Lab Assignment 6

In [ ]:
```
#Question No 1:

class Student:
    count = 0

    def __init__(self, name, depart, age, cgpa):
        self.name = name
        self.depart = depart
        self.age = age
        self.cgpa = cgpa
        Student.count = Student.count + 1
        self.count=Student.count

    def  get_details(self):
        print(f"ID: {self.count}\nName: {self.name}\nDepartment: {self.depar

    @classmethod
    def  from_String(cls,vari):
        name, depart, age, cgpa = vari.split("-")
        obj = cls(name, depart, age, cgpa)
        return obj

#============================================================================

s1 = Student("Samin", "CSE", 21, 3.91)
s1.get_details()
print("-----------------------")
s2 = Student("Fahim", "ECE", 21, 3.85)
s2.get_details()
print("-----------------------")
s3 = Student("Tahura", "EEE", 22, 3.01)
s3.get_details()
print("-----------------------")
s4 = Student.from_String("Sumaiya-BBA-23-3.96")
s4.get_details()

print("\nAn instance variable is the variable which is assigned a value insi
print("\nInstance methods require a class instance and can access the instan
ID: 1
Name: Samin
Department: CSE
Age: 21
CGPA: 3.91
-----------------------
ID: 2
Name: Fahim
Department: ECE
Age: 21
CGPA: 3.85
-----------------------
ID: 3
Name: Tahura
Department: EEE
Age: 22
CGPA: 3.01
-----------------------
ID: 4
Name: Sumaiya
Department: BBA
Age: 23
CGPA: 3.96

An instance variable is the variable which is assigned a value inside class methods and depends on the
A class variable is the one in which values are assigned in the class declaration and belongs to the cl

Instance methods require a class instance and can access the instance through self .
Class methods do not require a class instance. They can not access the instance ( self ) but they have
```

In [ ]:

```python
#Question No 2:

class Assassin:
    AssasinCount = 0

    def __init__(self, name, successRate):
        self.name = name
        self.successRate = successRate
        Assassin.AssasinCount = Assassin.AssasinCount + 1

    @classmethod
    def failureRate(cls, name, failureCount):
        return cls(name, 100 - failureCount)
        Assassin.AssasinCount = Assassin.AssasinCount + 1

    @classmethod
    def failurePercentage(cls, name, failurePercentage):
        return cls(name, 100 - failurePercentage)
        Assassin.AssasinCount = Assassin.AssasinCount + 1

    def printDetails(self):
        print(f"Name: {self.name}\nSuccess rate: {self.successRate}%\nTotal 

#============================================================================

john_wick = Assassin('John Wick', 100)
john_wick.printDetails()
print("================================")
nagisa = Assassin.failureRate('Nagisa', 20)
nagisa.printDetails()
print("================================")
akabane = Assassin.failurePercentage('Akabane', 10)
akabane.printDetails()
```

```
Name: John Wick
Success rate: 100%
Total number of Assassin: 1
================================
Name: Nagisa
Success rate: 80%
Total number of Assassin: 2
================================
Name: Akabane
Success rate: 90%
Total number of Assassin: 3
```

In [ ]:

```
#Question No 3:

class Passenger:
    count = 0

    def __init__(self, name):
        self.name = name
        Passenger.count = Passenger.count + 1

    def set_bag_weight(self, weight):
        self.weight = weight

    def printDetail(self):
        if (self.weight > 50):
            print(f"Name: {self.name}\nBus Fare: 550 taka")
        elif (self.weight <= 20):
            print(f"Name: {self.name}\nBus Fare: 450 taka")
        elif (20 < self.weight < 50):
            print(f"Name: {self.name}\nBus Fare: 500 taka")

#============================================================================

print("Total Passenger:", Passenger.count)
p1 = Passenger('Jack')
p1.set_bag_weight(90)
p2 = Passenger('Carol')
p2.set_bag_weight(10)
p3 = Passenger('Mike')
p3.set_bag_weight(25)
print("=========================")
p1.printDetail()
print("=========================")
p2.printDetail()
print("=========================")
p3.printDetail()
print("=========================")
print('Total Passenger:', Passenger.count)
```

```
Total Passenger: 0
=========================
Name: Jack
Bus Fare: 550 taka
=========================
Name: Carol
Bus Fare: 450 taka
=========================
Name: Mike
Bus Fare: 500 taka
=========================
Total Passenger: 3
```

