 is: Fair

The repair cost for levee 3 is: \$300,000

Process finished with exit code 0

Figure 3 illustrates the screenshot of the program for this problem.

The screenshot displays the PyCharm IDE interface. The top toolbar includes icons for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The project explorer on the left shows the file structure for 'Lab2', including 'venv', 'desktop.ini', 'fish.py', and four problem files. The main editor window shows the code for 'Lab 2 Problem 3.py'. The code is a Python script that defines a 'Levee' class and its subclasses, and then creates instances of these classes. The code is as follows:

```
1 #Don Baker
2 #Lab 2, Problem 3
3
4 #Write a python program to create any one of the following management systems.
5 #You can also pick one of your own.
6 #a. Library Management System (should have classes for Person, Student, Librarian, Book etc.)
7 #b. Airline Booking Reservation System (classes for Flight, Person, Employee, Passenger etc.)
8 #c. Hotel Reservation System (classes for Room, Occupants, Employee etc.)
9 #d. Student Enrollment System (classes for Student, System, Grades etc.)
10 #e. Expense Tracker System (classes for Expense, Transaction Category etc.)
11
12 #Prerequisites:
13 #Your code should have at least five classes.
14 #Your code should have __init__ constructor in all the classes
15 #Your code should show inheritance atleast once
16 #Your code should have one super call
17 #Use of self is required
18 #Use at least one private data member in your code.
19 #Use multiple inheritance atleast once
20 #Create instances of all classes and show the relationship between them.
21 #Your submission code should point out where all these things are present.
22
23
24 #Create the Python code for a levee inspection system
25
26 #Create class for Levee
```

The bottom panel shows the output of the program. The output is as follows:

```
Run: problem2_justin Lab 2 Problem 3
"D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 2\Lab2\venv\Scripts\python.exe" "D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 2\Lab2\Lab 2 Problem 3.py"
The levee name for Levee 1 is: MR361
The location for levee system 1 is: Nebraska
The owner for levee system 1 is USACE
Levee system 2 was constructed in: 1960
The condition of levee 2 is: Fair
The repair cost for levee 3 is: $300,000
Process finished with exit code 0
```

The status bar at the bottom shows the file encoding as 'PEP 8: block comment should start with #', the cursor position as '2:18', and the file encoding as 'CRLF', 'UTF-8'. The system clock shows '11:35 PM 2/15/2018'.

Figure 3 Screenshot Illustrating the Levee Inspection System Code and Results

4. Using Numpy create random vector of size 15 having only Integers in the range 0 -20. Write a program to find the most frequent item/value in the vector list.

Sample input:

[1,2,16,14,6,5,9,9,20,19,18]

Sample output:

Most frequent item in the list is 9

Results:

The following code was developed to create a a vector with 15 integers ranging from 0 to 20. The code will return the most frequent value in the list.

```
#Don Baker
#Lab 2

#Using Numpy create random vector of size 15 having only Integers in the range 0 - 20.
#Write a program to find the most frequent item/value in the vector list.

#Import the Numpy package
import numpy as np

#Ask the user to enter the low and high range for the random integer vector, and the
size of the vector.
l=int(input("Please enter the low integer value in the range of the random vector. The
lowest value allowed is zero.\n"))
h=int(input("Please enter the high integer value in the range of the random
vector.\n"))
s=int(input("Please enter the size of the vector."))

#Create random vector of integers from 0 to 20
Vec = np.random.randint(l,h,s)

#Print vector
print("\nHere is the random vector.")
print(Vec)

#Find and print the most frequent integer in the random vector
most_freq=np.bincount(Vec).argmax()
print("\nThe most frequent integer in the random vector is",most_freq,".")
```

The results of this code are shown below.

```
"D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 2\Lab2\venv\Scripts\python.exe"
"D:/Google Drive/UMKC/PhD/Classes/Python/Labs/Lab 2/Lab2/Lab 2 Problem 4.py"
```

Please enter the low integer value in the range of the random vector. The lowest value allowed is zero.

0

Please enter the high integer value in the range of the random vector.

20

Please enter the size of the vector.15

Here is the random vector.

```
[19  5  4 12  7  8 14 17  2 14  9 18 10  5 11]
```

The most frequent integer in the random vector is 5 .

Process finished with exit code 0

Figure 4 illustrates the screenshot of the program for this problem.

The screenshot displays the PyCharm IDE interface. The top toolbar includes icons for Run, Debug, and Search. The Project view on the left shows the file structure, with 'Lab2 Problem 4.py' selected. The main editor window shows the following Python code:

```
1 #Don Baker
2 #Lab 2
3
4 #Using Numpy create random vector of size 15 having only Integers in the range 0 - 20.
5 #Write a program to find the most frequent item/value in the vector list.
6
7 #Import the Numpy package
8 import numpy as np
9
10 #Ask the user to enter the low and high range for the random integer vector, and the size of the vector.
11 l=int(input("Please enter the low integer value in the range of the random vector. The lowest value allowed is zero.\n"))
12 h=int(input("Please enter the high integer value in the range of the random vector.\n"))
13 s=int(input("Please enter the size of the vector.\n"))
14
15 #Create random vector of integers from 0 to 20
16 Vec = np.random.randint(1,h,s)
17
18 #Print vector
19 print("\nHere is the random vector.")
20 print(Vec)
21
22 #Find and print the most frequent integer in the random vector
23 most_freq=np.bincount(Vec).argmax()
24 print("\nThe most frequent integer in the random vector is",most_freq,".")
```

The Run window at the bottom shows the execution output for 'Lab 2 Problem 4':

```
"D:\Google Drive\UMKC\PhD\Classes\Python\Labs\Lab 2\Lab2\venv\Scripts\python.exe" "D:/Google Drive/UMKC/PhD/Classes/Python/Labs/Lab 2/Lab2/Lab 2 Problem 4.py"
Please enter the low integer value in the range of the random vector. The lowest value allowed is zero.
0
Please enter the high integer value in the range of the random vector.
20
Please enter the size of the vector.15

Here is the random vector.
[19  5  4 12  7  8 14 17  2 14  9 18 10  5 11]

The most frequent integer in the random vector is 5 .

Process finished with exit code 0
```

The Windows taskbar at the bottom shows the system clock as 9:23 PM on 2/15/2018.

Figure 4 Screenshot Illustrating the Most Frequent Value in a 15 Value Random Vector