1 Author

Name: Rohit Abhishek
Student Id: 16158762

2 Objective

The objective is to get familiar with the concepts of dictionary, classes .etc basic python functionalities by solving the lab assignment. For this assignment I have implemented dictionaries, lists, classes, loops, numpy and other basic python function.

3 Features

The features of the each problem is

- Problem 1: This program would display all the books based on the price range input by the user. It would also check whether the input range is valid or not. I have added addition functionality by giving user the option of either using an existing dictionary of prices and books or creating a new one.
- Problem 2: This program would display the contact by name or number, edit contact by name or exit. Additional functionality has been added to the program by giving user option of either using an existing contact list or creating a new one. Also the program can check for duplicate names and do a second level of authentication incase there are multiple contacts with the same name.
- Problem 3: This program creates 7 classes for a conference management and has private data member, shows class inheritance, multiple inheritance and class instance.
- Problem 4: This program randomly generates a vector of size 15. Then it returns the most frequent item. Additional functionality has been added so that it returns multiple items.

In addition, flowcharts, algorithm, codes and output has been presented for a better understanding.

4 Configuration

Each program was implement in PyCharm which was installed on MAC Os with 2.5 GHz Intel Core i5 processor, 16 GB 1600 MHz DDR3 memory and storage of 500 GB.

5 Input/Output and code implementation

This section presents the code implementation. Each block of the code has been explained and the input/output has been shown.

5.1 Problem 1

Consider a shop in 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.

5.1.1 Approach

In this question, we need the user to enter the range of the price and display all the books whose prices fall within the range. Since the question does not specify whether the books and the prices are already stored or not, I am assuming both the scenarios. So, there will already be a stored dictionary. The user would be given an option to either create a new dictionary or create a new one.

The function ***create_dictionary*** will be used to create new dictionary. It will ask how many items the user wants in the dictionary and based on the values entered by the user, it would run a loop to ask the user the item name and its price. The code for the same is shown below.

```
def create_dictionary(): #function to create the dictionary
   umkc_dictionary={}
   total_books= int(input("how many items you want to add ")) #ask user how many item
       they want
   for i in range(total_books):
      book_name=input("Enter the name of %d item "%(i+1))
      book_price=int(input("Enter the price of %s "%book_name))
      umkc_dictionary[book_name]=book_price
   return umkc_dictionary
```

The function ***find_books*** will be used to find the books in the given range. It would ask the user for a lower and higher price range. It would also verify that lower range is less than equal to the higher range. In case, there is an error, it would as the user to enter the value again or exit. Once the range has been given, it would check for the books in the price range by iterating the dictionary. The code for the same is provided below.

```
def find_books(umkc_dictionary): #function to find the books within the range
   lower_range=int(input("Enter the lower range ")) #lower range
   higher_range=int(input("Enter higher range")) #higher range
   if lower_range>higher_range: #if lower range is greater than higher ranger
      user_retry=int(input("Range not valid (lower range is less than higher range) \n"
          #ask if user wants to retry
            "Press 1 to retry \n"
            "Press any other key to exit \n"))
      if user_retry==1:
         find_books(umkc_dictionary) #if user wants to retry, call the function again
      else:
         exit()

   for books,price in umkc_dictionary.items(): #loop to iterate the dictionary to find
```

```
      the books in the range
    if lower_range<= price <=higher_range:
      print(books)
```

Algorithm 1: Algorithm for Problem 1

Result: Found books within a given range
Define function to create dictionary;
Define function to find books in dictionary;
Define a dictionary with existing values;
Ask If user wants to use an existing value or create a new one ;
if *Use exiting dictionary* **then**
  Call function find_books;
  Ask lower range;
  Ask higher range;
else
  Create a new dictionary;
  Call find_books;
  Ask lower range;
  Ask higher range;
end

5.1.2 Code

This section presents the code along with comments to understand each block of the code.

```
def create_dictionary(): #function to create the dictionary
   umkc_dictionary={}
   total_books= int(input("how many items you want to add ")) #ask user how many item
       they want
   for i in range(total_books):
      book_name=input("Enter the name of %d item "%(i+1))
      book_price=int(input("Enter the price of %s "%book_name))
      umkc_dictionary[book_name]=book_price
   return umkc_dictionary


def find_books(umkc_dictionary): #function to find the books within the range
   lower_range=int(input("Enter the lower range ")) #lower range
   higher_range=int(input("Enter higher range")) #higher range
   if lower_range>higher_range: #if lower range is greater than higher ranger
      user_retry=int(input("Range not valid (lower range is less than higher range) \n"
          #ask if user wants to retry
            "Press 1 to retry \n"
            "Press any other key to exit \n"))
      if user_retry==1:
         find_books(umkc_dictionary) #if user wants to retry, call the function again
      else:
```

```
        exit()

   for books,price in umkc_dictionary.items(): #loop to iterate the dictionary to find
       the books in the range
      if lower_range<= price <=higher_range:
        print(books)




umkc_dictionary= {"python":50, "web": 30, "c":20,"java":40} #Existing dictionary
user_option= int(input("Would you like to use existing dictionary or create a new one\n"
     #ask if user wants to use an existing dictionary or create new dictionary
                 "1-Press 1 to use existing dictionary\n"
                 "2-Press 2 create new one \n"
                 "3-Press any other key to exit\n"))
if user_option==1:
   find_books(umkc_dictionary)
elif user_option==2:
   umkc_dictionary=create_dictionary()
   find_books(umkc_dictionary)
else:
   exit()
```

