CSEE-5590 - Special Topics  
 Python – Lab 1

**Author:**   
Kilaru Kalyan  
Class Id : 22  
Student ID: 16245031

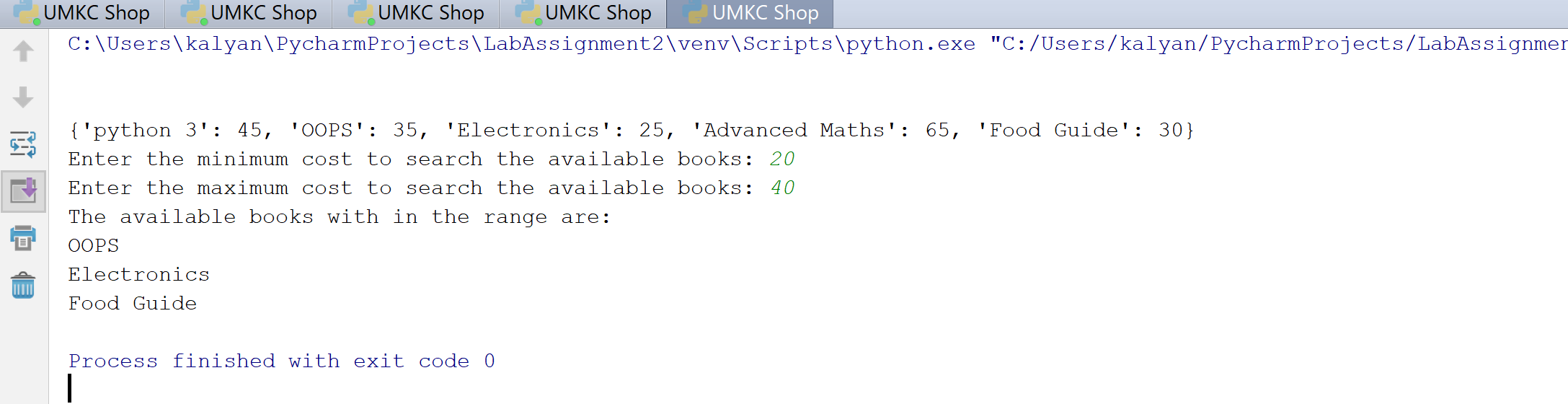
**Configuration:**  
IDE : pycharm Community Edition  
python : version 3.6.4

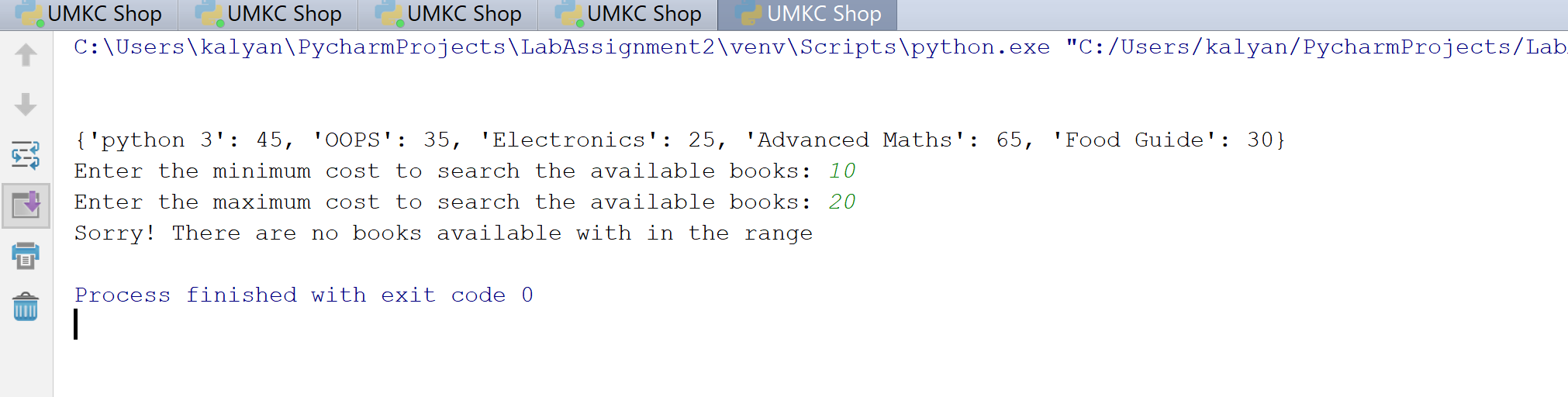
**Objective 1:** To write a program to get all the books that are in the specific price range,where the dictionary is used to store the values of books and its price.

