Jagan Institute of Management Studies

3, Institutional Area, Sector-5, Rohini, Delhi-110085

Course Code: MCA-166
Course Name: Python Programming
Mapping Outcomes: CO3, CO4

Assignment - III

1) Create a Class which Performs Basic Calculator Operations.

```
class Calculator:
    def run_cal(self):
        print("Welcome to Calculator!")

        while True:
            num1 = int(input("Enter number 1: "))
            num2 = int(input("Enter number 2: "))

            print("1. Add")
            print("2. Subtract")
            print("3. Divide")
            print("4. Multiply")
            print("5. Exit")

            choice = int(input("Enter Choice: "))

            if choice == 1:
                print("SUM:", self.__add(num1, num2))
            elif choice == 2:
                print("DIFF:", self.__sub(num1, num2))
            elif choice == 3:
                if num2 == 0:
                    print("Error: Division by zero is not allowed.")
                else:
                    print("DIV:", self.__div(num1, num2))
            elif choice == 4:
                print("PRO:", self.__mul(num1, num2))
            elif choice == 5:
                print("Exiting calculator.")
                break
            else:
                print("Invalid choice! Please try again.")

    def __add(self, n1, n2):
        return n1 + n2

    def __sub(self, n1, n2):
        return n1 - n2

    def __div(self, n1, n2):
        return n1 / n2

    def __mul(self, n1, n2):
        return n1 * n2


calc = Calculator()
calc.run_cal()
```

Output :-

```
Welcome to Calculator!
Enter number 1:  101
Enter number 2:  100
1. Add
2. Subtract
3. Divide
4. Multiply
5. Exit
Enter Choice:  1
SUM: 201
Enter number 1:  100
Enter number 2:  200
1. Add
2. Subtract
3. Divide
4. Multiply
5. Exit
Enter Choice:  4
PRO: 20000
Enter number 1:  10
Enter number 2:  20
1. Add
2. Subtract
3. Divide
4. Multiply
5. Exit
Enter Choice:  2
DIFF: -10
Enter number 1:  100
Enter number 2:  10
1. Add
2. Subtract
3. Divide
4. Multiply
5. Exit
Enter Choice:  3
DIV: 10.0
Enter number 1:  10
Enter number 2:  20
1. Add
2. Subtract
3. Divide
4. Multiply
5. Exit
Enter Choice:  5
Exiting calculator.
```

2) Every time a vote is cast the name of the candidate is appended to the data structure. Print the names of candidates who received maximum vote in lexicographical order and if there is a tie print lexicographically smaller name.

```
class voter:
    votescount={}
    def vote(self):
        cont=1
        while cont==1:
            name = input("Enter candidate name to vote : ")
            if name in voter.votescount :
                voter.votescount[name] +=1
            else :
                voter.votescount[name] =1
            cont=int(input("Do you wan to continue?:0/1"))
    def display(self):
        max=0
        winner=[]
        for cand in voter.votescount:
            if voter.votescount[cand]>max:
                max=voter.votescount[cand]
        for key,val in voter.votescount.items() :
            if(val==max):
                winner.append(key)
        print("Winners :")
        winner.sort()
        for win in winner:
            print(win)

v = voter()
v.vote()
v.display()
```

OUTPUT:-

```
Enter candidate name to vote :  sarthak
Do you wan to continue?:0/1 1
Enter candidate name to vote :  sanika
Do you wan to continue?:0/1 1
Enter candidate name to vote :  sanika
Do you wan to continue?:0/1 1
Enter candidate name to vote :  sanika
Do you wan to continue?:0/1 0
Winners :
sanika
```