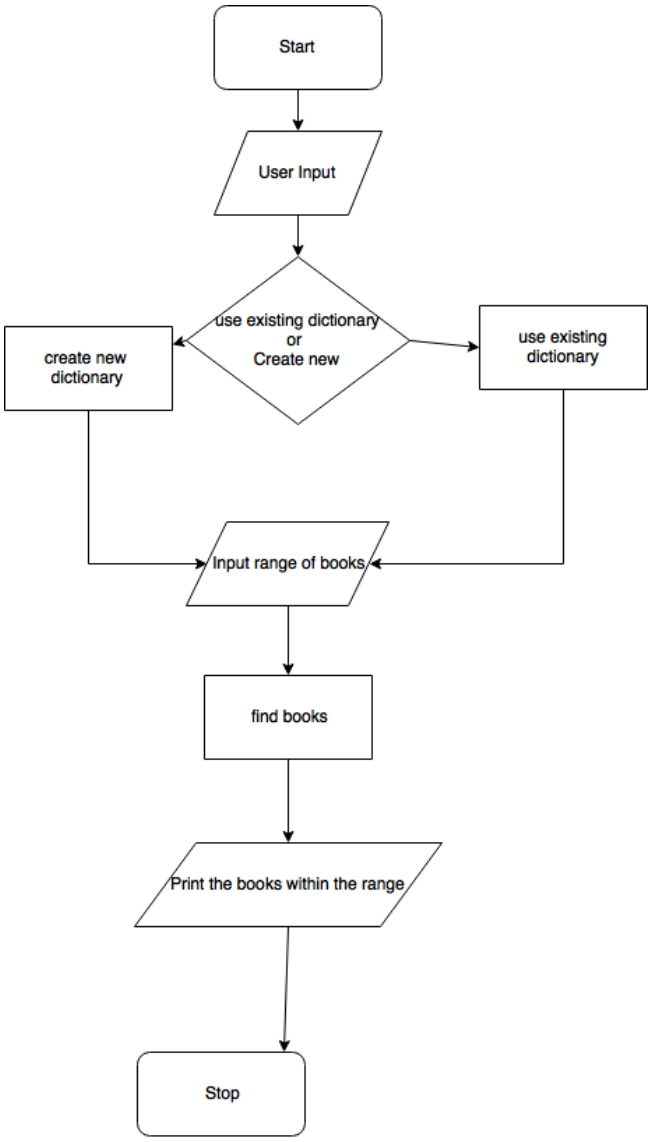


Figure 1: Problem 1:Flowchart

```
C:\Users\rabhishe\PycharmProjects\untitled3\venv1\Scripts\python.exe C:
Would you like to use existing dictionary or create a new one
1-Press 1 to use existing dictionary
2-Press 2 create new one
3-Press any other key to exit
1
Enter the lower range 10
Enter higher range30
web
c
```

```
C:\Users\rabhishe\PycharmProjects\untitled3\venv1\Scripts\pyth
Would you like to use existing dictionary or create a new one
1-Press 1 to use existing dictionary
2-Press 2 create new one
3-Press any other key to exit
2
how many items you want to add 6
Enter the name of 1 item python
Enter the price of python 10
Enter the name of 2 item c
Enter the price of c 20
Enter the name of 3 item java
Enter the price of java 30
Enter the name of 4 item routing
Enter the price of routing 40
Enter the name of 5 item verilog
Enter the price of verilog 50
Enter the name of 6 item oracle
Enter the price of oracle 60
Enter the lower range 30
Enter higher range50
java
routing
verilog
```

Figure 2: Problem 1:Output

5.2 Problem 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

5.2.1 Approach

In this question we need to create a program which would a)display contact by name b) display contact by number c)edit contact by name d)exit ; based on the option selected by the user. So for this we need to create 9 functions. Fist the user would be asked if he wants to create a new contact list or use an existing one. Then the user would be asked what does he want to do. Based on the option selected by the user, the particular function would be simulated.

Function ***display_ contact_ name*** will be used to display the contact names. The code is shown below

```
def display_contact_name(contact_list): #function to display the contact name
    for i in contact_list:
        print(i["name"], ", ", end=" ")
    print("\n")
```

Function ***display_ contact_ number*** would display the contact number along with the names. This function would display contact numbers and names only for those contacts whose number exists.

```
def display_contact_number(contact_list):#function to display the contact number
    for j in contact_list: # this won't display the contacts if there is no number
        assigned to the code
        if (j["number"]!=None) and (j["number"]!=""):
            print(j["name"], " ",j["number"])
```

Function ***edit_ contact_ by_ name*** will edit the contact by name. First it would call the function *number_ of_ contacts_ with_ same_ name* to check how many contact are there with the same name. If there are no contact with the name, it would output "Contact not found" and would ask if the user wants to try again. Also if it sees there are more than one contact with the same name, it would ask the user to enter either a mobile number or email address to go through second level of verification. The it would call function *edit_ details_ by_ mobile_ email* to ask edit the contact using mobile number or email. The code for the function is shown below

```
def edit_contact_by_name(contact_list): #function to edit contact by name
    contact_name=input("Enter the contact name to edit ")
    number_of_contacts_with_same_name=total_contacts_with_same_name(contact_name,
        contact_list)# we need to see if there are multiple contacts with the same name,
```

```
        if yes we need to enter other details to edit
    if number_of_contacts_with_same_name==0: #check if the enter name is present in the
        list or not
        print("Contact Not Found")
    elif number_of_contacts_with_same_name == 1: #unique contact found
        edit_details(contact_list, contact_name) #call edit_details function to edit the
            contact
        continue_again=input("Do you want to edit more contacts (Y)? ").lower() #ask if
            the user wants to continue
        if continue_again=="y":
            edit_contact_by_name(contact_list) #if yes iterate throught the same function
        else:
            return
    else:
        print("%d contacts with the name: %s"%(number_of_contacts_with_same_name,
            contact_name )) #if we multiple contacts with the same name
        second_input=input("Please input either mobile number or email address of the
            contact") #Enter secondary information, either mobile number or email address
        edit_details_by_mobile_email(contact_list, second_input) #call function to edit
            details using mobile or email
```

