Python and Deep Learning

Lab Assignment 2

Submitted by

Vamsi Krishna Challa

16242688

**Objective:**

1. To perform various operations like Manipulating, accessing, assessing, comparing lists and dictionaries.
2. Implementing and accessing various features of NumPy package.
3. Performing various functions on classes.

**Features:**

In this assignment we access both python inbuilt features and by installing required packages to

1. Access dictionaries to perform required operations on them.
2. Creating, accessing, manipulating the lists to produce user desired results
3. Implementing the object-oriented features of python i.e., class to perform functions more efficiently.
4. Installing and accessing NumPy package to perform operations more easily using inbuilt functions of that package.

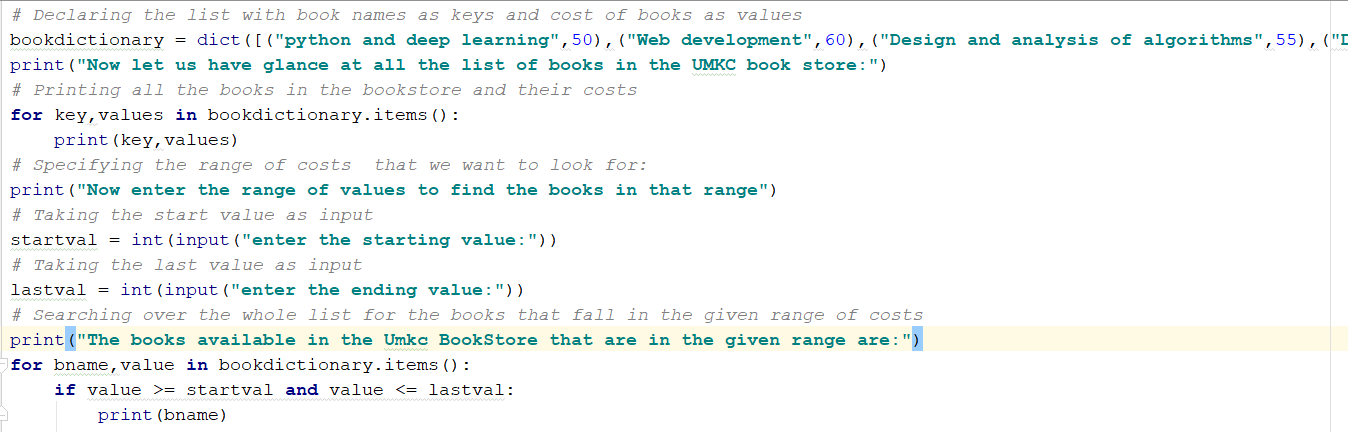
**Configuration:**

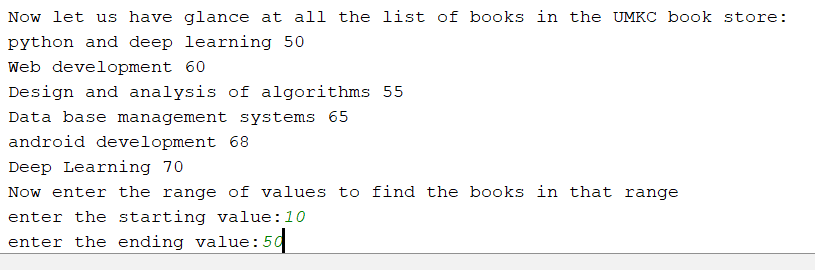
Software used: Python 3.4

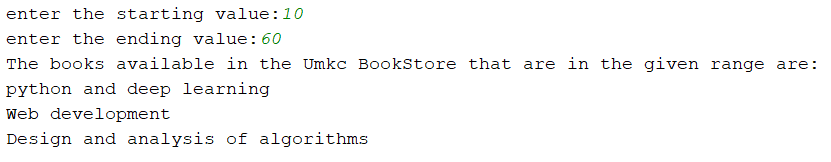
IDE: PyCharm

**Input / Output:**

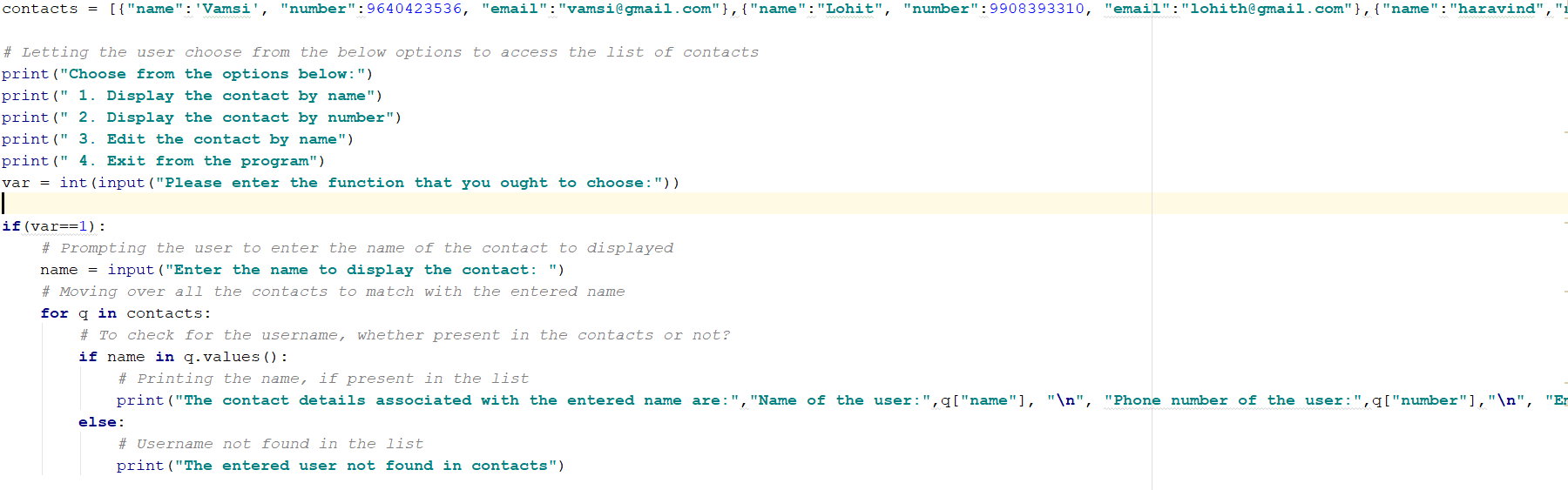
1. **Program for finding the books in the specified range of cost:**

