Navya Battu

Class Id : 8

ICP Id : 14

Objective : The following assignment focus on to make one familiar with python programming which includes different concepts like Lists,Dictionories,OOPS,Numpy.

Features :

1. Lists
2. Dictionaries
3. Iterative loops
4. Functions
5. OOPS concepts
6. Numpy
7. Python Inbuilt and mathematical Functions

Configurations : Python 3.6, PyCharm (COMMUNITY 2017.3) Implementation :

Question1:

Input :

umkcdict | dictinlist.py | classesss | randvector.py

```python
dict = {"python":50,"web":30,"c":20,"java":40}
minm = int(input("Enter the minm price range : "))
maxm = int(input("Enter the maxm price range : "))
booklist = []
for price in range(minm,maxm+1):
    for bookname in dict:
        if dict[bookname] == price:
            booklist.append(bookname)
print("you can purchase books : " , booklist)
```

```
/Users/navyabattu/venv/PythonLab2/bin/python /Users/navyabattu/PycharmProjects/PythonLab2/umkcdict
Enter the minm price range : 20
Enter the maxm price range : 40
you can purchase books :  ['c', 'web', 'java']

Process finished with exit code 0
```

Question2:

Input:

umkcdict | dictinlist.py | classesss | randvector.py

```python
n = int(input("Number of contacts : "))
contacts = []
contact_dict = {}
for i in range(n) :
    contact_dict["name"] = input("Enter name ")
    contact_dict["number"] = input("Enter number ")
    contact_dict["email"] = input("Enter email ")
    contacts.append(contact_dict.copy())
print(contacts)
def opt() :
    x = str(input("a)Display contact by name \nb)Display contact by number \nc)Edit contact by name \nd)Exit:\n "))
    return x

if opt()=='a':
    name = str(input("Enter the name: "))
    for a in contacts:
        if a['name'] == name:
            print(a)
if opt()=='b':
    num = str(input("Enter the number: "))
    for a in contacts:
        if a['number'] == num:
            print(a)
if opt()=='c':
    name = str(input("Enter the name: "))
    number = int(input("Enter the number: "))
    for index,a in enumerate(contacts):
        if a['name'] == name:
            contacts[index]['number']=number
    print(contacts)
if opt()=='d' :
    exit()
```

```
dictinlist  dictinlist  dictinlist  dictinlist  dictinlist  dictinlist  dictinlist
Number of contacts : 3
Enter name Navya
Enter number 8163281380
Enter email nbh95@mail.umkc.edu
Enter name Harika
Enter number 8163282345
Enter email ha5gz@mail.umkc.edu
Enter name Sulochana
Enter number 8164356754
Enter email smvbr@mail.umkc.edu
[{'name': 'Navya', 'number': '8163281380', 'email': 'nbh95@mail.umkc.edu'}, {'name': 'Harika', 'number': '8163282345', 'email': 'ha5gz@mail.umkc.edu'}, {'name': 'Sulochana',
a)Display contact by name
b)Display contact by number
c)Edit contact by name
d)Exit:
 a
Enter the name: Navya
{'name': 'Navya', 'number': '8163281380', 'email': 'nbh95@mail.umkc.edu'}
a)Display contact by name
b)Display contact by number
c)Edit contact by name
d)Exit:
 b
Enter the number: 8163282345
{'name': 'Harika', 'number': '8163282345', 'email': 'ha5gz@mail.umkc.edu'}
a)Display contact by name
b)Display contact by number
c)Edit contact by name
d)Exit:
 c
Enter the name: Sulochana
Enter the number: 9999999999
[{'name': 'Navya', 'number': '8163281380', 'email': 'nbh95@mail.umkc.edu'}, {'name': 'Harika', 'number': '8163282345', 'email': 'ha5gz@mail.umkc.edu'}, {'name': 'Sulochana',
a)Display contact by name
b)Display contact by number
c)Edit contact by name
d)Exit:
 d

Process finished with exit code 0
```

```
dictinlist  dictinlist  dictinlist  dictinlist  dictinlist  dictinlist  dictinlist
battu/PycharmProjects/PythonLab2/dictinlist.py

5@mail.umkc.edu'}, {'name': 'Harika', 'number': '8163282345', 'email': 'ha5gz@mail.umkc.edu'}, {'name': 'Sulochana', 'number': '8164356754', 'email': 'smvbr@mail.umkc.edu'}]

@mail.umkc.edu'}

z@mail.umkc.edu'}

5@mail.umkc.edu'}, {'name': 'Harika', 'number': '8163282345', 'email': 'ha5gz@mail.umkc.edu'}, {'name': 'Sulochana', 'number': 9999999999, 'email': 'smvbr@mail.umkc.edu'}]
```