In [ ]:

```python
#Question No 4:

class Travel:
    count = 0

    def __init__(self, source, destination):
        self.__source = source
        self.__destination = destination
        Travel.count = Travel.count + 1
        self.__time = 1

    def set_time(self, time):
        self.__time = time

    def display_travel_info(self):
        return (f"Source: {self.__source}\nDestination: {self.__destination}

    def set_destination(self, destination):
        self.__destination = destination

    def set_source(self, source):
        self.__source = source

#=============================================================================

print('No. of Traveller =', Travel.count)
print("=======================")
t1 = Travel("Dhaka","India")
print(t1.display_travel_info())
print("=======================")
t2 = Travel("Kuala Lampur","Dhaka")
t2.set_time(23)
print(t2.display_travel_info())
print("=======================")
t3 = Travel("Dhaka","New_Zealand")
t3.set_time(15)
t3.set_destination("Germany")
print(t3.display_travel_info())
print("=======================")
t4 = Travel("Dhaka","India")
t4.set_time(9)
t4.set_source("Malaysia")
t4.set_destination("Canada")
print(t4.display_travel_info())
print("=======================")
print('No. of Traveller =', Travel.count)
```

```
No. of Traveller = 0
=======================
Source: Dhaka
Destination: India
Flight Time: 1:00
=======================
Source: Kuala Lampur
Destination: Dhaka
Flight Time: 23:00
=======================
Source: Dhaka
Destination: Germany
Flight Time: 15:00
=======================
Source: Malaysia
Destination: Canada
Flight Time: 9:00
=======================
No. of Traveller = 4
```

In [ ]:
```
#Question No 5:

from datetime import date

class Employee:

    def __init__(self,name, workingPeriod):
        self.name = name
        self.workingPeriod = workingPeriod

    def workingPeriod(self, name, period):
        self.name = name
        self.period = period
        return (self.period)

    @classmethod
    def employeeByJoiningYear(cls, name, y_year):
        vari = date.today().year-y_year
        return cls(name, vari)

    @staticmethod
    def experienceCheck(work_time,gender):
        if gender == "female":
            if work_time < 3:
                return ("She is not experienced")
            else:
                return ("She is experienced")
        else:
            if work_time < 3:
                return ("He is not experienced")
            else:
                return ("He is experienced")

#============================================================================

employee1 = Employee('Dororo', 3)
employee2 = Employee.employeeByJoiningYear('Harry', 2016)
print(employee1.workingPeriod)
```

```python
print(employee2.workingPeriod)
print(employee1.name)
print(employee2.name)
print(Employee.experienceCheck(2, "male"))
print(Employee.experienceCheck(3, "female"))
```

```
3
5
Dororo
Harry
He is not experienced
She is experienced
```

In [ ]:

```
#Question No 6:

class Laptop:
    laptopCount=0

    def __init__(self, name, count):
        self.name = name
        self.count = count
        Laptop.laptopCount = Laptop.laptopCount + self.count

    @classmethod
    def resetCount(vari):
        vari.laptopCount = 0

    @staticmethod
    def advantage():
        print("Laptops are portable")

#============================================================================

lenovo = Laptop("Lenovo", 5)
dell = Laptop("Dell", 7)
print(lenovo.name, lenovo.count)
print(dell.name, dell.count)
print("Total number of Laptops", Laptop.laptopCount)
Laptop.advantage()
Laptop.resetCount()
print("Total number of Laptops", Laptop.laptopCount)
```

```
Lenovo 5
Dell 7
Total number of Laptops 12
Laptops are portable
Total number of Laptops 0
```

In [ ]:

```
#Question No 7:

class Cat:
    Number_of_cats = 0
    color = ""
    action = ""

    def __init__(self, color, action):
        self.color = color
        self.action = action
        Cat.Number_of_cats = Cat.Number_of_cats + 1

    @classmethod
    def no_parameter(cls):
        color = "White"
        action = "sitting"
        return cls(color, action)

    @classmethod
    def first_parameter(cls, color):
        color = color
        action = "sitting"
        return cls(color, action)

    @classmethod
    def second_parameter(cls, action):
        color = "Grey"
        action = action
        return cls(color, action)

    def changeColor(self, color):
        self.color = color

    def printCat(self):
        print(self.color + " cat is " + self.action)

#===========================================================================