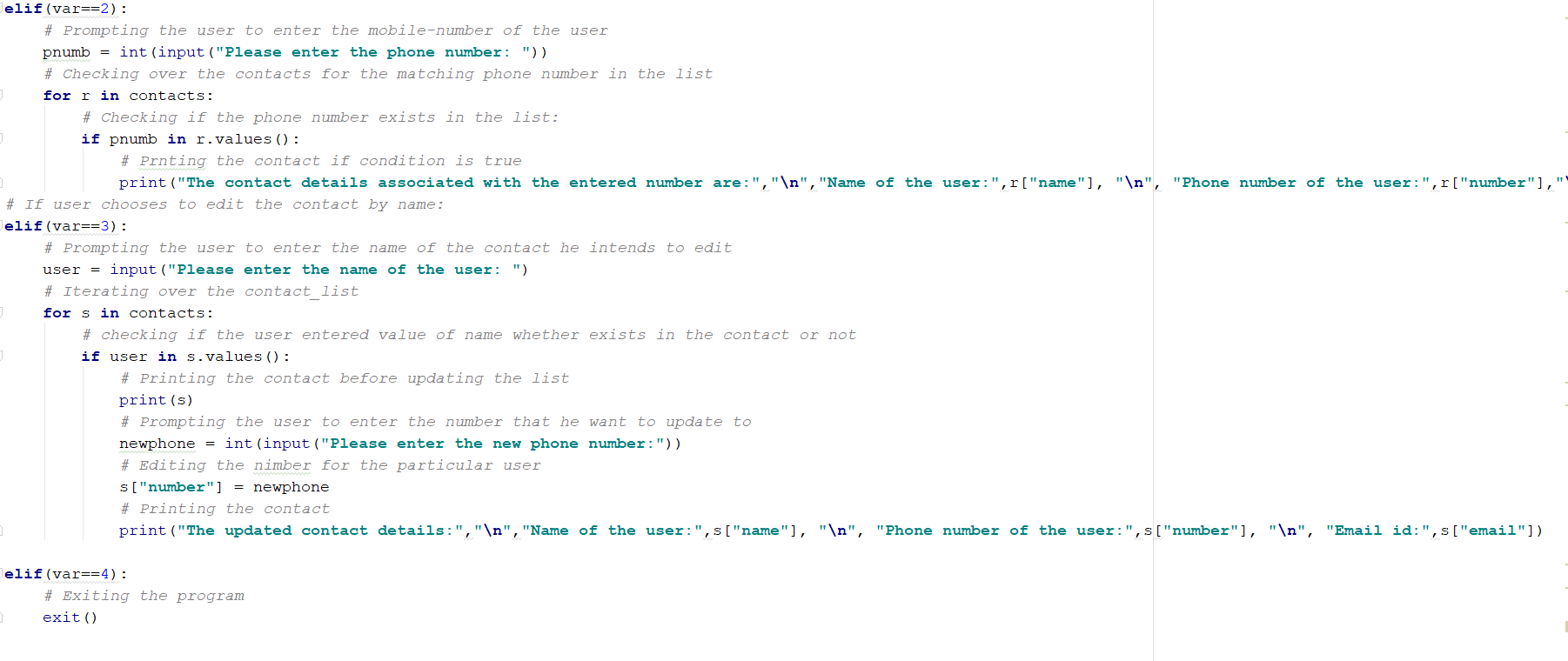




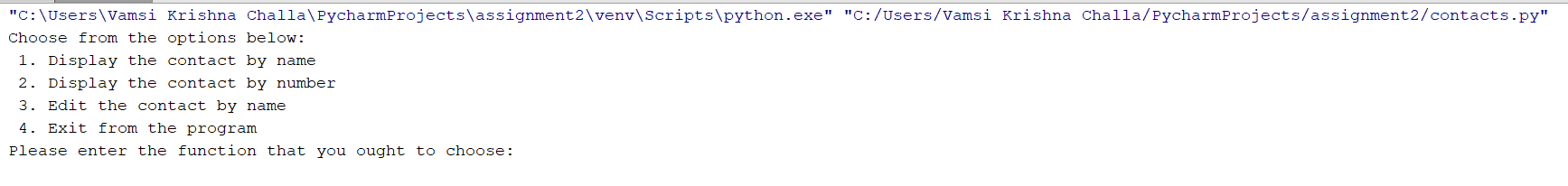


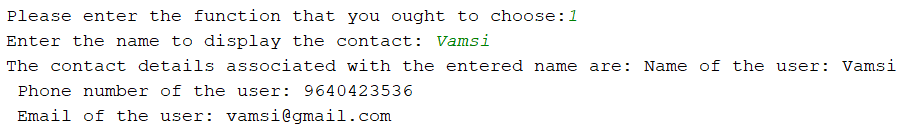
1. **Program to create, access and modify contact details:**