Question3:

Input:

umkcdict | dictinlist.py | classesss | randvector.py

```python
class Student:  # Class #1
    overall_total = 0
    def _init(self):  # use of __init_ method
        self.name = input('Enter the name of the student ')
        self.id = input('Enter the id number')
        Student.overall_total += 1
    def count(self):    # Defining a function inside a class
        print('The number of students enrolled are ', Student.overall_total)
    def display(self):
        print('The name of the student is ', self.name)
        print('The roll number of the student is ', self.id)
class TransferStudent(Student):  # Class #2
    def _init_(self):
        super(TransferStudent, self)._init_()  # use of super() call
        self.TransferredCredits = input('Enter the number of credits that are transferred')
class System:  # Class #3
    def _init_(self):
        self.TypeOfSystem = input('Enter the system online or inclass: ')
    def display(self):
        print('The system the student enrolled is: ', self.TypeOfSystem)
class Grades(TransferStudent):  # Class #4
    def _init_(self, grade, credits):
        TransferStudent._init_(self)  # Another way of inheriting the parent class
        self.Grades = grade
        self.EnrolledCredits = credits

    def TotalCredits(self):
        self.TotalCreditsEnrolled = self.TransferredCredits + credits
        print('The total number of credits completed: ', self.TotalCreditsEnrolled)

    def display(self):
        print('The name of the student is ', self.name)
        print('The roll number of the student is ', self.id)
        print('The Transferred credits are: ', self.TransferredCredits)
        print('The total number of credits enrolled: ', self.EnrolledCredits)
        print('The Grade obtained: ', self.Grades)
```

```python
class Attendance:  # Class #5
    def _init_(self, percentage):
        self.__attendance = percentage  #  private data member
        if self.__attendance < 65:
            print("Student's attendance is low")
#  instances of all the classes

Student1 = Student()
Student2 = Student()
Student3 = Student()

Student4 = TransferStudent()
Student5 = TransferStudent()

Student6 = Grades("B", "30")
Student7 = Grades("A", "45")

Student8 = Attendance(64)

Student7.display()

Student5.display()

Student7.count()
Student1.display()
```

Output :

```
Run: dictinlist dictinlist dictinlist dictinlist dictinlist dictinlist dictinlist dictinlist classesss
/Users/navyabattu/venv/PythonLab2/bin/python /Users/navyabattu/PycharmProjects/PythonLab2/classesss
Enter the name of the student Navya
Enter the roll number16251845
Enter the name of the student Harika
Enter the roll number16251234
Enter the name of the student Sulochana
Enter the roll number16257890
Enter the name of the student Dilip
Enter the roll number16253456
Enter the number of credits that are transferred3
Enter the name of the student Navya
Enter the roll number
```

Question4:

Input:

```python
from random import randint
import numpy as np
Z = np.random.randint(0,20, size=(15))
print("Array: %s"  %Z)
print("Frequent item in the list:")
print(np.bincount(Z).argmax())
```

Output :

```
Array: [ 2  3  5  3  6  6 18 12 12  5  5 12  9 15  4]
Frequent item in the list:
5

Process finished with exit code 0
```

Implementation :

Question 1 code Explanation :

umkcdict | dictinlist.py | classesss | randvector.py

```python
dict = {"python":50,"web":30,"c":20,"java":40}
minm = int(input("Enter the minm price range : "))
maxm = int(input("Enter the maxm price range : "))
booklist = []
for price in range(minm,maxm+1):
    for bookname in dict:
        if dict[bookname] == price:
            booklist.append(bookname)
print("you can purchase books : " , booklist)
```

1. Input dictionary of books and its price are given as keys and values
2. Taking the minimum and maximum range of price from user

3. Starting the for loop with price range and nested for loop with bookname in dictionary
4. Condition for checking the price of books for given range and appending the booknames to list
5. Printing the books which can be purchased