Function ***edit_details_by_mobile_email***
This function will be called if there are more than one contact with the same name. The user would be asked to enter either mobile number or email address. They do not need to specify whether it is mobile or email. The function would automatically detect it. At the same time this function would also check if the enter email is valid or not.

```
def edit_details_by_mobile_email(contact_list,second_input): #function to edit details
    using number or email
    contact_list=contact_list
    for i in contact_list:
        if i.get("number")==second_input or i.get("email")==second_input:
            edit_details = input("What contact details do you want to edit (Name/Number/
                email) ? ").lower()
            if edit_details == "name":
                new_name = input("Enter new name: ")
                i["name"] = new_name

            elif edit_details == "number":
                new_number = input("Enter new number: ")
                i["number"] = new_number

            elif edit_details == "email": # we need to check if the email address is
                valid
                new_email = input("Enter new email: ")
                email_exits = verify_email(new_email)
                while email_exits == False:
                    new_email = input("Enter correct email addres: ")
                    verify_email(new_email)
                    email_exits = verify_email(new_email)
                i["email"] = new_email
```

```
            else:
                print("Not a Valid choice: ")
                edit_contact_by_name()
        else:
            print("Details not found. Please try again")
```

Function ***edit_details***

This function would edit the details based on the name. So this function would be called only if unique user is present. It would ask the user what details needs to be edited and base on the input it would edit the details. This function would also check for the validity of the email address.

```
def edit_details(contact_list,contact_name):
    for i in contact_list:
        if i.get("name") == contact_name:
            edit_details = input("What contact details do you want to edit (Name/Number/
                email) ? ").lower()
            if edit_details == "name":
                new_name = input("Enter new name: ")
                i["name"] = new_name

            elif edit_details == "number":
                new_number = input("Enter new number: ")
                i["number"] = new_number

            elif edit_details == "email": # we need to check if the email address is
                valid
                new_email = input("Enter new email: ")
                email_exits = verify_email(new_email)
                while email_exits == False:
                    new_email = input("Enter correct email addres: ")
                    verify_email(new_email)
                    email_exits = verify_email(new_email)
                i["email"] = new_email
            else:
                print("Not a Valid choice: ")
                edit_contact_by_name()
```

Function ***total_contacts_with_same_name***

This function is used to see how many contacts are present with the same name.

```
def total_contacts_with_same_name(contact_name,contact_list):
    counter=0
    for i in contact_list:
        n = i.get("name")
        for j in n.split():
           # print(j, end=" ")
            if j == contact_name:
                counter = counter + 1
    return counter
```

Function ***verify_email***

This function is used to check the validity of the input email address. It will make use of module *validate_email* to check the validity.

```
def verify_email(new_email):
    return validate_email(new_email)
```

Function ***main***

This is main function which would ask if the user wants to create a new contact list or use an existing one. Then based on the user's input, it would call different corresponding functions. This function would also create a list of contact incase the user chooses to create one.

```
def main(contact_list):
    list_use= input("Do you want to use existing list or create a new list (press 1 to
        use existing list and 2 to create new list ? ")
    if list_use == "1":
        user_choice(contact_list)

    if list_use=="2":
        contact_list = []
        number_of_contacts = int(input("how many contact do you want to add? "))
        for i in range(number_of_contacts):
            name = input("Enter the full name: ")
            number = int(input("Enter the number: "))
            email = input("Enter the email address: ")
            while validate_email(email) == False:
                print("email not valid")
                email = input("Enter the email address again: ")
            contact_list.append({'name': name, 'number': number, 'email': email})
        user_choice(contact_list)
```

Function ***user_choice***

This function would be used to ask the user their choice and then based on the choice, it would call different corresponding functions.

```
def user_choice(contact_list):
    user_option = input("Press 1 to display contact name\n"
                        "Press 2 to display contact number\n"
                        "Press 3 to edit contact\n"
                        "Press 4 to exit\n")
    if user_option == '1':
        print("Contact Names")
        display_contact_name(contact_list)
    if user_option == '2':
        print("Contact Numbers")
        display_contact_number(contact_list)
    if user_option == '3':
        edit_contact_by_name(contact_list)
    if user_option == '4':
        exit()
    print("Updated contact list")
    print(contact_list)
    print("\n")
```

```
    contact_list = contact_list
    continue_option=input("Do you want to continue (Y to continue/ Any key to exit )? ")
    if (continue_option) == "y" or (continue_option) == "Y" :
        print(contact_list)
        user_choice(contact_list)
    return contact_list
```

Algorithm 2: Algorithm for Problem 2

Result: Contact displayed by name,Contacts displayed by number, Contacts edited, exit
Define function to display contact name;
Define function to display contact number;
Define function to edit contact by name;
Define function to edit contact by email or mobile;
Define function to find multiple contacts with the same name;
Define function to verify email;
Define function to ask user choice;
if $user\_input == displaycontactname$ **then**
  $callfunctiondisplay\_contact\_name$;
end
if $user\_input == displaycontactnumber$ **then**
  $callfunctiondisplay\_contact\_number$;
end
if $user\_input == editcontactbyname$ **then**
  $callfunctiondisplay\_contact\_by\_namer$;
end
if $user\_input == exit$ **then**
  $exit$;
end