3) Create Birthday Reminder Application to notify users using file handling and exception handling

```
from datetime import datetime
class ManageBday:
   def addBday(self):
      name = input("Enter name: ").lower()

      try:
         with open("bday.txt", "r") as bdays:
            birthday_list = bdays.readlines()
```

```
        except FileNotFoundError:
            birthday_list = []

        entered = False

        for index, bday in enumerate(birthday_list):
            if bday.startswith(name + ","):
                entered = True
                choice = int(input("Already Exists. Want to change? (1 for Yes, 0 for
No): "))
                if choice == 1:
                    month = int(input("Enter month (1-12): "))
                    day = int(input("Enter day (1-31): "))
                    year = int(input("Enter year (YYYY): "))
                    birthday_list[index] = f"{name},{day}/{month}/{year}\n"
                break

        if not entered:
            month = int(input("Enter month (1-12): "))
            day = int(input("Enter day (1-31): "))
            year = int(input("Enter year (YYYY): "))
            birthday_list.append(f"{name},{day}/{month}/{year}\n")

        with open("bday.txt", "w") as bdays:
            bdays.writelines(birthday_list)

        print("Birthday added/updated successfully.")

    def checkBday(self):
        name = input("Enter name to find: ").lower()
        try:
            with open("bday.txt", "r") as bdays:
                exists = False
                for bday_entry in bdays:
                    if bday_entry.startswith(name + ","):
                        date = bday_entry.strip().split(",")[1].split("/")
                        print("Birthday on:", date[0], date[1], date[2])
                        exists = True
                        break

                if not exists:
                    print("NOT FOUND")
        except FileNotFoundError:
            print("No birthdays stored yet.")
```

```
    from datetime import datetime

class ManageBday:
    def addBday(self):
        name = input("Enter name: ").lower()

        try:
            with open("bday.txt", "r") as bdays:
                birthday_list = bdays.readlines()
        except FileNotFoundError:
            birthday_list = []

        entered = False

        for index, bday in enumerate(birthday_list):
            if bday.startswith(name + ","):
                entered = True
                choice = int(input("Already Exists. Want to change? (1 for Yes, 0 for
No): "))
                if choice == 1:
                    month = int(input("Enter month (1-12): "))
                    day = int(input("Enter day (1-31): "))
                    year = int(input("Enter year (YYYY): "))
                    birthday_list[index] = f"{name},{day}/{month}/{year}\n"
                break

        if not entered:
            month = int(input("Enter month (1-12): "))
            day = int(input("Enter day (1-31): "))
            year = int(input("Enter year (YYYY): "))
            birthday_list.append(f"{name},{day}/{month}/{year}\n")

        with open("bday.txt", "w") as bdays:
            bdays.writelines(birthday_list)

        print("Birthday added/updated successfully.")

    def checkBday(self):
        name = input("Enter name to find: ").lower()
        try:
            with open("bday.txt", "r") as bdays:
                exists = False
                for bday_entry in bdays:
                    if bday_entry.startswith(name + ","):
```

```
                date = bday_entry.strip().split(",")[1].split("/")
                print("Birthday on:", date[0], date[1], date[2])
                exists = True
                break

        if not exists:
            print("NOT FOUND")
    except FileNotFoundError:
        print("No birthdays stored yet.")

def checkTodaysBday(self):
    today = datetime.today()
    today_month = str(today.month)  # No leading zero
    today_day = str(today.day)      # No leading zero

    try:
        with open("bday.txt", "r") as bdays:
            found = False
            for bday_entry in bdays:
                name, date = bday_entry.strip().split(",")
                year, month, day = date.split("/")  # Extract YYYY, MM, DD

                if month == today_month and day == today_day:
                    print(f"🎉 Today is {name.title()}'s Birthday! 🎂")
                    found = True
            if not found:
                print("No birthdays today.")
    except FileNotFoundError:
        print("No birthdays stored yet.")

mb = ManageBday()
print("WELCOME")
mb.checkTodaysBday()
print("ADDING A BDAY ENTRY ")
mb.addBday()
print("FINDING BDAY ")
mb.checkBday()
```

OUTPUT :-

Bday.txt

```
1 sarthak,2002/11/21
2 sanika,2003/4/3
3
```

```
WELCOME
🎉 Today is Sanika's Birthday! 🎂
ADDING A BDAY ENTRY
Enter name:  sarthak
Already Exists. Want to change? (1 for Yes, 0 for No):  1
Enter month (1-12):  11
Enter day (1-31):  1
Enter year (YYYY):  2003
Birthday added/updated successfully.
FINDING BDAY
Enter name to find:  sarthak
Birthday on: 1 11 2003
```

```
1 sarthak,1/11/2003
2 sanika,2003/4/3
3
```

