

**CSEE 5590 0001**  
**Special Topics SPRING 2018**

**PHYTHON LAB ASSIGNMENT - 2**

Submitted On 2/16/2018

Name: Satya Sai Deepthi Katta

Class ID: 21



**Department of Computer Science and Electrical Engineering**

## **DOCUMENT CONTENTS**

1. Author
2. Objective
3. Features
4. Configuration
5. Input/Output Screenshots
6. Implementation & Code Snippet
7. Deployment
8. Limitations
9. References

## **AUTHOR**

This document is a part of Lab Assignment #2 submitted by SATYA SAI DEEPTHI KATTA (ID: 21), a graduate student majoring in computer science at University of Missouri Kansas City. The lab assignment carried out under Python and Deep Learning course(CS5590) taught by Dr. Yungyug Lee along with Rashmi, Saira and Vijaya Yeruva.

## **OBJECTIVE**

The main aim of the lab work is to create an exposure to some concepts in Python language like:

- Sets and Dictionaries
- Numpy packages
- Building Management systems

The assignment is divided into four tasks which focuses to make one familiar with the python concepts listed above.

- To print list of books in the given price range.
- To perform user specified operations on the contact list.
- To create a management system using listed classes.
- To display the most frequent value in the vector list.

## FEATURES

The features involved in each task are documented below.

### Task 1:

#### ***Price Range Books List***

In UMKC book store, a dictionary is available with all the books and their prices. The main feature of the task is to write a program to find the books in the range given by the user.

### Task 2:

#### ***Performing Operations on Contact List***

The task is to perform operations on the contact list of the mobile. The operations include

- Displaying contact details by searching with name.
- Displaying contact details by searching with number.
- To edit a contact by name.
- Exit the program

### Task 3:

#### ***Create Management System***

This task involves building a management system among the list given with all the classes. The management system must have

- atleast five classes,
- should have `_init_` constructor in all the classes,
- the code should show inheritance atleast once and use multiple inheritance.
- code should use 'self' and should have one super call.
- usage of atleast one private data member in the code.
- instances need to be created for all the classes.
- code must point out all the features listed above with comments.

### Task 4:

#### ***Printing Frequent Value of List***

The simple task to display the most frequently occurring number in the random vector list of 15 numbers ranging from 0 to 20.

# CONFIGURATION

For executing the tasks given in the lab assignment, advanced version of **Python 3.6.4** is used and the code is built in **PYCHARM** Software.

## INPUT/OUTPUT SCREENSHOTS

### Task 1:

For this task, a list of books along with their prices are given. The price ranges are given by the user. The list of books with their prices in the given price range is given as output.

```
C:\Users\satyasaideepthi\PycharmProjects\Lab2\venv\Scripts\python.exe C:/Users/satyasaideepthi/PycharmProjects/Lab2/task1.py
{'Python': '15', 'Web': '30', 'C': '20', 'Java': '40', 'DeepLearning': '25'}
Enter the Initial Range: 30
Enter the Final Range: 40
You can purchase the books listed below:
Web : 30
Java : 40

Process finished with exit code 0
|
```

### Task 2:

This task involves performing operations on the contact list.

-For displaying contact by name, a name is given by user and the complete details of the name are given like name, number and email as output.

- For displaying contact by number is similar as above but the input given is a contact number and the output is the result of the contact details linked to the inputted number.

```
C:\Users\satyasaideepthi\PycharmProjects\Lab2\venv\Scripts\python.exe C:/Users/satyasaideepthi/PycharmProjects/Lab2/task2.py
Would you like to:
    1.) Display contact by Name
    2.) Display contact by Number
    3.) Edit a Contact by Name
    4.) Quit
Select an option: 1
Enter Name: rocky
Contact details of the entered name:
{'name': 'rocky', 'number': 3456789012, 'email': 'rocky@gmail.com'}
Would you like to:
    1.) Display contact by Name
    2.) Display contact by Number
    3.) Edit a Contact by Name
    4.) Quit
Select an option: 2
Enter Contact: 2345678901
Details of the entered number:
{'name': 'sandy', 'number': 2345678901, 'email': 'sandy@gmail.com'}
Would you like to:
    1.) Display contact by Name
    2.) Display contact by Number
    3.) Edit a Contact by Name
    4.) Quit
Select an option: 3
```

To edit the contact by name, the input needs to be given is a name for which number needs to be changed. After inserting the name, new number needs to be given by the user. After successful modification of the number, output displaying the modified number along with all contacts is given on console window.