5.2.2 Code

This section presents the code along with comments to understand each block of the code.

```
from validate_email import validate_email #module to check the validity of email address


def display_contact_name(contact_list): #function to display the contact name
    for i in contact_list:
        print(i["name"], ", ", end=" ")
    print("\n")

def display_contact_number(contact_list):#function to display the contact number
    for j in contact_list: # this won't display the contacts if there is no number
        assigned to the code
        if (j["number"]!=None) and (j["number"]!=""):
            print(j["name"], " ",j["number"])
```

```
def edit_contact_by_name(contact_list): #function to edit contact by name
    contact_name=input("Enter the contact name to edit ")
    number_of_contacts_with_same_name=total_contacts_with_same_name(contact_name,
        contact_list)# we need to see if there are multiple contacts with the same name,
         if yes we need to enter other details to edit
    if number_of_contacts_with_same_name==0: #check if the enter name is present in the
        list or not
        print("Contact Not Found")
    elif number_of_contacts_with_same_name == 1: #unique contact found
        edit_details(contact_list, contact_name) #call edit_details function to edit the
             contact
        continue_again=input("Do you want to edit more contacts (Y)? ").lower() #ask if
            the user wants to continue
        if continue_again=="y":
            edit_contact_by_name(contact_list) #if yes iterate throught the same function
        else:
            return
    else:
        print("%d contacts with the name: %s"%(number_of_contacts_with_same_name,
            contact_name )) #if we multiple contacts with the same name
        second_input=input("Please input either mobile number or email address of the
            contact") #Enter secondary information, either mobile number or email address
        edit_details_by_mobile_email(contact_list, second_input) #call function to edit
            details using mobile or email


def edit_details_by_mobile_email(contact_list,second_input): #function to edit details
    using number or email
    contact_list=contact_list
    for i in contact_list:
        if i.get("number")==second_input or i.get("email")==second_input:
            edit_details = input("What contact details do you want to edit (Name/Number/
                email) ? ").lower()
            if edit_details == "name":
                new_name = input("Enter new name: ")
                i["name"] = new_name

            elif edit_details == "number":
                new_number = input("Enter new number: ")
                i["number"] = new_number

            elif edit_details == "email": # we need to check if the email address is
                valid
                new_email = input("Enter new email: ")
                email_exits = verify_email(new_email)
                while email_exits == False:
                    new_email = input("Enter correct email addres: ")
                    verify_email(new_email)
                    email_exits = verify_email(new_email)
                i["email"] = new_email
            else:
                print("Not a Valid choice: ")
                edit_contact_by_name()
```

```
def edit_details(contact_list,contact_name): #this function would edit the contact name,
     mobile or email
    for i in contact_list:
        if i.get("name") == contact_name: #if contact name found
            edit_details = input("What contact details do you want to edit (Name/Number/
                email) ? ").lower() #ask user what to edit
            if edit_details == "name": #edit name
                new_name = input("Enter new name: ")
                i["name"] = new_name

            elif edit_details == "number": #edit number
                new_number = input("Enter new number: ")
                i["number"] = new_number

            elif edit_details == "email": #edit email
                new_email = input("Enter new email: ")
                email_exits = verify_email(new_email)
                while email_exits == False:
                    new_email = input("Enter correct email addres: ")
                    verify_email(new_email)# we need to check if the email address is
                        valid
                    email_exits = verify_email(new_email)
                i["email"] = new_email
            else:
                print("Not a Valid choice: ")
                edit_contact_by_name()


def total_contacts_with_same_name(contact_name,contact_list): #function to find contact
    with the same name
    counter=0
    for i in contact_list: # loop to see how name contacts are present with the same
        name
        n = i.get("name")
        for j in n.split():
           # print(j, end=" ")
            if j == contact_name:
                counter = counter + 1
    return counter


def verify_email(new_email): #function to verify the email
    return validate_email(new_email)


def main(contact_list): #main function
    list_use= input("Do you want to use existing dictionary or create a new dictionary (
        press 1 to use existing list and 2 to create new list ? ") #ask if user wants to
         use existing list or create a new one
    if list_use == "1": #use existing list
        user_choice(contact_list)
```

```
    if list_use=="2": #create new dictionary
        contact_list = []
        number_of_contacts = int(input("how many contact do you want to add? "))
        for i in range(number_of_contacts):
            name = input("Enter the full name: ")
            number = int(input("Enter the number: "))
            email = input("Enter the email address: ")
            while validate_email(email) == False: #check for validity of email
                print("email not valid")
                email = input("Enter the email address again: ")
            contact_list.append({'name': name, 'number': number, 'email': email})
        user_choice(contact_list)



def user_choice(contact_list): #function to ask user choices
    user_option = input("Press 1 to display contact name\n"
                        "Press 2 to display contact number\n"
                        "Press 3 to edit contact\n"
                        "Press 4 to exit\n")
    if user_option == '1': #to display contact name
        print("Contact Names")
        display_contact_name(contact_list)
    if user_option == '2': #to display contact numbers
        print("Contact Numbers")
        display_contact_number(contact_list)
    if user_option == '3': #to edit contact
        edit_contact_by_name(contact_list)
    if user_option == '4': #to exit
        exit()
    print("Updated contact list")
    print(contact_list)
    print("\n")
    contact_list = contact_list
    continue_option=input("Do you want to continue (Y to continue/ Any key to exit )? ")
         #ask if user wants to continue
    if (continue_option) == "y" or (continue_option) == "Y" :
        print(contact_list)
        user_choice(contact_list)
    return contact_list



stored_contact_list=[{"name":"name1","number":None ,"email":"name@rmail.com"},{"name":"
    name1","number":54321,"email":"name1@rmail.com"},{"name":"name3","number":5421,"
    email":"na1@rmail.com"}]
print(stored_contact_list)
main(stored_contact_list)
```