4) Create a class "Time" and initialize it with hrs and mins.
   1. Make a method addTime which should take two time object and add them. E.g.- (6 hour and 35 min)+(2 hr and 12 min) is (8 hr and 47 min)
   2. Make a method showTime which should print the time.
   3. Make a method showMinute which should display the total minutes in the Time. E.g.- (2 hr 6 min) should display 126 minutes.

```python
class Time:
    def __init__(self, hrs, mins):
        self.hrs = hrs
        self.mins = mins

    def addTime(self, other):
        total_mins = self.mins + other.mins
        total_hrs = self.hrs + other.hrs + (total_mins // 60)
        total_mins %= 60

        return Time(total_hrs, total_mins)

    def showTime(self):
        print(f"Time: {self.hrs} hr {self.mins} min")

    def showMinute(self):
        total_minutes = (self.hrs * 60) + self.mins
        print(f"Total minutes: {total_minutes}")

time1 = Time(6, 35)
time2 = Time(2, 12)

time3 = time1.addTime(time2)

time3.showTime()
time3.showMinute()
```

```
Time: 8 hr 47 min
Total minutes: 527
```

5) Write a Python class which has two methods get_String and print_String. get_String accept a string from the user and print_String print the string in upper case.

```
class StringManipulator:
    def __init__(self):
        self.user_string = ""

    def get_String(self):
        self.user_string = input("Enter a string: ")

    def print_String(self):
        print(self.user_string.upper())

obj = StringManipulator()
obj.get_String()
obj.print_String()
```

```
Enter a string:  hello my world
HELLO MY WORLD
```

6) Create a class “GrandMother” with a subclass “Mother” with a subclass “Daughter”. All the three classes to have there own constructors. The object of only “Daughter” class to be made to fetch all the details of Grandmother and mother.

```
class GrandMother:
    def __init__(self):
        self.gm_name = input("Enter Grandmother's Name: ")
        self.gm_age = int(input("Enter Grandmother's Age: "))

    def show_grandmother_details(self):
        print(f"GrandMother's Name: {self.gm_name}, Age: {self.gm_age}")


class Mother(GrandMother):
    def __init__(self):
        super().__init__()
        self.m_name = input("Enter Mother's Name: ")
        self.m_age = int(input("Enter Mother's Age: "))

    def show_mother_details(self):
        print(f"Mother's Name: {self.m_name}, Age: {self.m_age}")


class Daughter(Mother):
    def __init__(self):
        super().__init__()
        self.d_name = input("Enter Daughter's Name: ")
        self.d_age = int(input("Enter Daughter's Age: "))

    def show_daughter_details(self):
        print(f"Daughter's Name: {self.d_name}, Age: {self.d_age}")

    def show_all_details(self):
        print("\nFetching all family details...\n")
        self.show_grandmother_details()
        self.show_mother_details()
        self.show_daughter_details()
```

```
daughter_obj = Daughter()
daughter_obj.show_all_details()
```

OUTPUT :-

```
Enter Grandmother's Name:  dr.nirdosh
Enter Grandmother's Age:  81
Enter Mother's Name:  asha sengar
Enter Mother's Age:  53
Enter Daughter's Name:  saumya
Enter Daughter's Age:  28

Fetching all family details...

GrandMother's Name: dr.nirdosh, Age: 81
Mother's Name: asha sengar, Age: 53
Daughter's Name: saumya, Age: 28
```

7) Create a class “Parent” and a subclass “Child”. The “Parent” holds a constructor with two parameters name and age and also one method print() that prints name and age. The “child to have only one constructor that send the value to parent constructor and calls print() from its constructor only. Note: the object of only “Child” class to be created.

```
class Parent:
    def __init__(self, parent_name, parent_age):
        self.parent_name = parent_name
        self.parent_age = parent_age

    def display(self):
        print(f"Parent's Name: {self.parent_name}, Age: {self.parent_age}")


class Child(Parent):
    def __init__(self, parent_name, parent_age, child_name, child_age):
        super().__init__(parent_name, parent_age)
        self.child_name = child_name
        self.child_age = child_age
        self.display()
    def display(self):
        super().display()
        print(f"Child's Name: {self.child_name}, Age: {self.child_age}")

parent_name = input("Enter Parent's Name: ")
parent_age = int(input("Enter Parent's Age: "))

child_name = input("Enter Child's Name: ")
child_age = int(input("Enter Child's Age: "))

child_obj = Child(parent_name, parent_age, child_name, child_age)
```

```
Enter Parent's Name:  Asha
Enter Parent's Age:  53
Enter Child's Name:  Sarthak
Enter Child's Age:  22
Parent's Name: Asha , Age: 53
Child's Name: Sarthak, Age: 22
```