```

```python
print("Total number of cats:",Cat.Number_of_cats)
c1 = Cat.no_parameter()
c2 = Cat.first_parameter("Black")
c3 = Cat("Brown", "jumping")
c4 = Cat("Red", "purring")
c5 = Cat.second_parameter("playing")
print("=============================")
c1.printCat()
c2.printCat()
c3.printCat()
c4.printCat()
c5.printCat()
c1.changeColor("Blue")
c3.changeColor("Purple")
c1.printCat()
c3.printCat()
print("=============================")
print("Total number of cats:",Cat.Number_of_cats)
```

```
Total number of cats: 0
=============================
White cat is sitting
Black cat is sitting
Brown cat is jumping
Red cat is purring
Grey cat is playing
Blue cat is sitting
Purple cat is jumping
=============================
Total number of cats: 5
```

In [ ]:

```
#Question No 8:

import math

class Cylinder:
    radius = 5
    height = 18

    def __init__(self, num1, num2):
        self.num1 = num1
        self.num2 = num2
        print(f"Default radius = {Cylinder.radius} and height = {Cylinder.he
        Cylinder.radius = self.num1
        Cylinder.height = self.num2
        print(f"Updated: radius = {self.num1} and height = {self.num2}.")

    @classmethod
    def swap(vari, old, new):
        obj = vari(new, old)
        return obj

    @staticmethod
    def volume(num1, num2):
        print("Volume:", (math.pi*(float(Cylinder.radius)**2)*float(Cylinder

    @staticmethod
    def area(x, y):
        print("Area:", 2 * math.pi * x * x+2 * math.pi * x * y)

    @classmethod
    def changeFormat(vari, info):
        radius, height = info.split("-")
        main = vari(float(radius),float(height))
        return main

#=============================================================================

c1 = Cylinder(0,0)
```

```python
Cylinder.area(c1.radius, c1.height)
Cylinder.volume(c1.radius, c1.height)
print("===============================")
c2 = Cylinder.swap(8,3)
c2.area(c2.radius, c2.height)
c2.volume(c2.radius, c2.height)
print("===============================")
c3 = Cylinder.changeFormat("7-13")
c3.area(c3.radius, c3.height)
c3.volume(c3.radius, c3.height)
print("===============================")
Cylinder(0.3, 5.56).area(Cylinder.radius, Cylinder.height)
print("===============================")
Cylinder(3, 5).volume(Cylinder.radius, Cylinder.height)
```

```
Default radius = 5 and height = 18.
Updated: radius = 0 and height = 0.
Area: 0.0
Volume: 0.0
===============================
Default radius = 0 and height = 0.
Updated: radius = 3 and height = 8.
Area: 207.34511513692635
Volume: 226.1946710584651
===============================
Default radius = 3 and height = 8.
Updated: radius = 7.0 and height = 13.0.
Area: 879.645943005142
Volume: 2001.1945203366981
===============================
Default radius = 7.0 and height = 13.0.
Updated: radius = 0.3 and height = 5.56.
Area: 11.045839770021713
===============================
Default radius = 0.3 and height = 5.56.
Updated: radius = 3 and height = 5.
Volume: 141.3716694115407
```

In [ ]:
```
#Question No 9:

class Student:
    stdCount=0
    bracu=0
    other_institution = 0

    def __init__(self, name, department, universityName=""):
        self.name = name
        self.department = department

        if universityName != "":
            self.universityName = universityName
        else:
            self.universityName = "BRAC University"
        Student.stdCount = Student.stdCount + 1

        if universityName == "":
            Student.bracu = Student.bracu + 1
        else:
            Student.other_institution = Student.other_institution + 1

    def individualDetail(self):
        print(f"Name: {self.name}\nDepartment: {self.department}\nInstitutio

    @classmethod
    def printDetails(cls):
        print(f"Total Student(s): {Student.stdCount}\nBRAC University Studen

    @classmethod
    def createStudent(cls, name, department, universityName = ""):
        if universityName != "":
            cls.universityName=universityName
        else:
            cls.universityName = "BRAC University"
        obj = cls(name, department, universityName)
        return obj