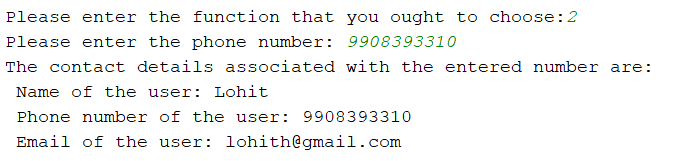


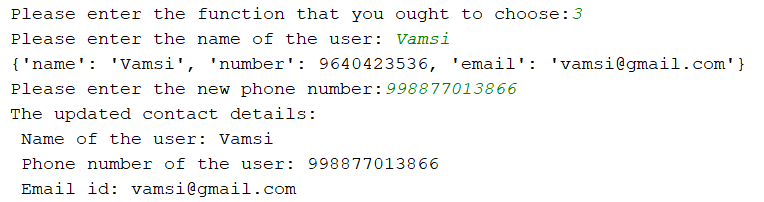


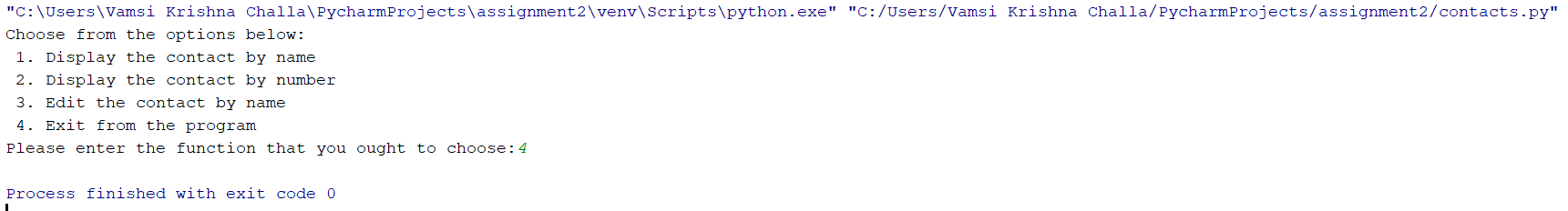
**Outputs:**



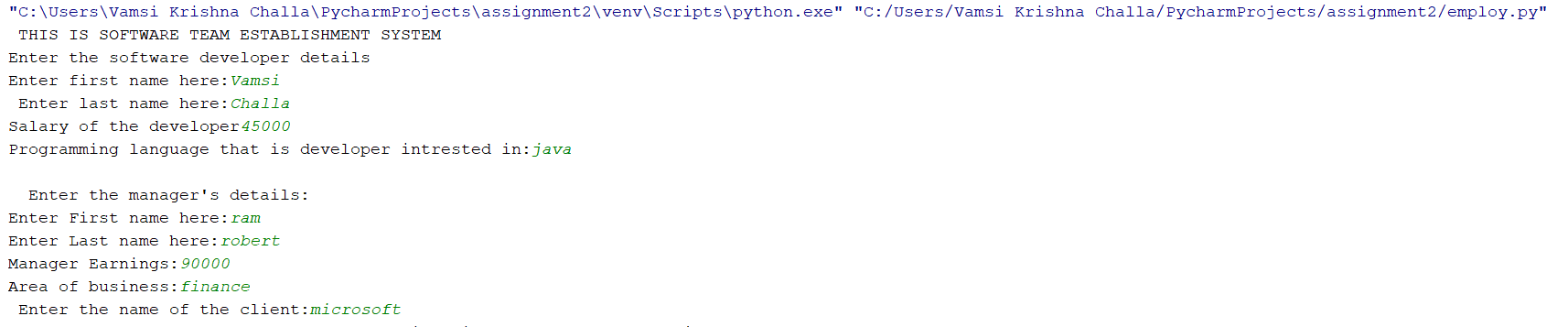


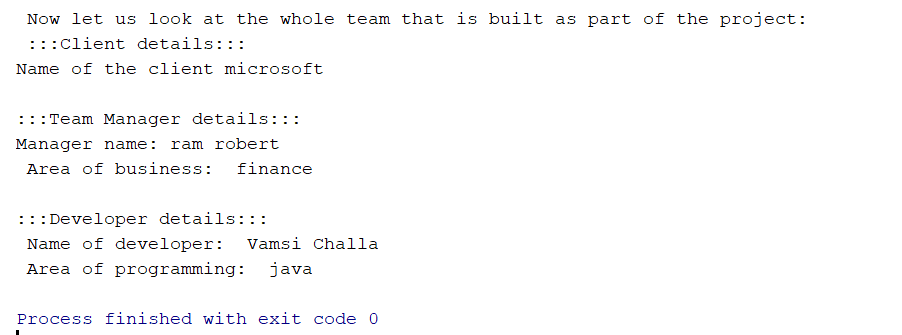




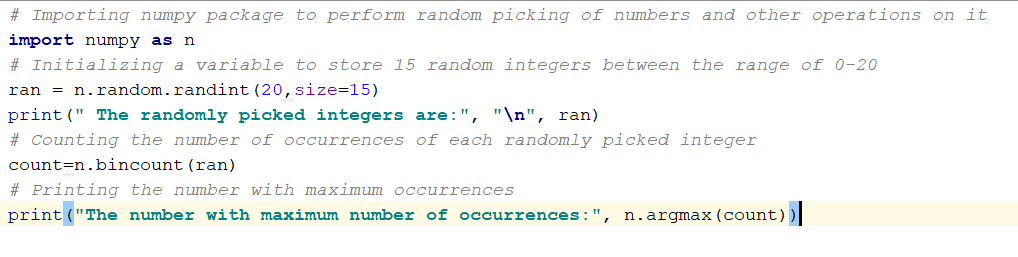


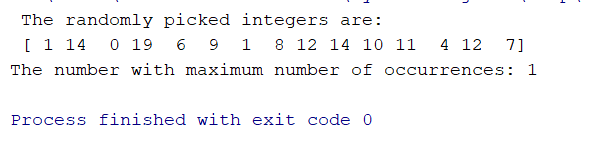
1. Program for depicting various operations on classes:





1. Program on NumPy package:





**Implementation:**

Program 1:

*# Declaring the list with book names as keys and cost of books as values*

bookdictionary = dict([(**"python and deep learning"**,50),(**"Web development"**,60),(**"Design and analysis of algorithms"**,55),(**"Data base management systems"**,65),(**"android development"**,68),(**"Deep Learning"**,70)])

print(**"Now let us have glance at all the list of books in the UMKC book store:"**)

*# Printing all the books in the bookstore and their costs*

**for** key,values **in** bookdictionary.items():  
 print(key,values)

*# Specifying the range of costs that we want to look for:*print(**"Now enter the range of values to find the books in that range"**)

*# Taking the start value as input*startval = int(input(**"enter the starting value:"**))