The option quits the program by displaying a message.

```
Select an option: 3
Enter name to edit its contact: satya
{'name': 'satya', 'number': 1234567890, 'email': 'satya@gmail.com'}
Enter New Number: 9999999999
Contact Modified!
Contact list with modified contact:
{'name': 'satya', 'number': 9999999999, 'email': 'satya@gmail.com'}
{'name': 'sandy', 'number': 2345678901, 'email': 'sandy@gmail.com'}
{'name': 'rocky', 'number': 3456789012, 'email': 'rocky@gmail.com'}
Would you like to:
    1.) Display contact by Name
    2.) Display contact by Number
    3.) Edit a Contact by Name
    4.) Quit
Select an option: 4
Program Quit!

Process finished with exit code 0
|
```

### **Task 3:**

Traveler's details are given as the input for the classes, the confirmation printing all the details is given as output for the task using multiple inheritance.

```
C:\Users\satyasaideepthi\PycharmProjects\Lab2\venv\Scripts\python.exe C:/Users/satyasaideepthi/PycharmProjects/Lab2/task3.py
--From the Person class--
Traveler's First Name:Samyu
Traveler's Last Name:Khanna
Traveler's Age:22
Traveler's Ticket ID:1234
--After using Multiple Inheritance,From Passenger class--
Traveler's First Name:Sydney
Traveler's Last Name:Sheldon
Traveler's Age:42
Reserved Seat Number: 15
Reserved Seat Letter: K
Airlines Class:Economy
Airlines Name: Delta
Airlines Number: 3435
Traveler's Ticket ID:1435

Process finished with exit code 0
```

### **Task 4:**

Random vector list is given as the input and the most frequently occurring number in the random list is given as output for this task.

```
C:\Users\satyasaideepthi\PycharmProjects\Lab2\venv\Scripts\python.exe C:/Users/satyasaideepthi/PycharmProjects/Lab2/task4.py
Generated Random Vector List:
[17 11 5 17 17 1 15 8 4 10 14 17 17 15 11]
Most frequent value in the vector list:
17

Process finished with exit code 0
```

## IMPLEMENTATION & CODE SNIPPET

### Task 1:

A dictionary list of all the books along with the prices are considered in dic\_list. First and last values of the range are taken from the user.

A for loop is used for iterating through the dictionary list items with the limits being the range. Whenever the cost is in between the range, then the book name with its cost is printed.

```
dic_list={"Python" : "15", "Web" : "30", "C" : "20", "Java" : "40", "DeepLearning" : "25"}
print(dic_list) #printing the list if dictionary items available
#initializing the range values
first=int(input("Enter the Initial Range: "))
last=int(input("Enter the Final Range: "))
print("You can purchase the books listed below:")
for name,cost in dic_list.items(): # for each book name and cost
    if int(cost) in range(first,last+1): #if the cost of book is in price range
        print('%s : %s' %(name, cost)) #the book name is printed
```

### Task 2:

A contact\_list is created which contains all the names, numbers and the email ID of the people in the mobile.

While loop is used for performing the operations on the contact list.

```
contact_list = [{"name":"satya", "number":1234567890, "email":"satya@gmail.com"},
                {"name":"sandy", "number":2345678901, "email":"sandy@gmail.com"},
                {"name":"rocky", "number":3456789012, "email":"rocky@gmail.com"}]

while True:
    print('' Would you like to:
          1.) Display contact by Name
          2.) Display contact by Number
          3.) Edit a Contact by Name
          4.) Quit'' )
```

If-else conditional statements are used along with for loops for displaying the desired outputs.

Sel variable is user defined variable for selecting any of the options available. For displaying contact details using name and number applies with the same logic.

The user given name or number is stored in a variable and the variable is iterated in the loop until it matches with any one of the values on the contact list. If given input matches the value then output prints all the details of the given input from contact list.

```
sel = input("Select an option: ") # for selecting a option in the given list

if sel == "1": #if selection is 1 then display contact details by name
    name = input("Enter Name: ") #enter a name to search
    for i in contact_list: # for loop to find name is present in contact list
        if name in i.values():
            print("Contact details of the entered name:")
            print(i) #details are printed

elif sel == "2": #if selection is 2 then display contact details by number
    num = int(input("Enter Contact: ")) #enter a number to search
    for j in contact_list: # for loop to find number is present in contact list
        if num in j.values():
            print("Details of the entered number:")
            print(j)
```

For editing the number, a contact name is taken as the input and the new number is requested from the user is stored in a new variable. Then the old number is replaced with the new variable number and the modified contact is stored. Finally print statement is given at the end of for loop to display the contact list along with modified contact.

