Assignment 06_20101197_AbirAhammedBhuiyan

October 3, 2022

[11]:
```python
#task_1

class Student:
    ID = 0
    def __init__(self, name, dept, age, cgpa):
        self.name = name
        self.dept = dept
        self.age = age
        self.cgpa = cgpa
        Student.ID+=1

    def get_details(self):
        print("ID:", Student.ID)
        print("Name:", self.name)
        print("Department:", self.dept)
        print("Age:", self.age)
        print("CGPA:", self.cgpa)

    @classmethod
    def from_String(cls, text):
        name, dept, age, cgpa = text.split('-')
        return cls(name, dept, age, cgpa)


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("\n")
print("Class Variables are variables that are shared by all instances of a␣
  ↪class. So while instance variable can be unique for each instance like our␣
  ↪name,email and pay; class variable should be the-same for each instance.")
print("\n")
print("Class methods have the ability to change the class state while Instance␣
  ↪methods have the ability to change state of each instance or object.")
```

```
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


Class Variables are variables that are shared by all instances of a class. So
while instance variable can be unique for each instance like our name,email and
pay; class variable should be the-same for each instance.


Class methods have the ability to change the class state while Instance methods
have the ability to change state of each instance or object.
```

[2]:
```
#task_2

class Assassin:

    num_of_assa = 0

    def __init__(self, name, sucrate):
        self.name = name
        self.sucrate = sucrate
        Assassin.num_of_assa+=1



    @classmethod
    def failureRate(cls, name, rate):
        return cls(name, 100-rate)


    @classmethod
    def failurePercentage(cls, name, rate):
        return cls(name, 100-rate)

    def printDetails(self):
        print("Name:", self.name)
        print("Success rate:"+str(self.sucrate)+"%")
        print("Total number of Assassin:", Assassin.num_of_assa)


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
```

[3]:
```
#task_3

class Passenger:
    count = 0
    base_fare = 450
    def __init__(self, name):
        self.name = name
        Passenger.count+=1

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

    def printDetail(self):
        print("Name:", self.name)

        if(self.bag_weight>=21 and self.bag_weight<=50):
            print("Bus Fare:", Passenger.base_fare+50, "taka")
        elif(self.bag_weight>=50):
            print("Bus Fare:", Passenger.base_fare+100, "taka")
        else:
            print("Bus Fare:", Passenger.base_fare, "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
```

[4]:
```python
#task_4

class Travel:
    count = 0

    def __init__(self, source, dest):
        self.source = source
        self.dest = dest
        self.time = 1
        Travel.count+=1


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

    def set_destination(self, dest):
        self.dest = dest

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


    def display_travel_info(self):
        return f"Source: {self.source}\nDestination: {self.dest}\nFight Time:␣
 ↪{self.time}:00"


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

[5]:
```
#task_5

class Employee:
    current_year = 2021
    def __init__(self, name, wp):
        self.name = name
        self.workingPeriod = wp


    @classmethod
    def employeeByJoiningYear(cls, name, year):
        wp = Employee.current_year - year
        return cls(name, wp)

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

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

[6]:
```
#task_6

class Laptop:

    laptopCount = 0

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

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

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

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
```

```
[7]: #task_7

class Cat:
    Number_of_cats = 0
    def __init__(self, color, verb):
        self.color = color
        self.verb = verb
        Cat.Number_of_cats+=1

    @classmethod
    def no_parameter(cls):
        return cls('White', 'sitting')

    @classmethod
    def first_parameter(cls, color):
        return cls(color, 'sitting')

    @classmethod
    def second_parameter(cls, verb):
        return cls('Grey', verb)

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

    def printCat(self):
        print(f"{self.color} cat is {self.verb}")

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
```

[8]:
```
#task_8

import math

class Cylinder:

    radius = 5
    height = 18


    def __init__(self, r, h):
        self.radius = r
        self.height = h

        print(f"Default radius={Cylinder.radius} and height={Cylinder.
 ↪height}\nUpdated: radius={self.radius} and height={self.height}")

        Cylinder.radius = self.radius
        Cylinder.height = self.height


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

    @staticmethod
    def volume(r, h):
        print("Volume: ", math.pi*r*r*h)

    @classmethod
    def swap(cls, h, r):
        return cls(r, h)
    @classmethod
    def changeFormat(cls, text):
        r, h = text.split("-")
        return cls(float(r), float(h))

c1 = Cylinder(0,0)
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:  11.045839770021711
================================
Default radius=0.3 and height=5.56
Updated: radius=3 and height=5
Volume:  141.3716694115407
```

[9]:
```
#task_9
```

```python
class Student:
    total_student = 0
    bracu_student = 0
    other_student = 0


    def __init__(self, name, dept, uni_name="BRAC University"):
        self.name = name
        self.dept = dept
        self.institution = uni_name
        Student.total_student+=1

        if(self.institution == "BRAC University"):
            Student.bracu_student+=1
        else:
            Student.other_student+=1

    def individualDetail(self):
        print("Name:", self.name)
        print("Department:", self.dept)
        print("Institution:", self.institution)

    @classmethod
    def printDetails(cls):
        print("Total Student(s):", cls.total_student)
        print("BRAC University Student(s):", cls.bracu_student)
        print("Other Institution Student(s):", cls.other_student)


    @classmethod
    def createStudent(cls, name, dept, uni_name="BRAC University"):
        return cls(name, dept, uni_name)

Student.printDetails()
print('#########################')
mikasa = Student('Mikasa Ackerman', "CSE")
mikasa.individualDetail()
print('------------------------------------------')
Student.printDetails()

print('========================')

harry = Student.createStudent('Harry Potter', "Defence Against Dark Arts",␣
 ↪"Hogwarts School")
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
```

[10]:
```
#task_10

class SultansDine:
    num_of_branches = 0
    total_sell = 0
    branches_info = []

    def __init__(self, name):
        self.name = name
        SultansDine.num_of_branches+=1
        SultansDine.branches_info.append(self)
    def sellQuantity(self, qnt):
        self.qnt = qnt

        if(self.qnt < 10):
            self.branch_sell = self.qnt * 300
        elif(self.qnt < 20):
            self.branch_sell = self.qnt * 350
        else:
            self.branch_sell = self.qnt * 400

        SultansDine.total_sell+=self.branch_sell

    def branchInformation(self):
        print("Branch Name:", self.name)
        print("Branch Sell: "+str(self.branch_sell)+" Taka")


    @classmethod
    def details(cls):
        print("Total Number of branch(s):", cls.num_of_branches)
        print(f"Total Sell: {cls.total_sell} Taka")

        for i in cls.branches_info:
            print(f"Branch Name: {i.name}, Branch Sell: {i.branch_sell} Taka")
            print(f"Branch consists of total sell's: {format((i.branch_sell/cls.
 ↪total_sell)*100, '.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 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 sell's: 65.57%
Branch Name: Baily Road, Branch Sell: 5250 Taka
Branch consists of total 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 sell's: 55.71%
Branch Name: Baily Road, Branch Sell: 5250 Taka
Branch consists of total sell's: 29.25%
Branch Name: Gulshan, Branch Sell: 2700 Taka
Branch consists of total sell's: 15.04%
```

```
[ ]:
```