**Implementation:**  
First, a dictionary with key value pairs is taken and then we ask the user to enter the minimum and the maximum cost to search for the books that can be affordable. By using the for loop we pass through all the items. We use the if conditional statement to check when the price is between the maximum and the minimum and then we print the books list that are with in the range given by the user.

**Code:**

*# The books dictionary with book name and the cost of the book*books = {**"python 3"**:45,**"OOPS"**:35,**"Electronics"**:25,**"Advanced Maths"**:65,**"Food Guide"**:30}  
print(**"\n"**)  
print(books) *# For printing the books  
  
# To get the range from the user*minimum\_cost = int(input(**"Enter the minimum cost to search the available books: "**))  
maximum\_cost = int(input(**"Enter the maximum cost to search the available books: "**))  
  
print(**"The available books with in the range are: "**)  
*# The available books within the range are displayed.*count = 0  
**for** key **in** books:  
 **if** books[key] >= minimum\_cost **and** books[key] <= maximum\_cost:  
 count = 1  
 print(key)  
**if** count == 0:  
 print(**"Sorry! There are no books available with in the range"**)

**Input/Output:**

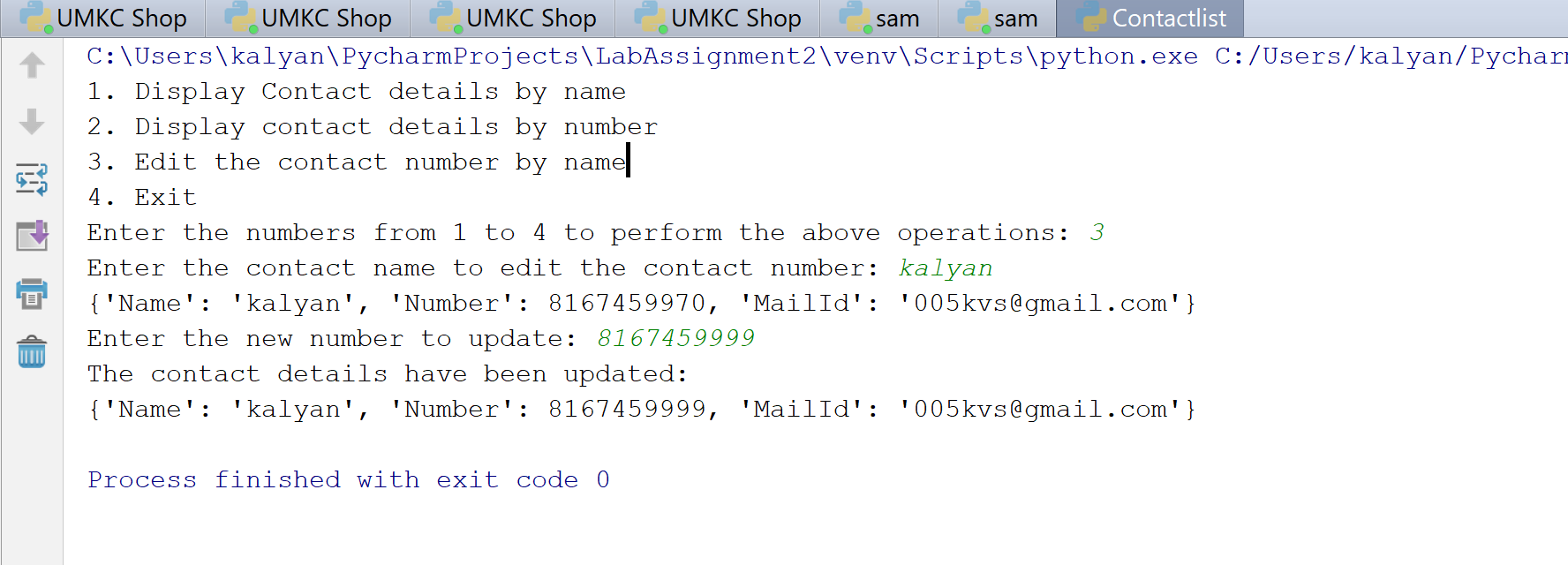
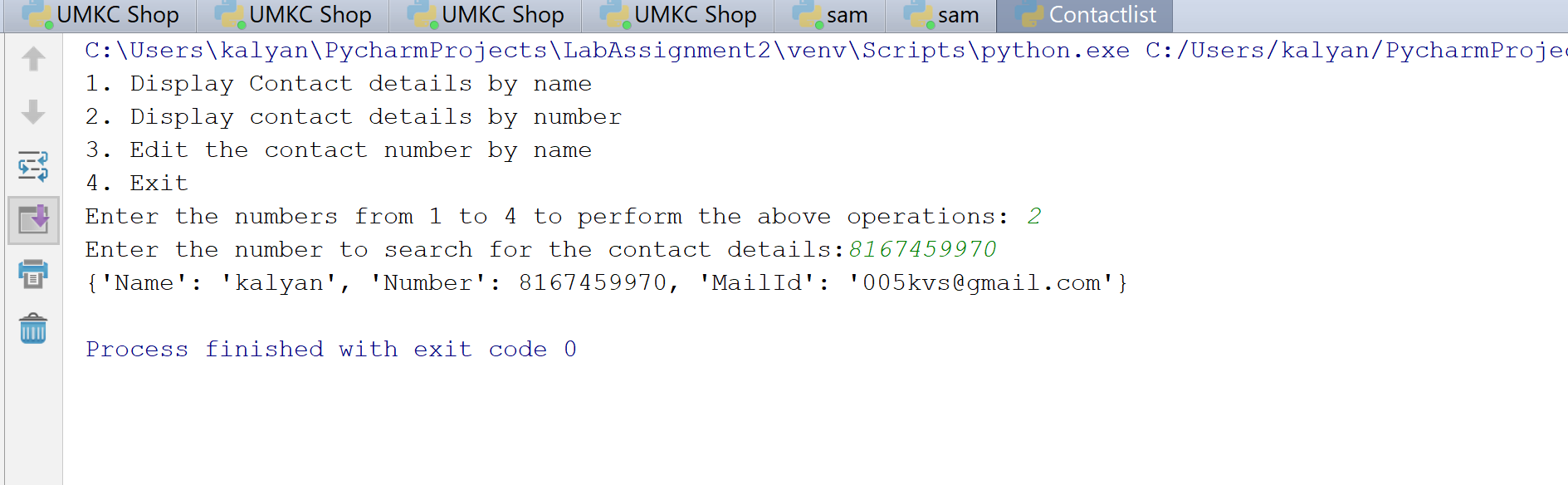
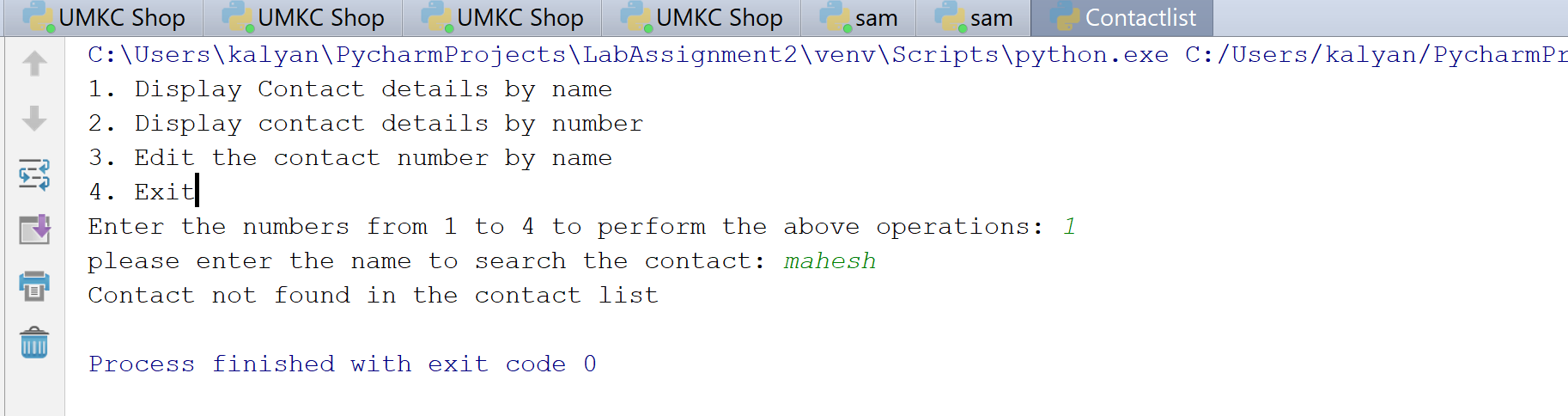
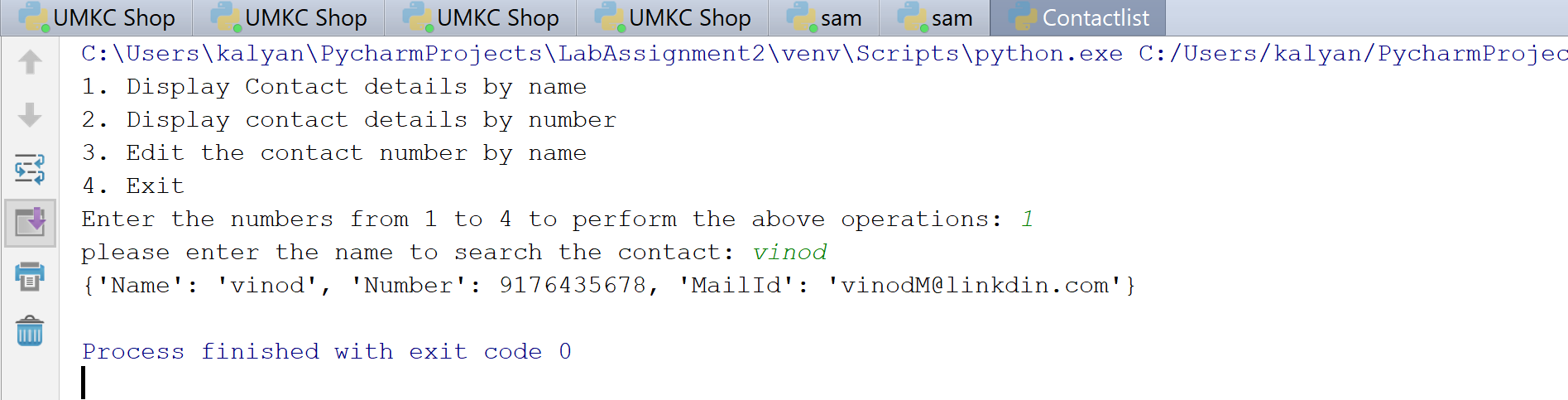