After executing all the selections, break statement stops the execution by getting it out of the while loop.

```
elif sel == "3":
    in_name = input("Enter name to edit its contact: ")
    for k in contact_list: #for loop to search for entered name
        if in_name in k.values():
            print(k) #displaying entered name details
            newnum = int(input("Enter New Number: "))
            k["number"] = newnum #new number is assigned to the contact
            print("Contact Modified!")
            #print(k)
            print("Contact list with modified contact:")
            print(k) #printing contact list after modifying the contact

elif sel == "4":
    print("Program Quit!")
    break
else:
    print("Invalid Input! Try again.") #if selected is in not among 4 given options
    break
```



### **Task 3:**

Airline Booking Reservation System is created with five classes namely: Booking, Airlines, Seat, Person, Passenger.

Booking class is used for booking the ticket either in economy or business class.

Airlines class is for booking a flight giving its flight number.

```
# AIRLINE BOOKING RESERVATION SYSTEM #

class Booking:          #class to book ticket
    def __init__(self, e, b): #initializing constructor
        self.economy_class = e
        self.business_class = b

    def get_Economy_class(self): # function to book economy class seat
        print("Airlines Class:" + str(self.economy_class))

    def get_Business_class(self): #funtion to book business class seat
        print("Airlines Class:" + str(self.business_class))

class Airline:          # class to book a flight ticket
    def __init__(self, num, name):      # _init_ constructor intializing value
        self.flight_number = num
        self.flight_name = name

    def get_flight_number(self):
        print("Airlines Number: " + str(self.flight_number))

    def get_flight_name(self):
        print("Airlines Name: " + str(self.flight_name))
```

Seat class gives the allocated seat number and the seat letter. While travelers details are taken in the Person class using multiple functions.

```
class Seat:          # class to book a seat for the ticket
    def __init__(self, n, p): #init constructor initializing seat number and seat letter
        self.seat_number = n
        self.seat_letter = p

    def get_seat_number(self):
        print("Reserved Seat Number: " + str(self.seat_number))

    def get_seat_letter(self):
        print("Reserved Seat Letter: " + str(self.seat_letter))
```

In Person class, a private data member is used “\_\_get\_SSN()”, which can’t be accessed in the inherited class.

```

class Person:      # Class person gets the details of the traveller
    count = 0

    def __init__(self, a, b, c, d, e):    #init constructor
        self.first_name = a      #getting persons first and last name
        self.last_name = b      #along with age and ID
        self.age = c
        self.ID = d
        self.SSN = e

    Person.count += 1 #count is incremented for each traveller
    #functions are created to get travellers details

    def get_first_name(self):
        print("Traveler's First Name:" + str(self.first_name))

    def get_last_name(self):
        print("Traveler's Last Name:" + str(self.last_name))

    def get_age(self):
        print("Traveler's Age:" + str(self.age))

    def get_ID(self):
        print("Traveler's Ticket ID:" + str(self.ID))

    def __get_SSN(self):    #using private data member
        print("Traveler's SSN:" + str(self.SSN))

```

Passenger class is created using multiple inheritance, above four classes are inherited in this class. Several parameters are given as the input for the passenger to access all the methods from the super classes listed above.

```

# Using Multiple Inheritance..
# Passenger class created with inheriting all the above four classes
class Passenger(Person, Seat, Booking, Airline):
    def __init__(self, f, l, a, id, ssn, fn, fno, sn, sno, e, b):    #init constructor
        Person.__init__(self, f, l, a, id, ssn)
        Airline.__init__(self, fn, fno)
        Seat.__init__(self, sn, sno)
        Booking.__init__(self, e, b)    #calling the super class constructor

#
print("--After using Multiple Inheritance,From Passenger class--")
person2 = Passenger("Sydney", "Sheldon", "42", "1435", "123456789", "3435", "Delta", "15", "K", "Economy","Business")
person2.get_first_name()
person2.get_last_name()
person2.get_age()
#person2.__get_SSN()
person2.get_seat_number()
person2.get_seat_letter()
person2.get_Economy_class()
person2.get_flight_name()
person2.get_flight_number()
person2.get_ID()

```

While accessing the data member from the Person class in Passenger class, error is displayed showing that attribute couldn't be found.

```

C:\Users\satyasaideepthi\PycharmProjects\Lab2\venv\Scripts\python.exe C:/Users/satyasaideepthi/PycharmProjects/Lab2/task3.py
--From the Person class--
Traceback (most recent call last):
Traveler's First Name:Samyu
Traveler's Last Name:Khanna
Traveler's Age:22
Traveler's Ticket ID:1234
  File "C:/Users/satyasaideepthi/PycharmProjects/Lab2/task3.py", line 94, in <module>
--After using Multiple Inheritance,From Passenger class--
    person2.__get_SSN()
Traveler's First Name:Sydney
Traveler's Last Name:Sheldon
AttributeError: 'Passenger' object has no attribute '__get_SSN'
Traveler's Age:42

Process finished with exit code 1

```

## **Task 4:**

Numpy library is imported into the variable np, random.randint creates the list of numbers with the given ranges for the limit.