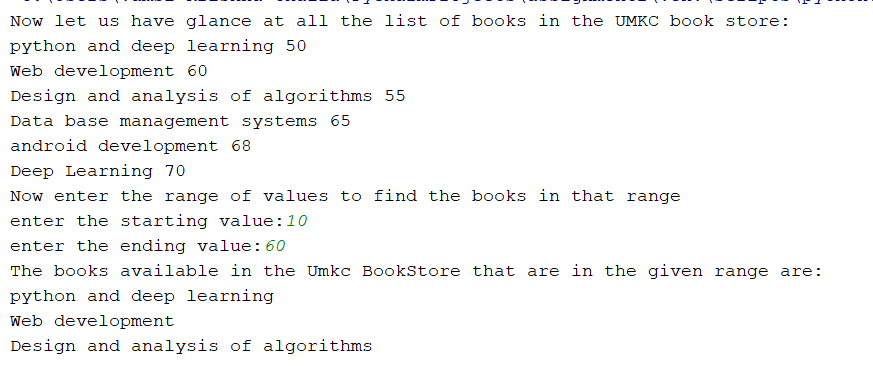
*# Taking the last value as input*lastval = int(input(**"enter the ending value:"**))

*# Searching over the whole list for the books that fall in the given range of costs*print(**"The books available in the Umkc BookStore that are in the given range are:"**)  
**for** bname,value **in** bookdictionary.items():  
 **if** value >= startval **and** value <= lastval:  
 print(bname)

In this program, we intend to check on the list to find out over various books in the book store and take inputs for the range of books that we intend to find out.

Initially we create the list of books and their costs that present in the UMKC book store. Now we take need to perform search operation over the list to find the books that fall under the user defined cost range. We take the user input for the range of costs. Then perform the validation over if-else condition over the for loop to check whether the book falls under particular user defined range or not. If it falls under that range, we print the name of the book.

Result:

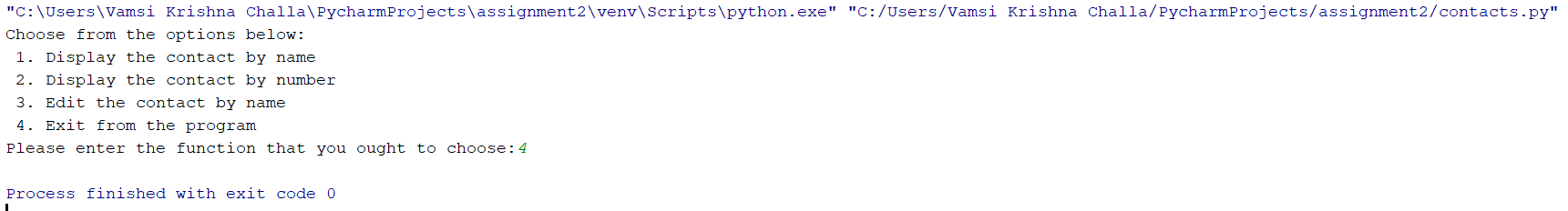
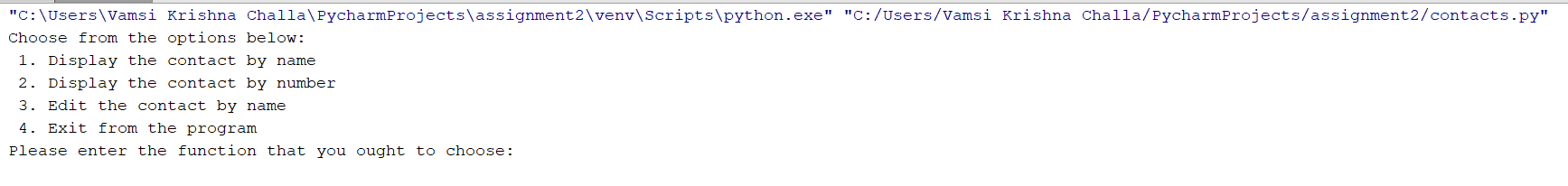
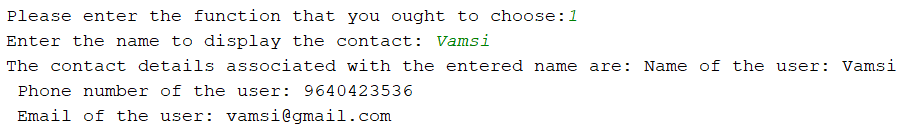
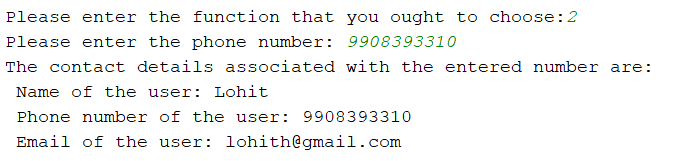
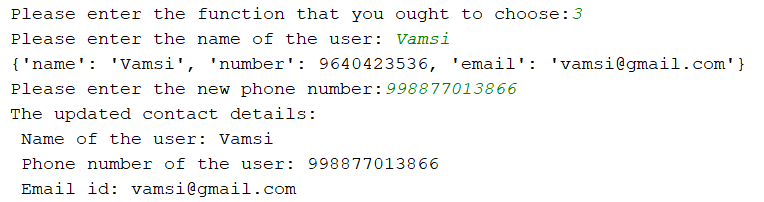