Question 2 code Explanation :

umkcdict | dictinlist.py | classesss | randvector.py

```python
n = int(input("Number of contacts : "))
contacts = []
contact_dict = {}
for i in range(n) :
    contact_dict["name"] = input("Enter name ")
    contact_dict["number"] = input("Enter number ")
    contact_dict["email"] = input("Enter email ")
    contacts.append(contact_dict.copy())
print(contacts)
def opt() :
    x = str(input("a)Display contact by name \nb)Display contact by number \nc)Edit contact by name \nd)Exit:\n "))
    return x

if opt()=='a':
    name = str(input("Enter the name: "))
    for a in contacts:
        if a['name'] == name:
            print(a)
if opt()=='b':
    num = str(input("Enter the number: "))
    for a in contacts:
        if a['number'] == num:
            print(a)
if opt()=='c':
    name = str(input("Enter the name: "))
    number = int(input("Enter the number: "))
    for index,a in enumerate(contacts):
        if a['name'] == name:
            contacts[index]['number']=number
    print(contacts)
if opt()=='d' :
    exit()
```

1. Take input from user to create the contacts
2. Initialize the list and dictionary
3. Based on the range of input given by user , using for loop created the certain number of contacts
4. Created contacts with the help of dictionary(contact_dict) and stored it in a list (contacts)
5. Created a function for options and selecting the option from user
6. Starting the conditional statements for each option
7. For option 'a' Display contact by name : We need to enter the name , it gives all the details of that particular contact

8. For option 'b' Display contact by number : We need to enter the number, it gives all the details of that particular contact
9. For option 'c' Edit Contact by name : We need to enter the name of the contact we want to update, enter the updated number , it will get updated in list.
10. Now it prints all contact details after update
11. For option 'd' Exit : exit() function exits from the program

Question 3 code Explanation :

umkcdict | dictinlist.py | classesss | randvector.py

```python
class Student:  # Class #1
    overall_total = 0
    def _init(self):  # use of __init_ method
        self.name = input('Enter the name of the student ')
        self.id = input('Enter the id number')
        Student.overall_total += 1
    def count(self):    # Defining a function inside a class
        print('The number of students enrolled are ', Student.overall_total)
    def display(self):
        print('The name of the student is ', self.name)
        print('The roll number of the student is ', self.id)
class TransferStudent(Student):  # Class #2
    def _init_(self):
        super(TransferStudent, self)._init_()  # use of super() call
        self.TransferredCredits = input('Enter the number of credits that are transferred')
class System:  # Class #3
    def _init_(self):
        self.TypeOfSystem = input('Enter the system online or inclass: ')
    def display(self):
        print('The system the student enrolled is: ', self.TypeOfSystem)
class Grades(TransferStudent):  # Class #4
    def _init_(self, grade, credits):
        TransferStudent._init_(self)  # Another way of inheriting the parent class
        self.Grades = grade
        self.EnrolledCredits = credits

    def TotalCredits(self):
        self.TotalCreditsEnrolled = self.TransferredCredits + credits
        print('The total number of credits completed: ', self.TotalCreditsEnrolled)

    def display(self):
        print('The name of the student is ', self.name)
        print('The roll number of the student is ', self.id)
        print('The Transferred credits are: ', self.TransferredCredits)
        print('The total number of credits enrolled: ', self.EnrolledCredits)
        print('The Grade obtained: ', self.Grades)
```

```python
class Attendance:  # Class #5
    def _init_(self, percentage):
        self.__attendance = percentage  #  private data member
        if self.__attendance < 65:
            print("Student's attendance is low")
#  instances of all the classes

Student1 = Student()
Student2 = Student()
Student3 = Student()

Student4 = TransferStudent()
Student5 = TransferStudent()

Student6 = Grades("B", "30")
Student7 = Grades("A", "45")

Student8 = Attendance(64)

Student7.display()

Student5.display()

Student7.count()
Student1.display()
```

1. Five classes were created for student enrollment system
2. Init constructor functions were declared
3. Self function is used
4. Instances were created for objects
5. Inheritance has been implemented for Student base class
6. Super function is used

Question 4 code Explanation :

```
from random import randint
import numpy as np
Z = np.random.randint(0,20, size=(15))
print("Array: %s"   %Z)
print("Frequent item in the list:")
print(np.bincount(Z).argmax())
```

1. Import randint from random
2. Import numpy library
3. Taking the random numbers in the range of 20 with size 15
4. Printing the Array generated
5. Finding the frequent integer with "np.bincount()" in the list and printing it

Limitations :

1. Calling function everytime to check condition in 2nd program
2. Additional libraries needed

References :

https://www.tutorialspoint.com/python/index.htm

https://www.geeksforgeeks.org/

https://stackoverflow.com