Bincount used for counting number of occurrences of each value in the list and the argmax gives the maximum of the bincount list.

```

import numpy as np
#random vector list of integers generated using numpy library
vector_list = np.random.randint(0, 20, 15)
print("Generated Random Vector List:")
print(vector_list) #list is printed
print("Most frequent value in the vector list:")
#np.bincount is used to count number of occurrences of each value in the list
#argmax gives the maximum of the bincount occurrences of the list
print(np.bincount(vector_list).argmax())

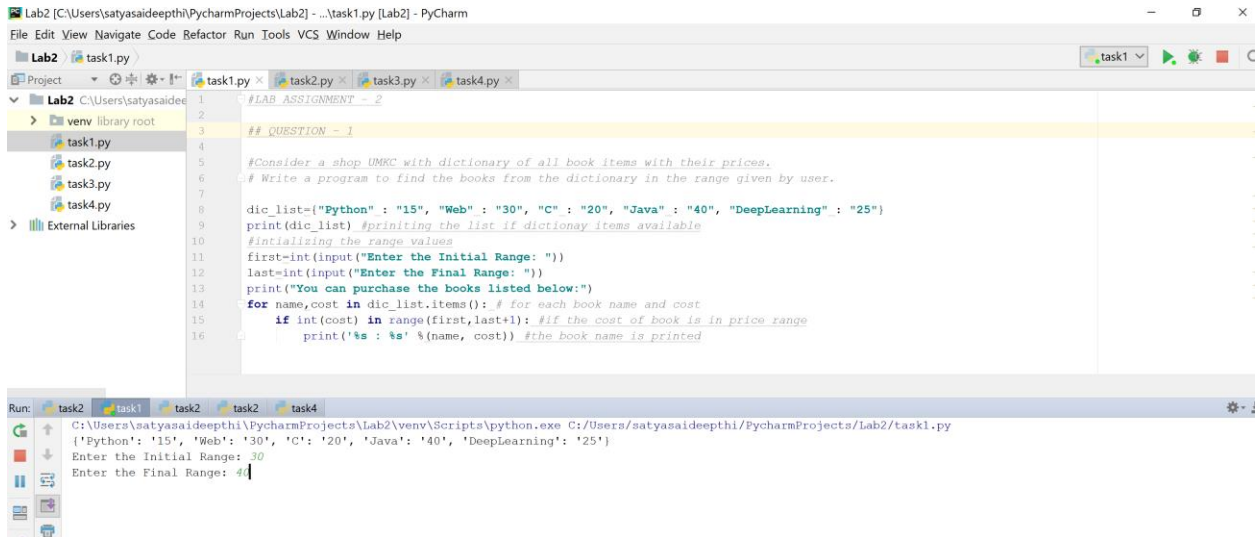
```

# DEPLOYMENT

For this assignment, all the code is deployed on PYCHARM software saving all the program files in .py format. The console window is used for giving the inputs and for getting the outputs from the code.

## Task 1 Deployment:

Step 1: Set the numbers for the price range



The screenshot shows the PyCharm IDE with a project named 'Lab2'. The file explorer on the left shows a directory structure with 'venv' and 'task1.py' through 'task4.py'. The main editor window displays the code for 'task1.py'. The code defines a dictionary 'dic\_list' with book names and prices, prompts the user for an initial and final range, and prints the books within that range. The Run window at the bottom shows the execution output, including the dictionary contents and the range input.