2. Program to perform various operations on LISTS:

contacts = [{**"name"**:**'Vamsi'**, **"number"**:9640423536, **"email"**:**"vamsi@gmail.com"**},{**"name"**:**"Lohit"**, **"number"**:9908393310, **"email"**:**"lohith@gmail.com"**},{**"name"**:**"haravind"**,**"number"**:7979876756,**"email"**:**"haravind@gmail.com"**}]  
  
*# Letting the user choose from the below options to access the list of contacts*print(**"Choose from the options below:"**)  
print(**" 1. Display the contact by name"**)  
print(**" 2. Display the contact by number"**)  
print(**" 3. Edit the contact by name"**)  
print(**" 4. Exit from the program"**)  
var = int(input(**"Please enter the function that you ought to choose:"**))  
  
**if**(var==1):  
 *# Prompting the user to enter the name of the contact to displayed* name = input(**"Enter the name to display the contact: "**)  
 *# Moving over all the contacts to match with the entered name* **for** q **in** contacts:  
 *# To check for the username, whether present in the contacts or not?* **if** name **in** q.values():  
 *# Printing the name, if present in the list* print(**"The contact details associated with the entered name are:"**,**"Name of the user:"**,q[**"name"**], **"\n"**, **"Phone number of the user:"**,q[**"number"**],**"\n"**, **"Email of the user:"**,q[**"email"**])  
 **else**:  
 *# Username not found in the list* print(**"The entered user not found in contacts"**)  
  
*# If user chooses to display the contact by the contact number***elif**(var==2):  
 *# Prompting the user to enter the mobile-number of the user* pnumb = int(input(**"Please enter the phone number: "**))  
 *# Checking over the contacts for the matching phone number in the list* **for** r **in** contacts:  
 *# Checking if the phone number exists in the list:* **if** pnumb **in** r.values():  
 *# Prnting the contact if condition is true* print(**"The contact details associated with the entered number are:"**,**"\n"**,**"Name of the user:"**,r[**"name"**], **"\n"**, **"Phone number of the user:"**,r[**"number"**],**"\n"**, **"Email of the user:"**,r[**"email"**])  
*# If user chooses to edit the contact by name:***elif**(var==3):  
 *# Prompting the user to enter the name of the contact he intends to edit* user = input(**"Please enter the name of the user: "**)  
 *# Iterating over the contact\_list* **for** s **in** contacts:  
 *# checking if the user entered value of name whether exists in the contact or not* **if** user **in** s.values():  
 *# Printing the contact before updating the list* print(s)  
 *# Prompting the user to enter the number that he want to update to* newphone = int(input(**"Please enter the new phone number:"**))  
 *# Editing the nimber for the particular user* s[**"number"**] = newphone  
 *# Printing the contact* print(**"The updated contact details:"**,**"\n"**,**"Name of the user:"**,s[**"name"**], **"\n"**, **"Phone number of the user:"**,s[**"number"**], **"\n"**, **"Email id:"**,s[**"email"**])  
  
**elif**(var==4):  
 *# Exiting the program* exit()

In this program, initially I have specified the list with contacts in the form of a list. Now we specify numbers to each operation that we are asked to perform on the lists. Now we take input from the user and then pass it on to the if-else statements. The execute corresponding function basing on the user choice.