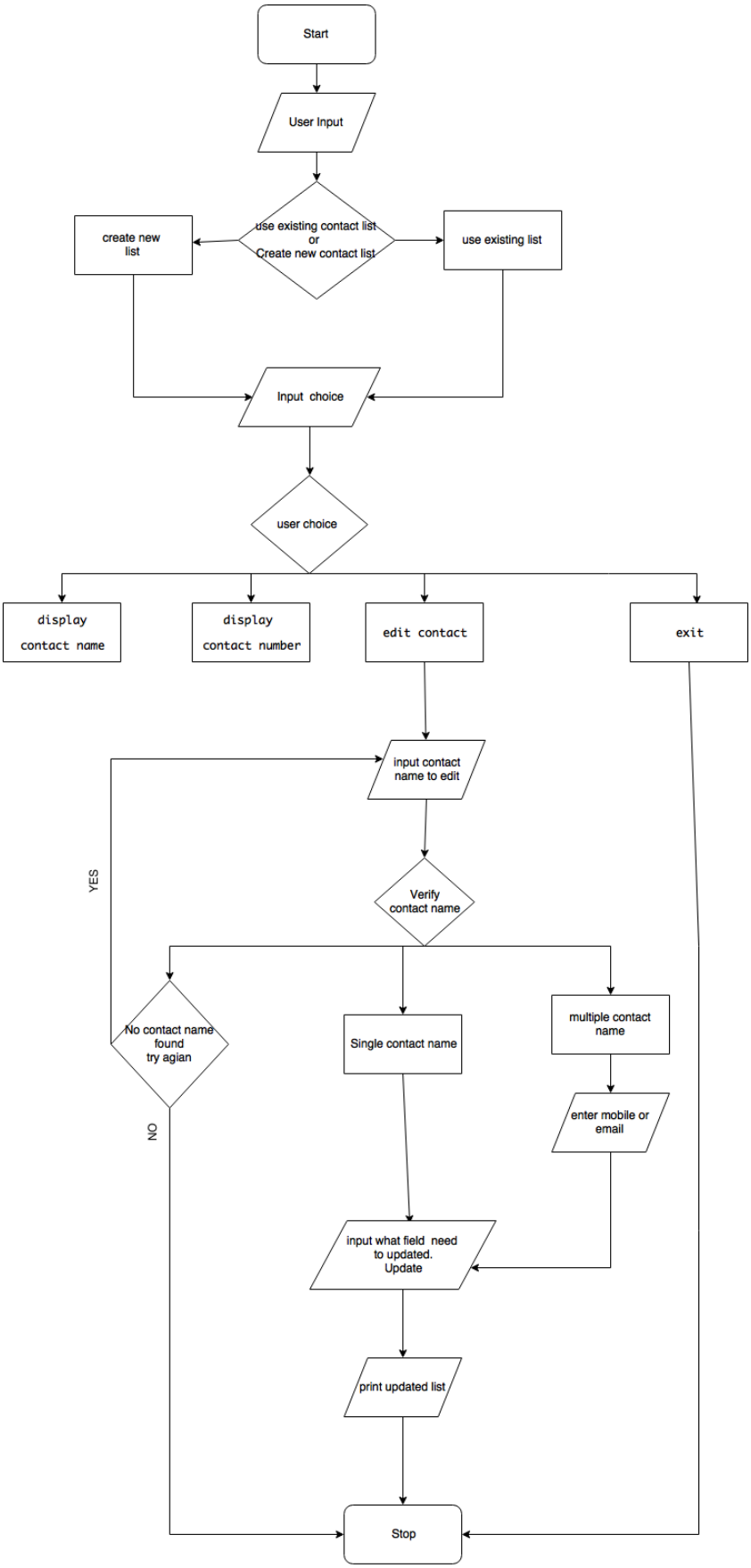


Figure 3: Problem 2:Flowchart

```
Do you want to use existing list or create a new list (press 1 to use existing list and 2 to create new list ? 1
Press 1 to display contact name
Press 2 to display contact number
Press 3 to edit contact
Press 4 to exit
1
Contact Names
name1 name2 ,   name name2 ,   name1 ,

Updated contact list
[{'name': 'name1 name2', 'number': None, 'email': 'name@rmail.com'}, {'name': 'name name2', 'number': 54321, 'email': 'name1@rmail.com'}, {'name': 'name1', 'number': 5421


Do you want to continue (Y to continue/ Any key to exit )? y
[{'name': 'name1 name2', 'number': None, 'email': 'name@rmail.com'}, {'name': 'name name2', 'number': 54321, 'email': 'name1@rmail.com'}, {'name': 'name1', 'number': 5421
Press 1 to display contact name
Press 2 to display contact number
Press 3 to edit contact
Press 4 to exit
2
Contact Numbers
name name2   54321
name1   5421
Updated contact list
[{'name': 'name1 name2', 'number': None, 'email': 'name@rmail.com'}, {'name': 'name name2', 'number': 54321, 'email': 'name1@rmail.com'}, {'name': 'name1', 'number': 5421


Do you want to continue (Y to continue/ Any key to exit )?
```

```
[{'name': 'name1', 'number': None, 'email': 'name@rmail.com'}, {'name': 'name2', 'number': 54321, 'email': 'name1@rmail.com'}, {'name': 'name3', 'number': 5421, 'email':
Do you want to use existing list or create a new list (press 1 to use existing list and 2 to create new list ? 1
Press 1 to display contact name
Press 2 to display contact number
Press 3 to edit contact
Press 4 to exit
3
Enter the contact name to edit name1
What contact details do you want to edit (Name/Number/email) ? number
Enter new number: 123456
Do you want to edit more contacts (Y)? n
Updated contact list
[{'name': 'name1', 'number': '123456', 'email': 'name@rmail.com'}, {'name': 'name2', 'number': 54321, 'email': 'name1@rmail.com'}, {'name': 'name3', 'number': 5421, 'emai


Do you want to continue (Y to continue/ Any key to exit )? n

Process finished with exit code 0
```

```
[{'name': 'name1', 'number': None, 'email': 'name@rmail.com'}, {'name': 'name1', 'number': 54321, 'email': 'name1@rmail.com'}, {'name': 'name3', 'number': 5421, 'email':
Do you want to use existing list or create a new list (press 1 to use existing list and 2 to create new list ? 1
Press 1 to display contact name
Press 2 to display contact number
Press 3 to edit contact
Press 4 to exit
3
Enter the contact name to edit name1
2 contacts with the name: name1
Please input either mobile number or email address of the contactname@rmail.com
What contact details do you want to edit (Name/Number/email) ? number
Enter new number: 8165857500
Updated contact list
[{'name': 'name1', 'number': '8165857500', 'email': 'name@rmail.com'}, {'name': 'name1', 'number': 54321, 'email': 'name1@rmail.com'}, {'name': 'name3', 'number': 5421,


Do you want to continue (Y to continue/ Any key to exit )? n

Process finished with exit code 0
```