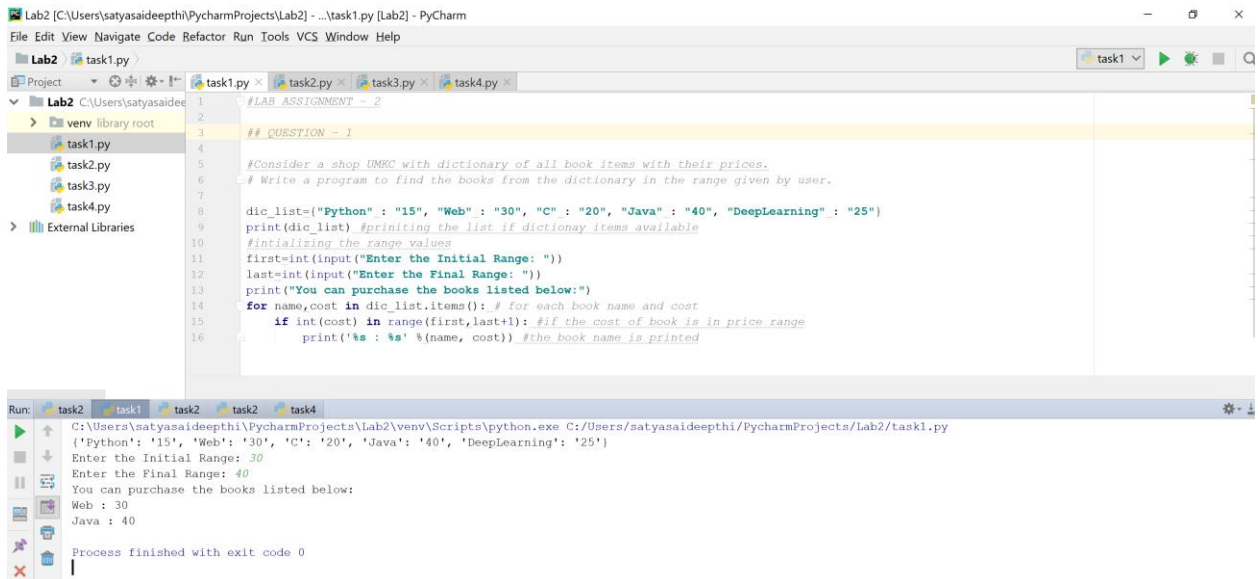
```
#LAB ASSIGNMENT - 2
## QUESTION - 1

#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.

dic_list={"Python": "15", "Web": "30", "C": "20", "Java": "40", "DeepLearning": "25"}
print(dic_list) #printing the list if dictionary items available
#initializing the range values
first=int(input("Enter the Initial Range: "))
last=int(input("Enter the Final Range: "))
print("You can purchase the books listed below:")
for name,cost in dic_list.items(): # for each book name and cost
    if int(cost) in range(first,last+1): #if the cost of book is in price range
        print('%s : %s' %(name, cost)) #the book name is printed
```

Run: task2 task1 task2 task2 task4  
C:\Users\satyasaideepthi\PycharmProjects\Lab2\venv\Scripts\python.exe C:/Users/satyasaideepthi/PycharmProjects/Lab2/task1.py  
{'Python': '15', 'Web': '30', 'C': '20', 'Java': '40', 'DeepLearning': '25'}  
Enter the Initial Range: 30  
Enter the Final Range: 40

Step 2: Get the output list within the range in the console window.



This screenshot shows the same PyCharm IDE setup as the previous one, but the Run window now displays the filtered output. It shows the dictionary, the range input, and the list of books that fall within the specified price range (30 to 40).

```
#LAB ASSIGNMENT - 2
## QUESTION - 1

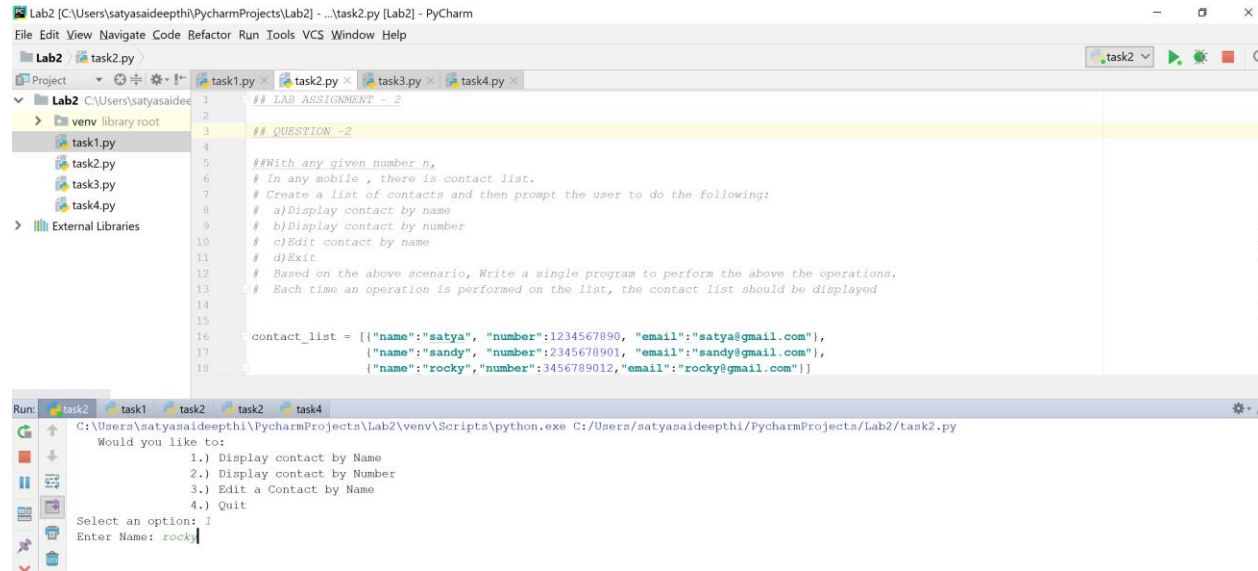
#Consider a shop UMKC with dictionary of all book items with their prices.
# Write a program to find the books from the dictionary in the range given by user.

dic_list={"Python": "15", "Web": "30", "C": "20", "Java": "40", "DeepLearning": "25"}
print(dic_list) #printing the list if dictionary items available
#initializing the range values
first=int(input("Enter the Initial Range: "))
last=int(input("Enter the Final Range: "))
print("You can purchase the books listed below:")
for name,cost in dic_list.items(): # for each book name and cost
    if int(cost) in range(first,last+1): #if the cost of book is in price range
        print('%s : %s' %(name, cost)) #the book name is printed
```