**Objective 2:** To perform different operations like displaying the contact details or the dictionary details based on one of the input and to edit the values in the dictionary.  
  
**Implementation:**First, we will display the operations that we are going to perform so that the user selects from it. Then we use the if conditions for different operations based on the user input. And in each operation, we go through a for loop with an if condition inside it. The for loop iterates through the dictionary sets and then the if loop will search for the value we are looking for. If the value is not found then we are going to print that the value is not found. Similarly, we will edit by giving the new value to the dictionary values and then display the updated contact list.

**Code:**

*# Function to perforn the operations***def** contacts\_list():  
 contacts = [{**"Name"**: **"kalyan"**, **"Number"**: 8167459970, **"MailId"**: **"005kvs@gmail.com"**}, *# Contact list* {**"Name"**: **"chaitanya"**, **"Number"**: 8161234569, **"MailId"**: **"cpk@yahoo.com"**},  
 {**"Name"**: **"vinod"**, **"Number"**: 9176435678, **"MailId"**: **"vinodM@linkdin.com"**}]  
  
 print(**"1. Display Contact details by name"**) *# To print the details of the operations performed* print(**"2. Display contact details by number"**)  
 print(**"3. Edit the contact number by name"**)  
 print(**"4. Exit"**)  
 operation = int(input(**"Enter the numbers from 1 to 4 to perform the above operations: "**)) *# user selected operation* search = 0  
 *# To search for a name and display the contact details* **if** operation == 1:  
 search\_name = (input(**"please enter the name to search the contact: "**))  
 **for** contact **in** contacts: *# For loop to retrieve through the dictionary* **if** search\_name **in** contact.values():  
 search = 1  
 print(contact)  
 **break  
 if** search == 0:  
 print(**"Contact not found in the contact list"**)  
 exit(0)  
  
 *# To search for a number and get the contact details* **elif** operation == 2:  
 search\_number = int(input(**"Enter the number to search for the contact details:"**))  
 **for** contact **in** contacts: *# For loop to retrieve through the dictionary* **if** contact[**"Number"**] == search\_number:  
 search = 1  
 print(contact)  
 **break  
 if** search == 0:  
 print(**"Contact not found in the contact list"**)  
 exit(0)  
  
 *# To search for the name and edit the number* **elif** operation == 3:  
 edit\_name = str(input(**"Enter the contact name to edit the contact number: "**))  
 **for** contact **in** contacts: *# For loop to retrieve through the dictionary* **if** contact[**"Name"**] == edit\_name:  
 search = 1  
 print(contact)  
 edit\_number = int (input(**"Enter the new number to update: "**))  
 contact[**"Number"**] = edit\_number *# Give the new number* print(**"The contact details have been updated: "**)  
 print(contact)  
 **break  
 if** search == 0:  
 print(**"Contact name not found on the contact list"**)  
 exit(0)  
 *# If the user want to exit* **elif** operation == 4:  
 print(**"Thanks for using the application."**)  
 exit(0)  
  
 **else**: *# If the user entered a wrong operation* print(**"Wrong input! Run it again :)"**)  
  
  
contacts\_list()