Figure 4: Problem 2:Output

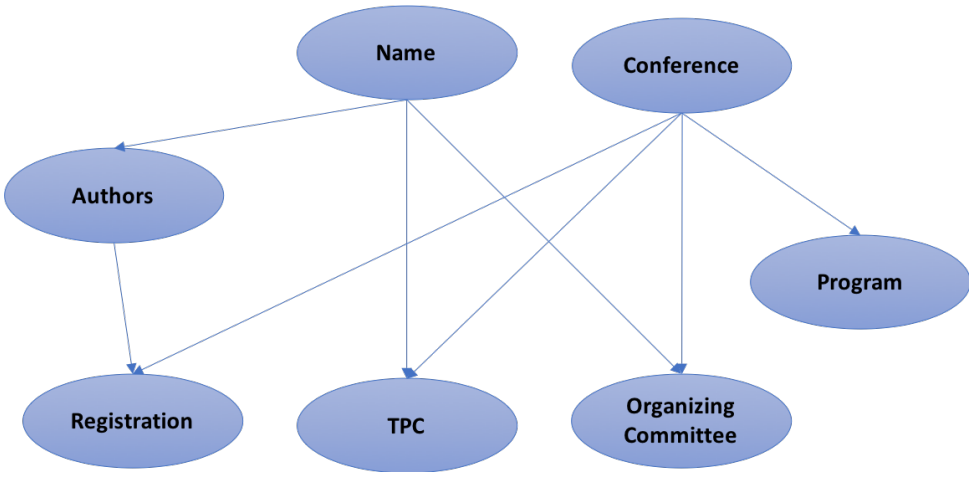


Figure 5: Problem 3 Class Inheritance

5.3 Problem 3

Write a python program to create any management system. The program should fulfill the following requisite. Should have atleast five classes.
Should have _init_ constructor in all the classes
Should show inheritance atleast once
Should have one super call
Use of self is required
Use at least one private data member
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.

5.4 Approach

For this problem, I have created 7 classes:

- Conference
- Name
- Authors
- Registration
- OrganizingCommittee
- TechnicalProgrammeCommitte
- Programme

All classes have _init_ constructors. Class *Conference* for the conference name.

```
class Conference: # Class 1 Conference Class
    def __init__(self,conference_name): # _init_ constructor
```

```
        self.conference_name=conference_name
```

Name Class for person's first and last name

```
class Name: #class 2 Name Class
    def __init__(self,first_name, last_name): # _init_ constructor
        self.first_name=first_name
        self.last_name=last_name
    def print_name(self):
        print("The Author Name is %s %s"%(self.first_name,self.last_name))
```

Authors class will inherit from *Name* class and will be used for Author's first name, last name and track information.

```
class Authors(Name):#class 3 # Author class inheritance from Name class
    #to count the total authors
    def __init__(self,first_name, last_name,author_track): # _init_ constructor
        super(Authors, self).__init__(first_name,last_name) #super call
        self.author_track=author_track
```

Registration class will show multiple inheritance from *Authors* and *Conference* classes. It will also have a data member to count the number of registrations. Here the registration number will be a private data attribute.

```
class Registration(Conference,Authors): #class 4 Registartion Class #inheritance from
    Confrence and Author class
    total_person=0
    def __init__(self,conference_name,first_name, last_name, author_track,
        registration_number): # _init_ constructor
        Conference.__init__(self,conference_name)
        Authors.__init__(self,first_name, last_name, author_track)
        self.__registration_number = registration_number #private data attribute
        Registration.total_person+=1
```

OrganizingCommittee class will show multiple inheritance from *Conference* and *Name* classes. and will be used organizing committee details and their role.

```
class OrganizingCommittee(Conference,Name): #class 5 #inheritance from Conference class
    and Name Class
    def __init__(self,conference_name,first_name, last_name,role): # _init_ constructor
        Conference.__init__(self,conference_name)
        Name.__init__(self,first_name, last_name)
```

TechnicalProgrammeCommitte class will show multiple inheritance from *Conference* and *Name* classes. and will be used tpc details and their tracks.

```
class TechnicalProgrammeCommitte(Conference,Name): #class 6 ##inheritance from
    Conference and Name Class
    def __init__(self,conference_name,first_name, last_name, track_name): # _init_
        constructor
        Conference.__init__(self,conference_name)
        Name.__init__(self,first_name, last_name)
        self.track_name=track_name
```

Programme class will show multiple inheritance from *Conference* and will be used for program name and schudule.

```
class Programme (Conference): #class 7 Inheritence from Conference class
    def __init__(self,conference_name, programme_name, schedule): # _init_ constructor
        Conference.__init__(self,conference_name)
        self.programme_name=programme_name
        self.schedule=schedule
```

Algorithm 3: Algorithm for Problem 3

Result: Created classes , demonstrate inheritance, private data member, instances
Define Class for Conference;
Define Class for Name;
Define Class forAuthors;
Define Class for Registration;
Define Class for OrganizingCommittee;
Define Class for TechnicalProgrammeCommitte;
Define Class for Programme;
Create Instances for all the class;