8) Write a program to create an abstract class "Animal" that holds three declaration functions based on their characteristics. Create four subclasses – "Mammals", "Reptiles", "birds", and "amphibians" inherited from "Animal" class.

```python
from abc import ABC, abstractmethod
class Animal(ABC):
    @abstractmethod
    def movement(self):
        pass

    @abstractmethod
    def sound(self):
        pass

    @abstractmethod
    def habitat(self):
        pass

class Mammals(Animal):
    def movement(self):
        print("Mammals walk or run on land.")

    def sound(self):
        print("Mammals produce sounds like growling, barking, or talking.")

    def habitat(self):
        print("Mammals live on land and some in water (like whales).")

class Reptiles(Animal):
    def movement(self):
        print("Reptiles crawl or slither.")

    def sound(self):
        print("Reptiles make hissing or croaking sounds.")

    def habitat(self):
        print("Reptiles live in deserts, forests, and wetlands.")

class Birds(Animal):
    def movement(self):
        print("Birds fly using their wings.")

    def sound(self):
        print("Birds chirp, sing, or squawk.")

    def habitat(self):
        print("Birds live in trees, forests, and near water.")
```

```
class Amphibians(Animal):
    def movement(self):
        print("Amphibians jump, swim, or crawl.")

    def sound(self):
        print("Amphibians produce croaking or ribbit sounds.")

    def habitat(self):
        print("Amphibians live both on land and in water.")

print("Mammals:")
mammal = Mammals()
mammal.movement()
mammal.sound()
mammal.habitat()

print("\nReptiles:")
reptile = Reptiles()
reptile.movement()
reptile.sound()
reptile.habitat()

print("\nBirds:")
bird = Birds()
bird.movement()
bird.sound()
bird.habitat()

print("\nAmphibians:")
amphibian = Amphibians()
amphibian.movement()
amphibian.sound()
amphibian.habitat()
```

OUPUT:-

```
Mammals:
Mammals walk or run on land.
Mammals produce sounds like growling, barking, or talking.
Mammals live on land and some in water (like whales).

Reptiles:
Reptiles crawl or slither.
Reptiles make hissing or croaking sounds.
Reptiles live in deserts, forests, and wetlands.

Birds:
Birds fly using their wings.
Birds chirp, sing, or squawk.
Birds live in trees, forests, and near water.

Amphibians:
Amphibians jump, swim, or crawl.
Amphibians produce croaking or ribbit sounds.
Amphibians live both on land and in water.
```