**Input/Output:**

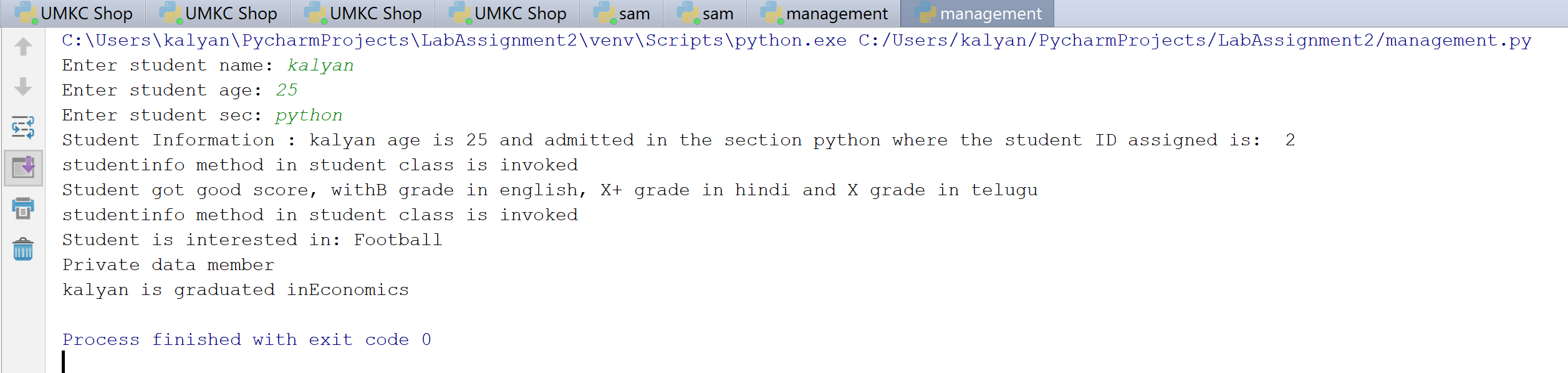


**Objective 3:** We will be creating a application based on the inputs given. Here we use all the object-oriented concepts in making this system. Inheritance, constructors, global variables, super method and any topics are covered in the process of writing the code.  
  
**Implementation:**First a class is created then five different classes are created, where two of these classes inherit the System and the Student classes. The constructor is declared in each of the classes so that we can initialize the values. The classes that are inherited will also initialize its values and then the calls the super method before that, which call the constructor in the super class. We also used self for all the methods. Later, we created the instance of the classes created and passes the values to them through arguments.

**Code:**

**class** System(object):  
  
 *# student id is initially assigned to 1 and is incremented when ever a new student is added* id\_student = 1  
  
 *# The initialize constructor is defined where the initial values are assigned* **def** \_\_init\_\_(self, name, age, sec):  
 System.id\_student += 1  
 self.name = name *# initializing name* self.age = age *# initializing age of a student* self.sec = sec *# initializing sec of a student  
  
 # student info function is used to display the details of the students and give the student an id* **def** studentinfo(self):  
 print(**"Student Information : "** + self.name + **" age is "** + str(  
 self.age) + **" and admitted in the section "** + self.sec + **" where the student ID assigned is: "** + str(System.id\_student))  
  
 *# The below is the student class which inherits the system class***class** Student(System):  
  
 **def** \_\_init\_\_(self, name, age, sec):  
 *# The super method calls the parent class constructor* super(Student, self).\_\_init\_\_(name, age, sec)  
  
 **def** studentinfo(self):  
 print(**"studentinfo method in student class is invoked"**)  
  
 *# This medthod prints the class name* **def** classidentity(self):  
 print(**"Student Class method"**)  
  
 *# Student sport method tells you which sport student plays* **def** studentsport(self):  
 print(**"Student is interested in: "** + self.game)  
  
 *# Only the student object can have an access to this private member* **def** \_\_idnum(self):  
 print(**"Private data member"**)  
  
  
 *# Student is parent class which is inherited by the grades class***class** Grades(Student):  
 *# The constructor which takes various courses as input and initialize them* **def** \_\_init\_\_(self, english, hindi, telugu):  
 self.english = english  
 self.hindi = hindi  
 self.telugu = telugu  
  
 *# This method displays the scpre of the students in each course* **def** score(self):  
 print(  
 **"Student got good score, with"** + self.english + **" grade in english, "** + self.hindi + **" grade in hindi and "** + self.telugu + **" grade in telugu "**)  
  
 **def** studentinfo(self):  
 print(**"Grade class"**)  
  
 *# The final score of the student is defined here* **def** finalscore(self):  
 print(**"Student graduated with good grade"**)  
  
  
*# Class enroll that extends the student and system class. so that the subclass can access the parents class methods***class** Enroll(Student, System):  
   
 *# the constructor that takes the initial values* **def** \_\_init\_\_(self, name, age, sec):  
 *# Super which call the parent constructor* super(Enroll, self).\_\_init\_\_(name, age, sec)  
  
  
*# studentsports class extends the student class***class** studentsports(Student):  
  
 **def** \_\_init\_\_(self, game):  
 self.game = game  
  
  
*# Class grade that inherits different classes***class** grad(Student, System):  
 *# The initialize constructor* **def** \_\_init\_\_(self, name, degree):  
 self.degree = degree  
 self.name = name  
  
