ANSWERS ASSIGNMENT – 8

TOPICS – Class & Object

1. **A Python Program to define Student class and create object to it. Also, we will call the method and display the student's details.**

Solution:

```python
class Student(object):
    firstName = "Snehal"
    studAge = 22
    studMarks = 95

    # Method:
    def displayDetails(self):
        print("My Name is {}".format(self.firstName))
        print("I am {} years old".format(self.studAge))
        print("I have Scores {}".format(self.studMarks))


# Object:
stud = Student()

# 2 type method call:
# on object
stud.displayDetails()

# on className
Student.displayDetails(stud)
```

2. **A Python Program to create Student class with a constructor having more than one Parameter.**

Solution:

```python
class Student(object):

    # Constructor:
    def __init__(self, name, age, marks):
        self.firstName = name
        self.studAge = age
        self.studMarks = marks

    # Method:
    def talk(self):
        print("My Name is {}".format(self.firstName))
        print("I am  {} years old".format(self.studAge))
        print("I have Scores {}".format(self.studMarks))

# Way 1:

# Object:
stud = Student('Shreya', 24, 80)
print(stud.firstName)
print(stud.studAge)
print(stud.studMarks)

# Way 2:
stud = Student('Savi', 21, 90)
# # 2 type method call
# on object
stud.talk()

# on className
Student.talk(stud)

# Way 3:

# using list:
list1 = [Student('Shreya', 24, 80), Student('Sayali', 24, 80), Student
   ('Sam', 24, 80), Student('Sanu', 24, 80), Student('snehu', 24, 80)]
```

```
# Student.talk(list1)

# for i in list1:
#     i.talk()
#     Student.talk(i)

for i in range(len(list1)):
    # Student.talk(list1[i])
    list1[i].talk()
```

3. A Python Program to understand instance variables.

Solution:

```
class Student(object):

    def __init__(self, name='', age=0, marks=0):
        # instance variable
        self.firstName = name
        self.age = age
        self.marks = marks


# Default Constructor:
sayli = Student(1)
print("Name:", sayli.firstName)
print("Age:", sayli.age)
print("Marks:", sayli.marks)

# parameterized constructor:
savi = Student('Savi', 21, 80)
print("Name:", savi.firstName)
print("Age:", savi.age)
print("Marks:", savi.marks)
```

4. **A Python Program to understand class variable or static variable.**

Solution:

```python
class Players(object):
    # static variable
    firstname = 'Virat'
    lastName = 'Kohali'

    def __init__(self, pno, pty):
        # instance variable
        self.playerNum = pno
        self.playerType = pty

    @classmethod
    def setfirstname(cls, nm):
        cls.firstname = nm

a = Players(3, 'Batsman')

# on class:
print(Players.firstname)

# on object:
print(a.firstname)
print(a.lastName)
print(a.playerNum)
print(a.playerType)

# change value of firstName
print(Players.setfirstname('sachin'))
```

5. **A Python Program using a student class with instance methods to process the data of several students.**

Solution:

```python
class Student(object):

    def __init__(self, name="chinmay", marks=90):
        self.firstName = name
        self.marks = marks

    def display(self):
        print('Student Name {}'.format(self.firstName))
        print('Student Marks {}'.format(self.marks))

    def calculateGrade(self):
        if self.marks > 85:
            print("Grade A")
        elif self.marks > 70:
            print("Grade B")
        elif self.marks > 40:
            print("Grade c")
        else:
            print("Fail")

stud = int(input("Please enter the number of student"))
i = 0
studList = []

# Way 1:

# while i < stud:
#     studName = input("Enter Student Name:")
#     studMarks = int(input("Enter Student Marks"))
#     stud1 = Student(studName, studMarks)
#     stud1.display()
#     stud1.calculateGrade()
#     i += 1
# Way 2:
while i < stud:
    studName = input("Enter Student Name:")
    studMarks = int(input("Enter Student Marks"))
    studList.append(Student(studName, studMarks))
    i += 1
for i in studList:
    i.display()
    i.calculateGrade()
```

6. **A Python Program to store data into instances using mutator methods and to retrieve data from the instance using accessor methods.**