9) Implement ATM simulation system displaying its operations using object oriented features.

```
class Account:
    def __init__(self, account_number, name, balance=0):
        self.account_number = account_number
        self.name = name
        self.__balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.__balance += amount
            print(f"₹{amount} deposited successfully. New Balance: ₹{self.__balance}")
        else:
            print("Invalid deposit amount!")

    def withdraw(self, amount):
        if amount > 0 and amount <= self.__balance:
            self.__balance -= amount
            print(f"₹{amount} withdrawn successfully. New Balance: ₹{self.__balance}")
        elif amount > self.__balance:
            print("Insufficient funds!")
        else:
            print("Invalid withdrawal amount!")

    def check_balance(self):
        print(f"Current Balance: ₹{self.__balance}")

    def get_details(self):
        return f"Account Holder: {self.name}\nAccount Number: {self.account_number}\nBalance: ₹{self.__balance}"


class ATM:
    def __init__(self):
        self.accounts = {}

    def create_account(self, account_number, name, initial_balance=0):
        if account_number in self.accounts:
            print("Account number already exists!")
        else:
            self.accounts[account_number] = Account(account_number, name, initial_balance)
            print("Account created successfully!")

    def access_account(self, account_number):
        return self.accounts.get(account_number, None)

    def display_menu(self):
        print("\n===== ATM Menu =====")
        print("1. Deposit Money")
        print("2. Withdraw Money")
        print("3. Check Balance")
        print("4. Show Account Details")
        print("5. Exit")
```

```
atm = ATM()
atm.create_account(101, "Sarthak", 5000)
account = atm.access_account(101)

if account:
    while True:
        atm.display_menu()
        choice = int(input("Enter your choice: "))

        if choice == 1:
            amount = float(input("Enter deposit amount: "))
            account.deposit(amount)

        elif choice == 2:
            amount = float(input("Enter withdrawal amount: "))
            account.withdraw(amount)

        elif choice == 3:
            account.check_balance()

        elif choice == 4:
            print(account.get_details())

        elif choice == 5:
            print("Thank you for using the ATM. Goodbye!")
            break

        else:
            print("Invalid choice. Please try again!")
```

```
Account created successfully!

===== ATM Menu =====
1. Deposit Money
2. Withdraw Money
3. Check Balance
4. Show Account Details
5. Exit
Enter your choice:  1
Enter deposit amount:  6000
₹6000.0 deposited successfully. New Balance: ₹11000.0

===== ATM Menu =====
1. Deposit Money
2. Withdraw Money
3. Check Balance
4. Show Account Details
5. Exit
Enter your choice:  2
Enter withdrawal amount:  600
₹600.0 withdrawn successfully. New Balance: ₹10400.0
```

```
===== ATM Menu =====
1. Deposit Money
2. Withdraw Money
3. Check Balance
4. Show Account Details
5. Exit
Enter your choice:  3
Current Balance: ₹10400.0

===== ATM Menu =====
1. Deposit Money
2. Withdraw Money
3. Check Balance
4. Show Account Details
5. Exit
Enter your choice:  4
Account Holder: Sarthak
Account Number: 101
Balance: ₹10400.0

===== ATM Menu =====
1. Deposit Money
2. Withdraw Money
3. Check Balance
4. Show Account Details
5. Exit
Enter your choice:  5
Thank you for using the ATM. Goodbye!
```

10) Python Program to Append, Delete and Display Elements of a List Using Classes.

```python
class ListManager:
    def __init__(self):
        self.elements = []
    def append_element(self, element):
        """Adds an element to the list."""
        self.elements.append(element)
        print(f"Element '{element}' added successfully!")

    def delete_element(self, element):
        """Removes an element from the list if it exists."""
        if element in self.elements:
            self.elements.remove(element)
            print(f"Element '{element}' removed successfully!")
        else:
            print(f"Element '{element}' not found in the list.")

    def display_elements(self):
        """Displays the current elements in the list."""
        if self.elements:
            print("Current List Elements:", self.elements)
        else:
            print("The list is empty.")

lm = ListManager()
```

```
while True:
    print("\n===== List Operations Menu =====")
    print("1. Append Element")
    print("2. Delete Element")
    print("3. Display Elements")
    print("4. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
        element = input("Enter element to append: ")
        lm.append_element(element)

    elif choice == 2:
        element = input("Enter element to delete: ")
        lm.delete_element(element)

    elif choice == 3:
        lm.display_elements()

    elif choice == 4:
        print("Exiting the program. Goodbye!")
        break

    else:
        print("Invalid choice! Please enter a valid option.")
```

OUPUT:-

```
===== List Operations Menu =====
1. Append Element
2. Delete Element
3. Display Elements
4. Exit
Enter your choice:  1
Enter element to append:  10
Element '10' added successfully!

===== List Operations Menu =====
1. Append Element
2. Delete Element
3. Display Elements
4. Exit
Enter your choice:  1
Enter element to append:  20
Element '20' added successfully!

===== List Operations Menu =====
1. Append Element
2. Delete Element
3. Display Elements
4. Exit
Enter your choice:  2
Enter element to delete:  20
Element '20' removed successfully!

===== List Operations Menu =====
1. Append Element
2. Delete Element
3. Display Elements
4. Exit
Enter your choice:  3
Current List Elements: ['10']

===== List Operations Menu =====
1. Append Element
2. Delete Element
3. Display Elements
4. Exit
Enter your choice:  4
Exiting the program. Goodbye!
```