Result:

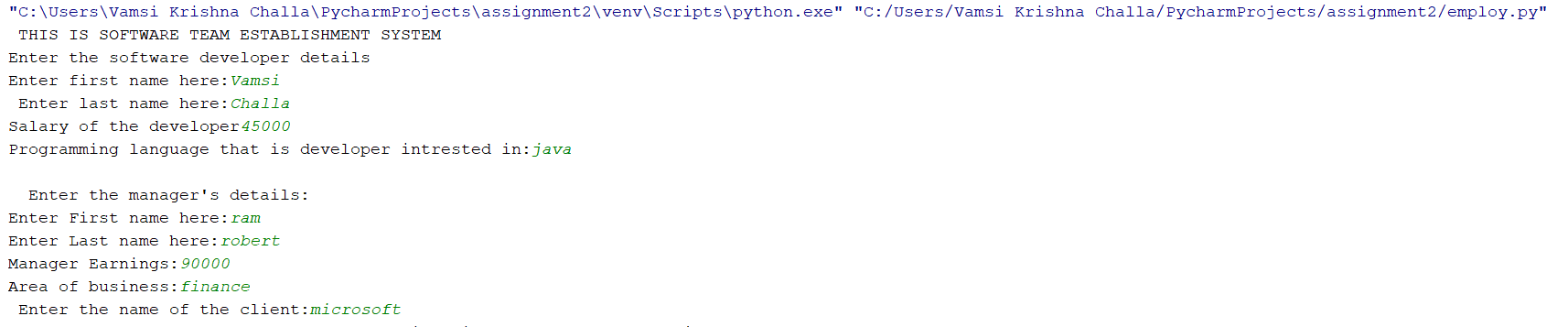


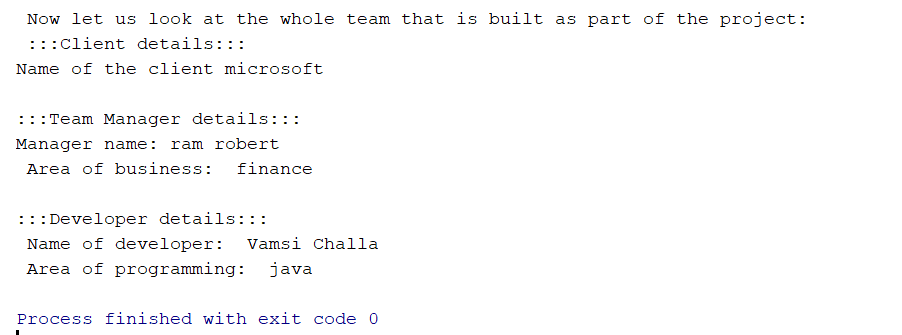
1. Program for performing operations on classes:

*# This is a simple team building system  
# Class 1: Declaring the employee class***class** Employee:  
  
 *# Initializing the employee class and adding both first name, last name and also salary* **def** \_\_init\_\_(self, fname, lname, sal):  
 self.firstname = fname  
 self.lastname = lname  
  
 *# You can see here that the employee salary is made privated* self.\_\_salary = sal  
  
 **def** combinename(self):  
 **return** self.firstname + **' '** + self.lastname  
  
 **def** employesal(self):  
 print(self.combinename(), **" Salary is "**, self.\_\_salary)  
  
*# Class 2: Declaring the class developer  
# Single inheritance is being implemented here. You can see employee class being inherited by developer class.***class** Developer(Employee):  
 *# Initializing the developer class* **def** \_\_init\_\_(self, firname, lastname, sal, language):  
 Employee.\_\_init\_\_(self, firname, lastname, sal)  
 self.programminglanguage=language  
 *# Access the developer's name* **def** getdevop(self):  
 **return** self.firstname + **' '** + self.lastname  
 *# Access programming language of the developer* **def** getprogramminglanguage(self):  
 **return** self.programminglanguage  
  
*#Class 3: Declaring the manager class***class** Manager():  
  
 *#Initializing the manager class* **def** \_\_init\_\_(self, managername, msurname, managersal, bu):  
 self.fn = managername  
 self.ln = msurname  
 self.msal = managersal  
 self.bu=bu  
  
 *# returning the manager name* **def** mangrnme(self):  
 **return** self.fn + **' '** + self.ln  
  
 **def** getbu(self):  
 **return** self.bu  
  
*# Class 4: Declaring the client class***class** Clients():  
  
 *# Initializing the client class using self* **def** \_\_init\_\_(self, cname):  
 self.clientnm= cname  
  
 **def** accessclientname(self):  
 **return** self.clientnm  
  
*# Class 5: Declaring team class and we are using the concept of multiple inheritance***class** Team(Developer, Manager, Clients):  
  