5.4.1 Code

This section presents the code along with comments to understand each block of the code.

```
import random

class Conference: # Class 1 Conference Class
    def __init__(self,conference_name): # _init_ constructor
        self.conference_name=conference_name


class Name: #class 2 Name Class
    def __init__(self,first_name, last_name): # _init_ constructor
        self.first_name=first_name
        self.last_name=last_name
    def print_name(self):
        print("The Author Name is %s %s"%(self.first_name,self.last_name))
```

```
class Authors(Name):#class 3 # Author class inheritance from Name class
    #to count the total authors
    def __init__(self,first_name, last_name,author_track): # _init_ constructor
        super(Authors, self).__init__(first_name,last_name) #super call
        self.author_track=author_track



class Registration(Conference,Authors): #class 4 Registartion Class #inheritance from
    Confrence and Author class
    total_person=0
    def __init__(self,conference_name,first_name, last_name, author_track,
        registration_number): # _init_ constructor
        Conference.__init__(self,conference_name)
        Authors.__init__(self,first_name, last_name, author_track)
        self.__registration_number = registration_number #private data attribute
        Registration.total_person+=1


class OrganizingCommittee(Conference,Name): #class 5 #inheritance from Conference class
    and Name Class
    def __init__(self,conference_name,first_name, last_name,role): # _init_ constructor
        Conference.__init__(self,conference_name)
        Name.__init__(self,first_name, last_name)
        self.role=role


class TechnicalProgrammeCommitte(Conference,Name): #class 6 ##inheritance from
    Conference and Name Class
    def __init__(self,conference_name,first_name, last_name, track_name): # _init_
        constructor
        Conference.__init__(self,conference_name)
        Name.__init__(self,first_name, last_name)
        self.track_name=track_name


class Programme (Conference): #class 7 Inheritence from Conference class
    def __init__(self,conference_name, programme_name, schedule): # _init_ constructor
        Conference.__init__(self,conference_name)
        self.programme_name=programme_name
        self.schedule=schedule


c=Conference
n=Name
a=Authors
r=Registration
o=OrganizingCommittee
t=TechnicalProgrammeCommitte
p=Programme

value_1_conference= c("ICC")
value_1_name=n("Rohit","Abhishek")
value_1_author=a("Rohit","Abhishek","Smart Cities")
```

```
value_2_name=n("tom","trump")
value_2_author=a("tom","trump","Security")
value_3_name=n("Abc","def")
value_3_author=a("Abc","def","big data")

value_1_registration=r("ICC","Rohit","Abhishek","Smart Cities",''.join(random.choice
    ('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ') for i in range(12)))
value_2_registration=r("ICC","tom","trump","Security",''.join(random.choice('0123456789
    ABCDEFGHIJKLMNOPQRSTUVWXYZ') for i in range(12)))
value_3_registration=r("ICC","Abc","def","big data",''.join(random.choice('0123456789
    ABCDEFGHIJKLMNOPQRSTUVWXYZ') for i in range(12)))
value_1_organizing_committe1=o("ICC","Deep", "Medhi","Executive-Chair")
value_1_organizing_committe2=o("ICC","Yi", "Qian","TPC Chair")
value_1_organizing_committe3=o("ICC","Rohit", "Abhishek","Web Content Chair")
value_1_tpc1=t("ICC","Jack","Daniels","Wireless Communications")
value_1_tpc2=t("ICC","Chivas","Regal","Security")
value_1_programme1=p("ICC","Keynote","9:00 AM - 10:30 AM")
value_1_programme2=p("ICC","Smart Cities Track,","12:00 PM - 2:30 PM")

print("Conference Name is :", value_1_conference.conference_name)
print("Author Name is :", value_1_author.first_name,value_1_author.last_name)
print("Author registered for \"%s\" track" %(value_1_author.author_track))
print("Author's registration code is:",value_1_registration.
    _Registration__registration_number)

print("Author Name is :", value_2_author.first_name,value_2_author.last_name)
print("Author registered for \"%s\" track" %(value_2_author.author_track))
print("Author's registration code is:",value_2_registration.
    _Registration__registration_number)

print("Author Name is :", value_3_author.first_name,value_3_author.last_name)
print("Author registered for \"%s\" track" %(value_3_author.author_track))
print("Author's registration code is:",value_3_registration.
    _Registration__registration_number)
print("Total Author registered are ",r.total_person)

print("Organizing Committee Members are:")
print(value_1_organizing_committe1.first_name,value_1_organizing_committe1.last_name
      ,":",value_1_organizing_committe1.role,"\n",value_1_organizing_committe2.
          first_name,value_1_organizing_committe2.last_name
      ,":",value_1_organizing_committe2.role,"\n",value_1_organizing_committe3.
          first_name,value_1_organizing_committe3.last_name
      , ":",value_1_organizing_committe3.role,"\n")
print("Technical Programme Committee Members are:")

print(value_1_tpc1.first_name,value_1_tpc1.last_name,":",value_1_tpc1.track_name,"\n",
value_1_tpc2.first_name,value_1_tpc2.last_name,":",value_1_tpc2.track_name)

print("Programme ")
print(value_1_programme1.programme_name,":",value_1_programme1.schedule,"\n",
     value_1_programme2.programme_name, ":", value_1_programme2.schedule)
```

```
Conference Name is : ICC
Author Name is : Rohit Abhishek
Author registered for "Smart Cities" track
Author's registration code is: FB0CQAEWQ4LU
Author Name is : tom trump
Author registered for "Security" track
Author's registration code is: J4ODLFR20YF0
Author Name is : Abc def
Author registered for "big data" track
Author's registration code is: CS5FHPFKXM5W
Total Author registered are  3
Organizing Committee Members are:
Deep Medhi : Executive-Chair
 Yi Qian : TPC Chair
 Rohit Abhishek : Web Content Chair

Technical Programme Committee Members are:
Jack Daniels : Wireless Communications
 Chivas Regal : Security
Programme
Keynote : 9:00 AM - 10:30 AM
 Smart Cities Track, : 12:00 PM - 2:30 PM

Process finished with exit code 0
```

Figure 6: Problem 3:Output

5.5 Problem 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.

5.5.1 Approach

In this program, we need to randomly generate random vector of size 15 and then print the value whose frequency is the most. For this we need to create two functions. One function would generate the random vector, and other would be use to find the frequency.