Run: task2 task1 task2 task2 task4  
C:\Users\satyasaideepthi\PycharmProjects\Lab2\venv\Scripts\python.exe C:/Users/satyasaideepthi/PycharmProjects/Lab2/task1.py  
{'Python': '15', 'Web': '30', 'C': '20', 'Java': '40', 'DeepLearning': '25'}  
Enter the Initial Range: 30  
Enter the Final Range: 40  
You can purchase the books listed below:  
Web : 30  
Java : 40  
Process finished with exit code 0

## Task 2 Deployment:

Step 1: Enter the selection of the operation to be performed. For displaying contact by name, give the input as name.

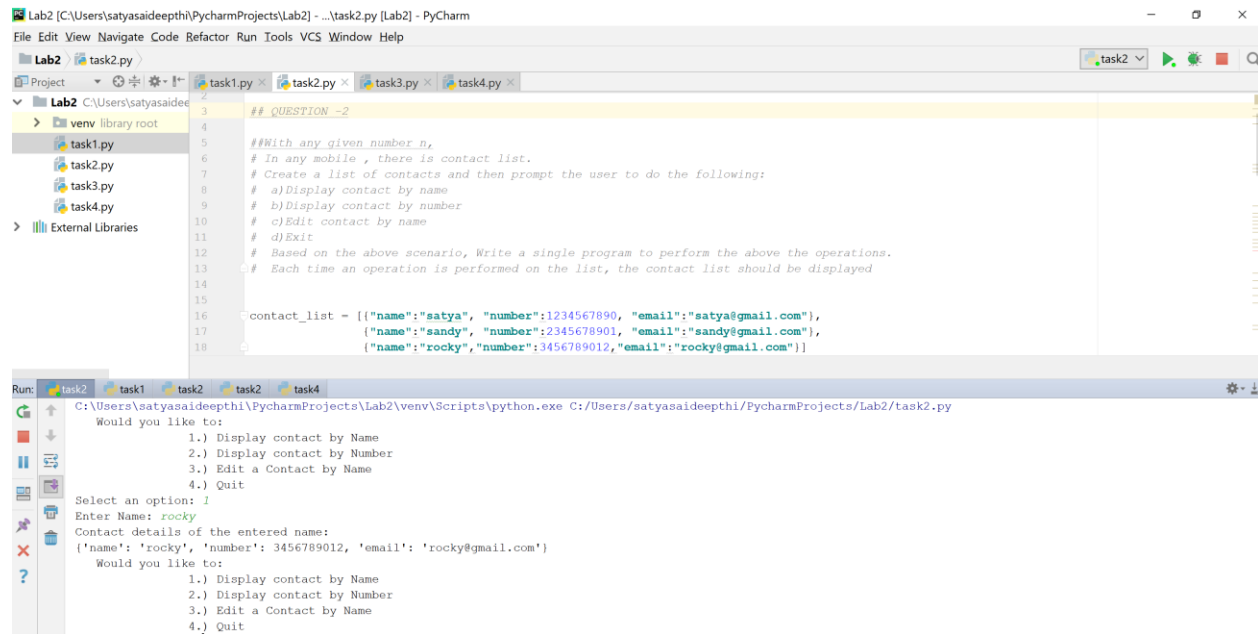


The screenshot shows the PyCharm IDE with a project named 'Lab2'. The file explorer on the left shows a directory structure with 'venv' and 'task1.py' through 'task4.py'. The main editor displays 'task2.py' with the following code:

```
1  ## LAB ASSIGNMENT - 2
2
3  ## QUESTION -2
4
5  ##With any given number n,
6  # In any mobile , there is contact list.
7  # Create a list of contacts and then prompt the user to do the following:
8  # a)Display contact by name
9  # b)Display contact by number
10 # c)Edit contact by name
11 # d)Exit
12 # Based on the above scenario, Write a single program to perform the above the operations.
13 # Each time an operation is performed on the list, the contact list should be displayed
14
15
16 contact_list = [{"name":"satya", "number":1234567890, "email":"satya@gmail.com"},
17                 {"name":"sandy", "number":2345678901, "email":"sandy@gmail.com"},
18                 {"name":"rocky", "number":3456789012, "email":"rocky@gmail.com"}]
```

The Run console at the bottom shows the execution of 'task2.py'. It prompts 'Would you like to:' followed by a list of options: 1.) Display contact by Name, 2.) Display contact by Number, 3.) Edit a Contact by Name, 4.) Quit. The user selects option 1, and then enters 'rocky' as the name.