Solution:

```python
class Student(object):

    def setNameMarks(self, name, marks):
        self.firstName = name
        self.marks = marks

    def calculateGrade(self):
        if self.marks > 85:
            print("Grade A")
        elif self.marks > 70:
            print("Grade B")
        elif self.marks > 40:
            print("Grade c")
        else:
            print("Fail")

    def display(self):
        print("Student Name{}".format(self.firstName))
        print("Student Marks {}".format(self.marks))

stud = int(input("Please enter the number of student"))
i = 0
```

```python
while i < stud:
    studName = input("Enter Student Name:")
    studMarks = int(input("Enter Student Marks"))
    stud1 = Student()
    stud1.setNameMarks(studName, studMarks)
    stud1.display()
    stud1.calculateGrade()
    i += 1
```

7. **A Python Program to use class method to handle the common feature of all the instances of Bird class.**

Solution:

```python
class Bird(object):
    wings = 2

    def __init__(self, eyes, wing):
        self.eyes = eyes
        # instance variable
        print(self.eyes)
        self.wings = wing

    @classmethod
    def fly(cls, name):
        print("{} bird has two {} wings".format(name, cls.wings))
        cls.wings = 3


bird = Bird(2)
# bird1 = Bird()
# bird.fly("Chinmay")
# bird1.fly("prati")

Bird.fly("chinmay")
Bird.fly("sarika")

# class variable: wings
# instance variable : eyes
```

8. **A Python Program to create a static method that counts the number of instances created for a class.**

Solution:

```python
class Student(object):
    counter = 0

    def __init__(self, name, rollNo):
        self.firstName = name
        self.rollNo = rollNo

        Student.counter += 1

    @staticmethod
    def printNumberObject():
        print(Student.counter)


stud1 = Student('pratiksha', 30)
stud2 = Student('prati', 31)
stud3 = Student('patu', 32)
Student.printNumberObject()
```

9. **A Python Program to create a Bank class where deposits and withdrawals can be handled by using instance method.**

Solution:

```python
class Bank(object):
    def __init__(self, name, bal=0):
        self.name = name
        self.balance = bal

    def deposite(self, amount):
        self.balance += amount
        return self.balance

    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance -= amount
            return self.balance
        else:
            print("Insufficient Balance..")

    def displayBalance(self):
        print("Net Available Balance =", self.balance)

name = input("Enter your name??")
person1 = Bank(name)
person1.deposite(10000)
person1.displayBalance()
person1.withdraw(5000)
person1.displayBalance()
```

10. **A Python Program to create Emp class and make all the members of the Emp class available to another class, i.e. MyClass**

Solution:

```python
class Employee:
    def __init__(self, name, age):
        self.firstName = name
        self.age = age

class MyClass:
    @staticmethod
    def displayDetails(obj):
        print("First Name:", obj.firstName)
        print("Age:", obj.age)

emp1 = Employee('Sarika', 22)
MyClass.displayDetails(emp1)
```

11. **A Python Program to calculate power value of a number with the help of static method.**

Solution:

```python
class Power:

    @staticmethod
    def displayPower(x, y):
        print('Power', x**y)


Power.displayPower(2, 4)
```

12. **A Python Program to create DOB class within Person class.**

Solution:

```python
class Person(object):

    def __init__(self):
        self.firstName = "Gauri"
        self.dob = self.DOB()

    def displayName(self):
        print("FirstName:", self.firstName)

    class DOB:
        def __init__(self):
            self.Date = 9
            self.Month = 8
            self.Year = 1998

        def displayDate(self):
            print("Date Of Birth:{}/{}/{}".format(self.Date,
                          self.Month, self.Year))

name = Person()
name.displayName()
x = name.dob
x.displayDate()
```

13. A Python Program to create another version of DOB class within Person class.

Solution:

```python
class Person(object):

    def __init__(self):
        self.firstName = "Chinmay"
        self.dob = self.DOB()

    def displayName(self):
        print("FirstName:", self.firstName)

    class DOB:
        def __init__(self):
            self.Date = 7
            self.Month = 11
            self.Year = 1990

        def displayDate(self):
            print("Date Of Birth:{}/{}/{}".format(self.Date,
                      self.Month, self.Year))

name = Person()
name.displayName()
x = Person().DOB()
x.displayDate()
```