URBAN EDGE

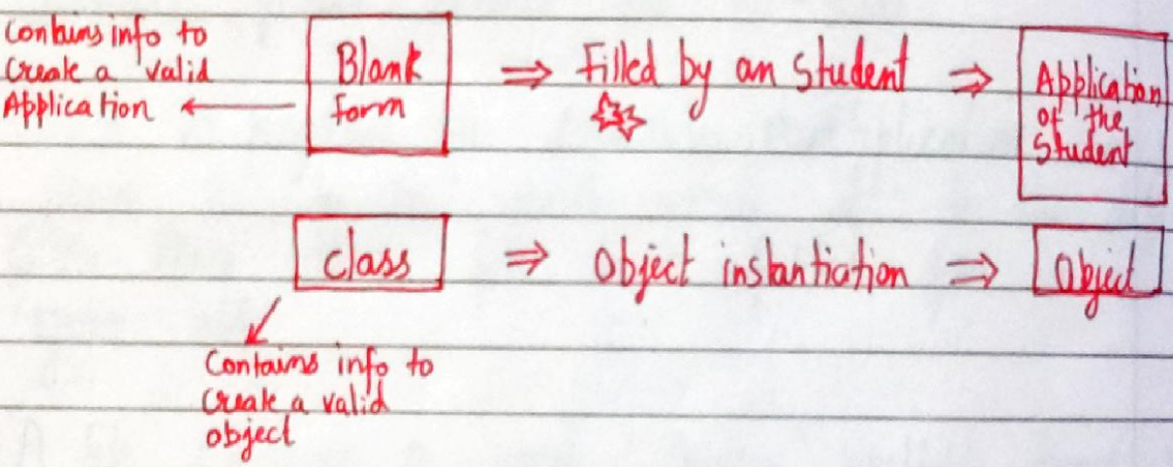
Chapter 10 - Object Oriented Programming

Solving 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.
↳ Implements DRY principle

Class

A class is a blueprint for creating objects.



The syntax of a class looks 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.

Objects of a given class can invoke the methods available to it without revealing the implementation details to the user. → Abstraction & Encapsulation!

URBAN EDGE

Modelling a problem in OOPs

We identify the following in our problem

Noun → Class → Employee
Adjective → Attributes → name, age, salary
Verbs → Methods → getSalary(), increment()

Class Attributes

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

Example :

```
Class Employee :
    company = "Google"      → [Specific to each class]

harry = Employee()          → object instantiation
harry.company
Employee.company = "YouTube" → changing class attribute
```

Instance Attributes

An attribute that belongs to the Instance (object)
Assuming the class from the previous example :

```
harry.name = "Harry"
harry.salary = "30K"        ⇒ Adding instance attributes
```

Note : Instance attributes take preference over class attributes during assignment & retrieval

harry.attribute1 → ① Is attribute1 present in object ?
② Is attribute1 present in class ?

'self' parameter
self refers to the instance of the class
It is automatically passed with a function call from an object

```
harry.getSalary()
```
→ here self is harry
↳ equivalent to Employee.getSalary(harry)

The function getsalary is defined as:

```
class Employee:
    Company = "Google"
    def getSalary(self):
        print("Salary is not there")
```

Static method
Sometimes we need a function that doesnt use the self parameter. we can define a static method like this:

```
@staticmethod
def greet():
    print("Hello user")
```
→ decorator to mark greet as a static method

__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 argument and can also take further arguments

URBAN EDGE ___/___/___

For Example:

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

    def getSalary(self):
        ...

harry = Employee("Harry")
```

↳ Object can be instantiated using constructor like this!

URBAN EDGE

Chapter 10 - Practice Set

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

2. Write a class Calculator capable of finding square, cube and squareroot of a number.

3. Create a class with a class attribute a; create an object from it and set a directly using object.a = 0. Does this change the class attribute?

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

5. Write a class Train which has methods to book a ticket, get status (no of seats) and get fare information of trains running under Indian Railways.

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