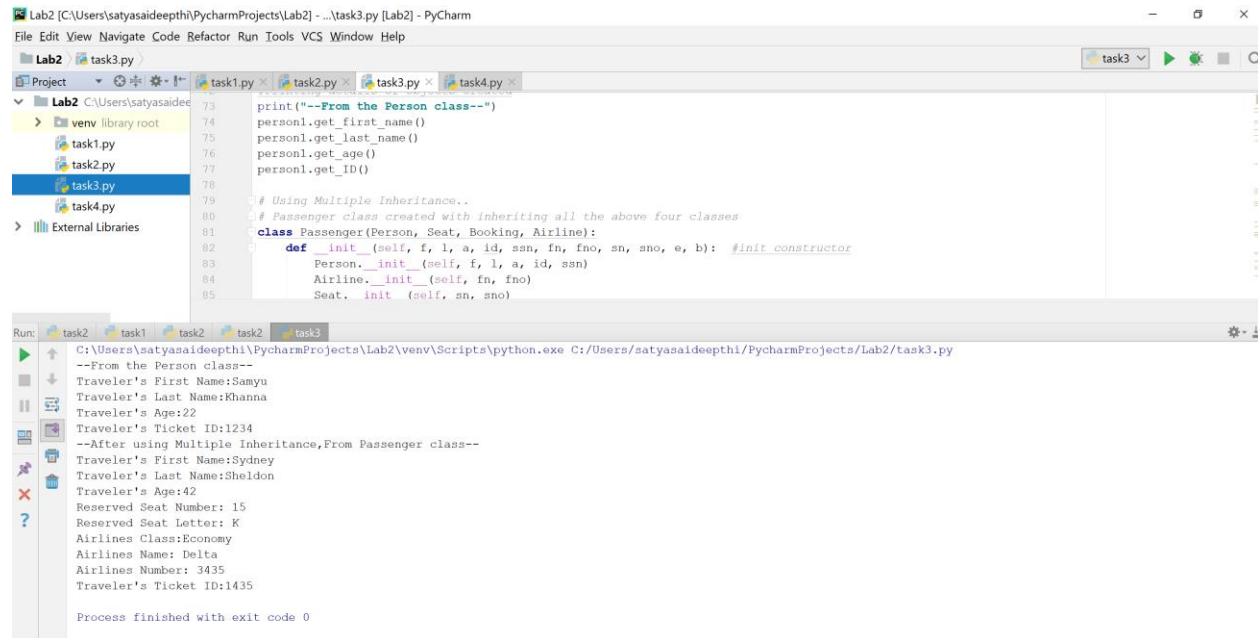
Step 2: Get the output in the console. And select any option till using the option 4 to end the program.



The screenshot shows the PyCharm IDE with the same project and code as the previous screenshot. The Run console now shows the output after selecting option 1 and entering 'rocky'. It displays the contact details for 'rocky' and then prompts 'Would you like to:' followed by the same list of options. The user has not yet selected an option in this screenshot.