Function ***random_ array*** will be used to generate random vector of size 15. The code is shown below.

```
def random_array():
    z= np.random.randint(0, 20, 15) #creating random values
    print(z)
    return z
```

Function ***most_frequent_ value*** will be used to find the most frequent value. The approach would be to find the most frequent value by using the function *np.bincount().argmax()* . However this won't display if there are more than one most frequent items. So we need to create an array of unique items and then check the frequency of each item in the vector list.

```
def most_frequent_value(z):
    most_frequency=np.bincount(z).argmax() #this would give the most frequent value
                                #however if 2 values are most frequent, it wont display
                                     both of the values
    unique_values=np.unique(z) #this would create array of uniques values
    print("The most frequent values have been repeated %d times and the values are "%(z.
        count(most_frequency)))
    for i in unique_values:
        if z.count(most_frequency)==z.count(i):
            print(i)
```

Algorithm 4: Algorithm for Problem 4

Result: Find most frequent value
Define function to create random array;
Define function to find the most frequent item;
call the function random_array to generate random vectors ;
if *If number of most frequent items >1* **then**
| Find all items and print them
else
| print the most frequent item
end

5.5.2 Code

This section presents the code along with comments to understand each block of the code.

```
import numpy as np #import the numpy library
from collections import Counter
from datetime import datetime
import datetime
import time

a = time.time()
def random_array():
    z= np.random.randint(0, 20,15) #creating random values
    print(z)
    return z
def most_frequent_value(z):
    most_frequency=np.bincount(z).argmax() #this would give the most frequent value
                                  #however if 2 values are most frequent, it wont display
                                       both of the values
    unique_values=np.unique(z) #this would create array of uniques values
    print("The most frequent values have been repeated %d times and the values are "%(z.
        count(most_frequency)))
    for i in unique_values:
        if z.count(most_frequency)==z.count(i):
            print(i)




z=random_array()

#z= [ 6 , 4 , 9 , 3, 9 ,18, 7, 15, 9, 17, 1 ,19, 11, 17, 0]
most_frequent_value(list(z))

b = time.time()
print("Time to simulate ",(b-a))
```

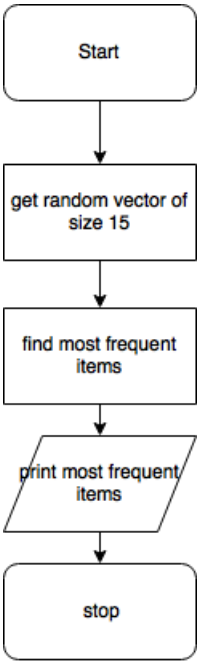
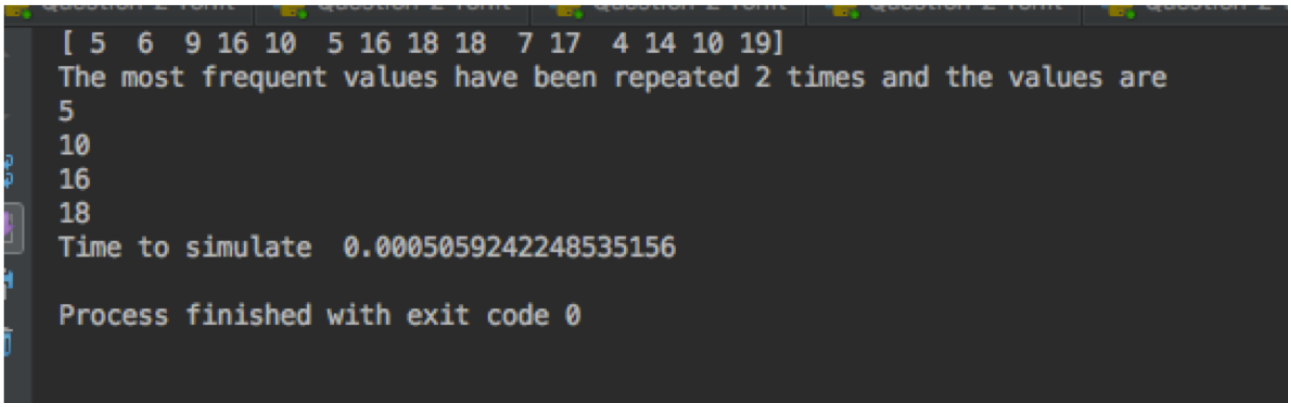


Figure 7: Problem 4:Flowchart



```
[ 5  6  9 16 10  5 16 18 18  7 17  4 14 10 19]
The most frequent values have been repeated 2 times and the values are
5
10
16
18
Time to simulate  0.0005059242248535156

Process finished with exit code 0
```

Figure 8: Problem 4:Output

6 Deployment

For deployment of the code, the following process should be followed:

- Install python on your system. It can be download from https://www.python.org/downloads/
- Download the code from my git. (https://github.com/ra5xc/CSEE5590/tree/master/Assignment%202)
- Import libraries validate_email, random, numpy, time
- Execute the code using python

I would highly suggest to read the report before executing the code to better understand the functionality of the codes. The report can be downloaded from my git as well.

7 Limitation

In problem 5.1 we are using dictionary, which uses hashing process, so it takes up lot of space if we have lots of items. Using list of lists would have been better as we are iterating through items. Its easier for a user to enter a bad hash function when the dictionary is large. The insert time is more. The performance is limited when we have lot of items in the dictionary.

In problem 5.2 we are using list of dictionaries, which again faces the issues as in problem 5.1. The performance, time complexity and memory would be limitation as the size increases.

In problem 5.3 The use of Class completely depends on how many and what kind of scenarios/data needs to be processed. If we are processing highly related set of datas, then class maybe useful. If not, then it tends to make the program more complex and slower as compared to using functions. Classes are not recommended if they just have static method.

In problem 5.5 Numpy has limitations over finding the most frequent items. If they are multiple items with same frequency, it would display only one. So we have to create multiple arrays to find multiple most frequent elements . For larger items, this poses a limitation. It would increase the complexity and as a result the processing time would be more.

8 References

https://github.com
https://www.jetbrains.com/pycharm/