URBAN EDGE

Chapter 11 - Inheritance & more on OOPs

Inheritance is a way of creating a new class from an existing class

Syntax:

```
Class Employee:                    → Base Class
    # Code
    ...

Class Programmer (Employee):       → Derived or child class
    # Code
```

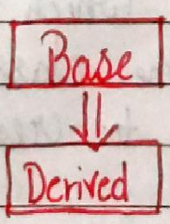
We can use the methods and attributes of Employee in Programmer object.
Also, we can overwrite or add new attributes and methods in Programmer class.

Types of Inheritance

1. Single inheritance
2. Multiple inheritance
3. Multilevel inheritance

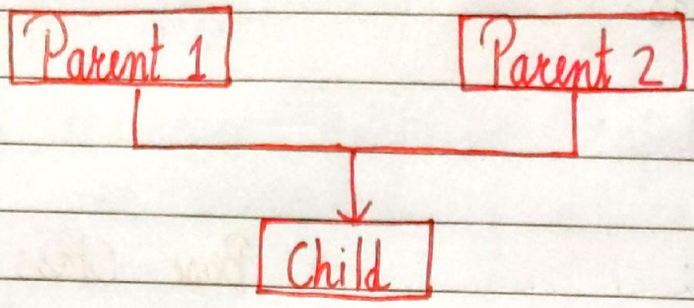
Single Inheritance

Single inheritance occurs when child class inherits only a single parent class



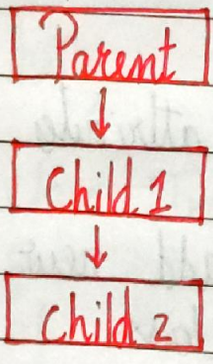
Multiple Inheritance

Multiple inheritance occurs when the child class inherits from more than one parent class.



Multilevel Inheritance

When a child class becomes a parent for another child class



Super() method

Super method is used to access the methods of a super class in the derived class

```
super().__init__()
```

↳ Calls constructor of the base class

Class methods

A class method is a method which is bound to the class and not the object of the class.

@classmethod decorator is used to create a class method

Syntax to create a class method:

```
@classmethod
def (cls, p1, p2):
    ...
```

@property decorators
Consider the following class

```
class Employee:
    @property
    def name(self):
        return self.ename
```

if e = Employee() is an object of class employee, we can print(e.name) to print the ename/call name() function.

@.getters and @.setters
The method name with @property decorator is called getter method
We can define a function + @name.setter decorator like below:

```
@name.setter
def name(self, value):
    self.ename = value
```

Operator overloading in Python
Operators in python can be overloaded using dunder methods.
These methods are called when a given operator is used on the objects.

Operators in python can be overloaded using the following methods:

$p_1 + p_2 \rightarrow p_1.\_\_add\_\_(p_2)$

$p_1 - p_2 \rightarrow p_1.\_\_sub\_\_(p_2)$

$p_1 * p_2 \rightarrow p_1.\_\_mul\_\_(p_2)$

$p_1 / p_2 \rightarrow p_1.\_\_truediv\_\_(p_2)$

$p_1 // p_2 \rightarrow p_1.\_\_floordiv\_\_(p_2)$

Other dunder/magic methods in python

`__str__()` → used to set what gets displayed upon calling `str(obj)`

`__len__()` → used to set what gets displayed upon calling `.__len__()` or `len(obj)`

URBAN EDGE

Chapter 11 - Practice Set

1. Create a class C2dVector and use it to create another class representing a 3-d vector.

2. Create a class pets from a class Animals and further create class Dog from Pets. Add a method bark to class Dog.

3. Create a class Employee and add salary and increment properties to it.
Write a method salaryAfterIncrement method with a @property decorator with a setter which changes the value of increment based on the salary.

4. Write a class Complex to represent complex numbers, along with overloaded operators + and * which adds and multiplies them

5. Write a class vector representing a vector of n dimension. Overload the + and * operator which calculates the sum and the dot product of them.

6. Write __str__() method to print the vector as follows:

$$7\hat{i} + 8\hat{j} + 10\hat{k}$$

Assume vector of dimension 3 for this problem.

URBAN EDGE

7 Override the __len__() method on Vector of problem 5 to display the dimension of the Vector.

URBAN EDGE

Project 2 - The Perfect Guess

We are going to write a program that generates a random number and asks the user to guess it.
If the player's guess is higher than the actual number, the program displays "Lower number please".
Similarly if the user's guess is too low, the program prints "higher number please"
When the user guesses the correct number, the program displays the number of guesses the player used to arrive at the number.

Hint: Use the random module