```

```python
#===========================================================================

Student.printDetails()
print('#########################')
mikasa = Student('Mikasa Ackerman', "CSE")
mikasa.individualDetail()
print('------------------------------------------')
Student.printDetails()
print('========================')
harry = Student.createStudent('Harry Potter', "Defence Against Dark Arts", "
harry.individualDetail()
print('-------------------------------------------')
Student.printDetails()
print('=========================')
levi = Student.createStudent("Levi Ackerman", "CSE")
levi.individualDetail()
print('--------------------------------------------')
Student.printDetails()
```

```
Total Student(s): 0
BRAC University Student(s): 0
Other Institution Student(s): 0
#########################
Name: Mikasa Ackerman
Department: CSE
Institution: BRAC University
------------------------------------------
Total Student(s): 1
BRAC University Student(s): 1
Other Institution Student(s): 0
========================
Name: Harry Potter
Department: Defence Against Dark Arts
Institution: Hogwarts School
-------------------------------------------
Total Student(s): 2
BRAC University Student(s): 1
Other Institution Student(s): 1
=========================
Name: Levi Ackerman
Department: CSE
Institution: BRAC University
--------------------------------------------
Total Student(s): 3
BRAC University Student(s): 2
Other Institution Student(s): 1
```

In [ ]:

```python
#Question No 10:

class SultansDine:
    branchNum = 0
    all_sell = 0
    lst= []

    def __init__(self, location):
        self.location = location
        SultansDine.branchNum = SultansDine.branchNum + 1
        SultansDine.lst.append(location)

    def sellQuantity(self,quantity):
        if quantity < 10:
            self.quantity  = quantity * 300
        elif quantity < 20:
            self.quantity = quantity * 350
        else:
            self.quantity = quantity * 400

        self.b_sell = self.quantity
        SultansDine.lst.append(self.b_sell)
        SultansDine.all_sell = SultansDine.all_sell + self.b_sell
        self.persent = (self.b_sell/SultansDine.all_sell) * 100

    def branchInformation(self):
        print(f"Branch Name: {self.location}")
        print(f"Branch Sell: {self.b_sell} Taka")

    @classmethod
    def details(cls):
        print(f"Total Number of branch(s): {SultansDine.branchNum}")
        print(f"Total Sell: {SultansDine.all_sell} Taka")
        var = len(SultansDine.lst)

        for i in range(0, var, 2):
            persent = (SultansDine.lst[i+1]/SultansDine.all_sell) * 100

            new_sell_per =(round(persent, 2))
            print(f"Branch Name: {SultansDine.lst[i]}, Branch Sell: {SultansI
            print(f"Branch consists of total b_sell's: {new_sell_per:.2f}%")

#============================================================================

SultansDine.details()
print('########################')
dhanmodi = SultansDine('Dhanmondi')
dhanmodi.sellQuantity(25)
dhanmodi.branchInformation()
print('-----------------------------------------')
SultansDine.details()
print('=======================')
baily_road = SultansDine('Baily Road')
baily_road.sellQuantity(15)
baily_road.branchInformation()
print('-----------------------------------------')
SultansDine.details()
print('=======================')
gulshan = SultansDine('Gulshan')
gulshan.sellQuantity(9)
gulshan.branchInformation()
print('-----------------------------------------')
SultansDine.details()
```

```
Total Number of branch(s): 0
Total Sell: 0 Taka
########################
Branch Name: Dhanmondi
Branch Sell: 10000 Taka
-----------------------------------------
Total Number of branch(s): 1
Total Sell: 10000 Taka
Branch Name: Dhanmondi, Branch Sell: 10000 Taka
Branch consists of total b_sell's: 100.00%
=======================
Branch Name: Baily Road
Branch Sell: 5250 Taka
-----------------------------------------
Total Number of branch(s): 2
Total Sell: 15250 Taka
Branch Name: Dhanmondi, Branch Sell: 10000 Taka
Branch consists of total b_sell's: 65.57%
Branch Name: Baily Road, Branch Sell: 5250 Taka
Branch consists of total b_sell's: 34.43%
========================
Branch Name: Gulshan
Branch Sell: 2700 Taka
-----------------------------------------
Total Number of branch(s): 3
Total Sell: 17950 Taka
Branch Name: Dhanmondi, Branch Sell: 10000 Taka
Branch consists of total b_sell's: 55.71%
Branch Name: Baily Road, Branch Sell: 5250 Taka
Branch consists of total b_sell's: 29.25%
Branch Name: Gulshan, Branch Sell: 2700 Taka
Branch consists of total b_sell's: 15.04%
```