 **def** \_\_init\_\_(self, dfname, dlname, dsalary, dproglang, managername, msurname, managersal, mbu, cname):  
 *# Manager class being instantiated* Manager.\_\_init\_\_(self, managername, msurname, managersal, mbu)  
 *# Client class being instantiated* Clients.\_\_init\_\_(self, cname)  
 *# Developer class being instantiated* Developer.\_\_init\_\_(self,dfname,dlname,dsalary,dproglang) *#instance of developer class*print(**" THIS IS SOFTWARE TEAM ESTABLISHMENT SYSTEM"**)  
print(**"Enter the software developer details"**)  
devopfirst = input(**"Enter first name here:"**)  
devoplast = input(**" Enter last name here:"**)  
devoppay = input(**"Salary of the developer"**)  
devoplanguage = input(**"Programming language that is developer intrested in:"**)  
  
print(**"\n"**,**" Enter the manager's details:"**)  
managerf = input(**"Enter First name here:"**)  
managerl = input(**"Enter Last name here:"**)  
managerpay = input(**"Manager Earnings:"**)  
managerbusiness = input(**"Area of business:"**)  
client = input(**" Enter the name of the client:"**)  
teambuild = Team(devopfirst, devoplast, devoppay, devoplanguage, managerf, managerl, managerpay, managerbusiness, client)*#instance of team class*print(**" Now let us look at the whole team that is built as part of the project:"**)  
print(**" :::Client details:::"**)  
print(**"Name of the client"**, teambuild.accessclientname(), **"\n"**)  
  
print(**":::Team Manager details:::"**)  
print(**"Manager name:"**, teambuild.mangrnme())  
print(**" Area of business: "**, teambuild.getbu(), **"\n"**)  
  
  
print(**":::Developer details:::"**)  
print(**" Name of developer: "**,teambuild.getdevop())  
print(**" Area of programming: "**,teambuild.getprogramminglanguage())

This is simple program about project team establishment system. I have declared five classes for this purpose. I have also implemented multiple inheritance over the class Team by inheriting manager, developer and client classes. I have also declared the salary of developer as private member of the class. This program takes the input from the user and then displays the team that is built for a particular project.

Result:





1. Program to print the number with highest frequency of occurrence among the fifteen numbers that are picked up using random function:

*# Importing numpy package to perform random picking of numbers and other operations on it*

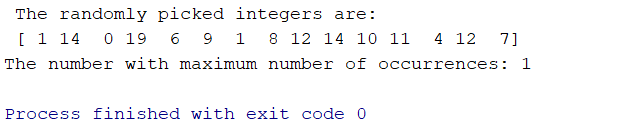
**import** numpy **as** n

*# Initializing a variable to store 15 random integers between the range of 0-20*ran = n.random.randint(20,size=15)  
print(**" The randomly picked integers are:"**, **"\n"**, ran)

*# Counting the number of occurrences of each randomly picked integer*count=n.bincount(ran)

*# Printing the number with maximum occurrences*print(**"The number with maximum number of occurrences:"**, n.argmax(count))

We need to import the NumPy package. By default, NumPy might not be available for the project interpreter to import it. We need to install the package first, then we can import the features of the package into our program. Now take a variable to store the list of randomly generated integers using “random” function and “randint” method that are inbuilt to the NumPy package. Once the numbers are randomly picked, we count the frequency of each number in the list using “bincount” function that comes with NumPy package. Now we use “argmax” function to find out the number with maximum number of occurrences in the list and print it.



**Deployment:**

In this assignment, we have created, manipulated and access both lists and dictionaries, which have a very great influence in python. We have also implemented classes and have performed many operations like accessing and instantiating the objects, inheritance etc. We have also implemented NumPy package which we have used to access features like, random printing of numbers, bin count to count the frequency of numbers, argmax to find the number with maximum frequency.

**Limitations:**

This implementation has several limitations due to program scope. We have restricted some parts to static based retrieval, making it not feasible for new set of values. If we perform dynamic based, then we can perform operations more efficiently.

For instance, in the case of the third part of assignment we have just written the program to take input and deliver the output by substantiating the whole content onto class. But this piece of code cannot be implemented directly to real-time environment. If this needs to be connected to web application, we need to implement data base connections and allocate dedicate memory for it to work.

In the first part of assignment we have seen that, we implement only static list of books. For a book store, dynamic list is to be implemented. Dynamic list would facilitate easy adding of new content to the data base and can be made easy in maintain the number of books in the book store.

In the second part of assignment, I have implemented contacts list statically. This can be connected to a web application, which would be of a great use in handling the user data. Due to lack of database resource, we skip were unable implement onto a web application.

References:

* [www.stackoverflow.com](http://www.stackoverflow.com)
* [www.geekforgeeks.com](http://www.geekforgeeks.com)
* [www.tutorialspoint.com](http://www.tutorialspoint.com)
* [www.python.org](http://www.python.org)