Chapter 8

OOP

OOP in Python

To map with real world scenarios, we started using objects in code.

This is called object oriented programming.

Class & Object in Python

Class is a blueprint for creating objects.

```
#creating class

class Student:
        name =“Ariful Islam”

#creating object (instance)

s1 = Student( )
print( s1.name )
```

_ _init_ _ Function

Constructor
All classes have a function called __init__(), which is always executed when the object is being initiated.

```
#creating class

class Student:
        def __init__( self, fullname ):
                self.name = fullname
```

```
#creating object
s1 = Student( “karan” )
print(s1.name)
```

*The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

Class & Instance Attributes

Class.attr
obj.attr

Methods

Methods are functions that belong to objects.

```
#creating class
class Student:
    def __init__( self, fullname ):
        self.name = fullname

def hello( self ):
    print( “hello”, self.name)
```

```
#creating object

s1 = Student( “karan” )
s1.hello( )
```

Let‘s Practice

Create a Student class that:

- Takes a student's name and marks for 3 subjects as arguments in the constructor.
- Has a method calculate_average() that calculates and prints the average marks of the student.

Static Methods

Methods that don’t use the self parameter (work at class level)

```
class Student:

    @staticmethod       #decorator
    def college( ):
        print( “ABC College”)
```

*Decorators allow us to wrap another function in order to extend the behaviour of the wrapped function, withoutpermanently modifying it

Important

Abstraction

Hiding the implementation details of a class and only showing the essential features to the user.

Page 151

Important

Encapsulation

Wrapping data and functions into a single unit (object).

Let‘s Practice

Create Account class with 2 attributes - balance & account no.
Create methods for debit, credit & printing the balance.
Create an Account class that:

- Has two attributes: balance and account_no (both initialized through the constructor).
- Has methods:
  - credit(amount): Adds money to the balance.
  - debit(amount): Deducts money if sufficient balance is available, otherwise prints "Insufficient balance".
  - print_balance(): Prints the current account balance.

del keyword

Used to delete object properties or object itself

```
del s1.name
del s1
```

Inheritance

When one class(child/derived) derives the properties & methods of another class (parent/base).

```
class Car:
        ...

class ToyotaCar(Car):
        ...
```

Inheritance

Types

- Single Inheritance
- Multi_level Inheritance
- Multiple Inheritance

Super method

super() method is used to access methods of the parent class.

Class method

A Class method is bound to the class & receives the class as an implicit first argument.

Note – Static method can't access or modify class state & generally for utility

```
Class Student:
    @classmethod # decorator
    def college(cls):
        pass
```

Polymorphism : Operator Overloading

When the same operator is allowed to have different meaning according to the context.

Operators & dunder functions

a+b	#addition	a.__add__(b)
a-b	#subtraction	a.__sub__(b)
a*b	#multiplication	a.__mul____(b)
a/b	#division	a.__truediv____(b)
a%b	#modulus	a.__mod____(b)

Let’s Practice

Qs. Define a Circle class to create a circle with radius r using the constructor.

Define an Area() method of the class which calculates the area of the circle

Define a Perimeter() method of the class which allows you to calculate the perimeter of the circle

Let’s Practice

Qs. Define a Employee class with attributes role, department & salary, this class also a showDetails() method.

Create an Engineer class that inherits properties from employee & has additional attributes: name & age

Let’s Practice

Qs. Create a class called Order which stores item & its price.
Use Dunder function __gt__() to convey that:
order1>order2 if price of oder1>price of oder2