10.

Class & Objects

Page No.
Date

Solve a problem by Creating objects is one of the most popular approaches in programming. This is called object oriented programming.

This Concept focuses on using Reusable Code.
↳ Implemented DRY principle.

Class

A Class is a blueprint for Creating objects.

Contains info to Create a Valid Application	Blank form	→ filled by an student →	Application of the Student

class → Object instantiation → object

Contain info to Create a valid object.

Page No.
Date

The syntax of class look like this:

```
Class Employee:
    # methods & variables
```

ClassName is written in pascalcase

Object

An object is an instantiation of a class. When class is defined, a template (info) is defined. Memory is allocated only after object instantiation.

Object of given class can invokes the method available to it without revealing the implementation details to user.

↳ Abstraction & Encapsulation!

Modelling a problem in OOPs

We identify the following in our program

Noun ⟶ class ⟶ Employee
Adjective ⟶ Attributes ⟶ Name, age, salary.
Verbs ⟶ Methods ⟶ getSalary().

Page No.
Date

Class Attributes:

An attribute that belongs to the class rather than a particular object.

Example:

```
Class Employee
        Company = "Google"
```

↳ [specific to each class]

```
Tejas = Employee() ⟶ object instantiation.
Tejas.Company
Employee.Company = "youtube" ⟶ Change class attributes.
```

Instance Attributes

An attribute that belongs to the instance (object) assuming the class from the previous example.

```
Tejas.name = "Tejas"
Tejas.salary = "900k"  ⇒ Adding instance Attributes.
```

Page No.
Date

Note: Instance attributes take preference over class attributes during assignment & retrival.

Tejas Attributes →

1] Is attribute present in object 1.

2] Is attribute present in class.

'Self parameter'

Self refers to instance of the class it is automatically passed with a function call from an object.

Tejas.getSalary() → here Self is Tejas

↳ equivalent to employee.getSalary(Tejas)

The function getSalary is defined as:

```
Class Employee:
    Company = "Google"
    def getsalary(self):
        print("salary is not there")
```

Static Method

Sometime we need a function that doesn't use the self parameter. We can define a static method like this.

@staticmethod → decorator to make greet as a static method.

```
def greet():
    print("Hello Tejas")
```

__init__() Constructor

__init__() is a special method which is first run as soon as the object is created.

__init__() method is also known as constructor

It takes self arguments & can also take further arguments.

For Example:

```
Class Employee
        def __init__(self, name):
                self.name = name

        def getSalary(self):
                ....
Tejas = Employee("Tejas")
```

↳ object can be instantiated using Constructor like this.

Q10.

Practice Set

1] Create a class programmer for storing information of few programmer working at microsoft.

2] Write a class Calculator capable of finding square cable & squareroot of a number.

3] Create a class with a class attributes a; Create an object from it & set a directly using object. a = 0. Does this change the class attributes.

4] Add a static method in problem 2 to greet the user with hello.

5] Write a class Train which has method to book a tickets. get status (no of seats) & get fare information of trains running under Indian railway.

6] Can you change the self parameter inside a class to something else (say 'Tejas'). Try changing self to 'slf' or 'Tejas' & see the effects.