 **def** studentdetails(self):  
 print(self.name + **" is graduated in"** + self.degree)  
  
  
*# object creation*name = input(**"Enter student name: "**)  
age = int(input(**"Enter student age: "**))  
sec = input(**"Enter student sec: "**)  
  
systema = System(name, age, sec)  
  
systema.studentinfo()  
  
sa = Student(name, age, sec)  
  
sa.studentinfo()  
sb = Student(name, age, sec)  
sc = Student(name, age, sec)  
  
gradea = Grades(**'B'**, **'X+'**, **'X'**)  
  
gradea.score()  
enrolla = Enroll(**'kalyan'**, 26, **'Fresher'**)  
enrolla.studentinfo()  
sportsa = studentsports(**'Football'**)  
sportsa.studentsport()  
sa.\_Student\_\_idnum()  
gs = grad(**'kalyan'**, **'Economics'**)  
gs.studentdetails()

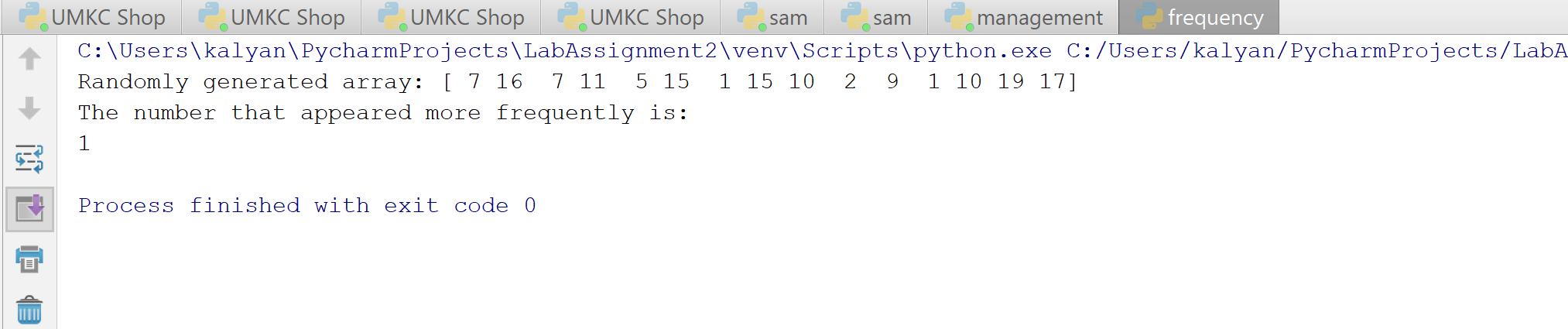
**Input/Output:**

****

**Objective 4:** To get a random list of 15 numbers ranging from 0 to 20 and then to find the number is more frequent in the list.   
  
**Implementation:**we imported the numpy module and then by using this we generated the list that consists of 15 numbers. The range of these numbers are from 0 to 20. Then by using the bincount method we will get the count of each number. Then we print the number that has maximum number of occurrences.

**Code:**  
*# We are importing the numpy as nmp***import** numpy **as** nmp  
  
*# We are initializing the vector with 15 random numbers from 0 to 20*vector = nmp.random.randint(0,20, size=15)  
*# To print the generated array*print(**"Randomly generated array:"**,end=**" "**)  
print(vector) *# to print the random array*frequent\_number = nmp.bincount(vector) *# We will get the count of each random number*print(**"The number that appeared more frequently is: "**) *# The random number which occurs maximum times*print(nmp.argmax(frequent\_number)) *# To print the more frequent random number*

**Input/Otput:**

****

**Deployment:**

The code is written in python in pycharm IDE  
Imported the numpy module for the objective-4  
Ran the code in IDE and the outputs are checked in console

**Limitations:**Code is not robust  
Inputs are given manually  
Validations are not taken care for inputs in dictionary values.

**References:**https://www.python-courses.eu/dictionaries.php