## Task 3 Deployment:

The passenger details are already given as the input in the program, the detailed accessing of the methods is shown in the console window printing the passengers airline booking confirmation.



```
Lab2 [C:\Users\satyasaideepthi\PycharmProjects\Lab2] - ...task3.py [Lab2] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help

Project: Lab2
venv library root
task1.py
task2.py
task3.py
task4.py
External Libraries

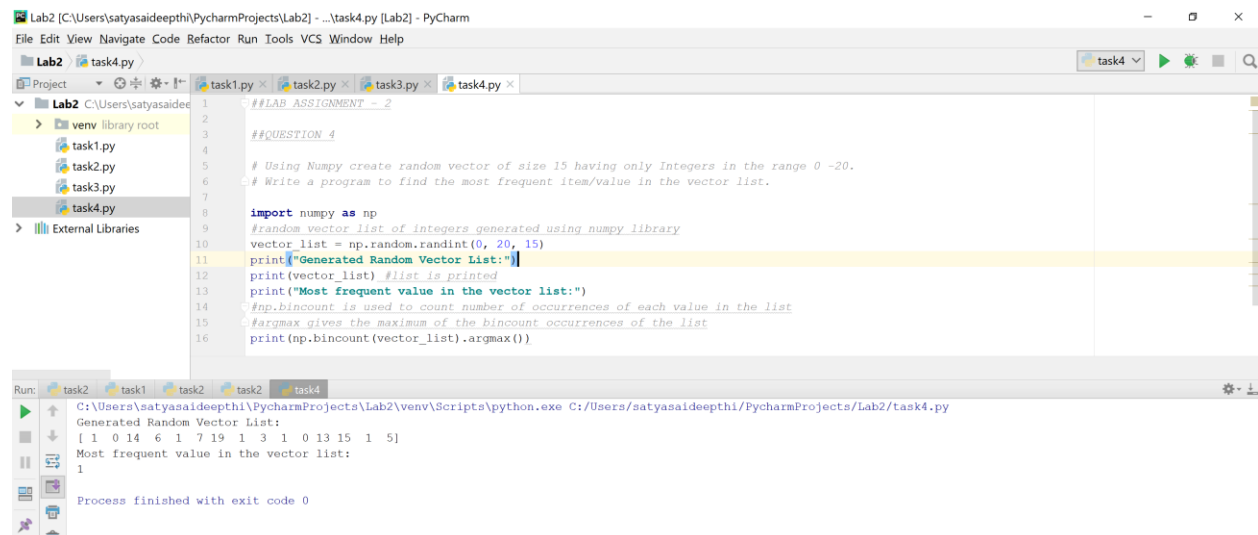
task3.py
73 print("--From the Person class--")
74 person1.get_first_name()
75 person1.get_last_name()
76 person1.get_age()
77 person1.get_ID()
78
79 # Using Multiple Inheritance..
80 # Passenger class created with inheriting all the above four classes
81 class Passenger(Person, Seat, Booking, Airline):
82     def __init__(self, f, l, a, id, ssn, fn, fno, sn, sno, e, b): #init constructor
83         Person.__init__(self, f, l, a, id, ssn)
84         Airline.__init__(self, fn, fno)
85         Seat.__init__(self, sn, sno)

Run: task2 task1 task2 task2 task3
C:\Users\satyasaideepthi\PycharmProjects\Lab2\venv\Scripts\python.exe C:/Users/satyasaideepthi/PycharmProjects/Lab2/task3.py
--From the Person class--
Traveler's First Name:Samyu
Traveler's Last Name:Khanna
Traveler's Age:22
Traveler's Ticket ID:1234
--After using Multiple Inheritance,From Passenger class--
Traveler's First Name:Sydney
Traveler's Last Name:Sheldon
Traveler's Age:42
Reserved Seat Number: 15
Reserved Seat Letter: K
Airlines Class:Economy
Airlines Name: Delta
Airlines Number: 3435
Traveler's Ticket ID:1435

Process finished with exit code 0
```

## Task 4 Deployment:

As the input is generating the random list, the output is directly displayed in the output console.



```
Lab2 [C:\Users\satyasaideepthi\PycharmProjects\Lab2] - ...task4.py [Lab2] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help

Project: Lab2
venv library root
task1.py
task2.py
task3.py
task4.py
External Libraries

task4.py
1 ##LAB ASSIGNMENT - 2
2
3 ##QUESTION 4
4
5 # Using Numpy create random vector of size 15 having only Integers in the range 0 -20.
6 # Write a program to find the most frequent item/value in the vector list.
7
8 import numpy as np
9 #Random vector list of integers generated using numpy library
10 vector_list = np.random.randint(0, 20, 15)
11 print("Generated Random Vector List:")
12 print(vector_list) #list is printed
13 print("Most frequent value in the vector list:")
14 #np.bincount is used to count number of occurrences of each value in the list
15 #argmax gives the maximum of the bincount occurrences of the list
16 print(np.bincount(vector_list).argmax())

Run: task2 task1 task2 task2 task4
C:\Users\satyasaideepthi\PycharmProjects\Lab2\venv\Scripts\python.exe C:/Users/satyasaideepthi/PycharmProjects/Lab2/task4.py
Generated Random Vector List:
[ 1  0 14  6  1  7 19  1  3  1  0 13 15  1  5]
Most frequent value in the vector list:
1

Process finished with exit code 0
```

## **LIMITATIONS**

The provided code meets all the requirements of the tasks given. But few limitations of the code are listed below:

- In Task 2, names given as input needs to be case sensitive and the contact number accepting doesn't limit to 10 digits.
- 

## **REFERENCES**

- [1] <https://codereview.stackexchange.com/questions/127229/program-to-edit-an-address-book-stored-as-a-json-file>
- [2] <https://codereview.stackexchange.com/questions/49124/airline-hotel-reservation-system-in-python-follow-up>
- [3] <https://stackoverflow.com/questions/1518522/python-most-common-element-in-a-list>