1. Object-oriented programming is a programming paradigm that provides a means of structuring programs so that properties and behaviours are bundled into individual objects. It uses objects and classes in the programming. It also contains some real world entities like inheritance, polymorphism and encapsulation. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.

2. Python searches for an attribute in an upward tree of attributes. It first searches for the attribute in its instance and then looks in the class it is generated from, to all super classes listed in its class header.

3. Class is a template for creating an object and object is an instance of class. A Class is created once. Many objects are created using a class. Class cannot be manipulated because they don’t have allocated memory but objects can be manipulated.

4. Instance Methods

* Class Methods
* Static Methods
* **self** is the first argument for instance methods which refers to the object itself.
* **cls** is the first argument for class methods which refers to the class itself.

**5. \_\_init\_\_ is a reseved method in python classes. It serves the role of a constructor in object oriented terminology. This method is called when an object is created from a class and it allows the class to initialize the attributes of the class.**

**6. To create a class instance, we need to call the class by its name and pass the arguments to the class, which its init method accepts.**

**Example: my\_name = my\_class("Kunal","Ishank") Here my\_name is an instance of class my\_class with attributes "Kunal" and "Ishank".**

**7. class keyword is used to create a class in python. The syntax to create a class in python is class <classname>.**

**Example: class Name: ➞ this creates a class called Name.**

**8. Superclass/Parent class is given as an argument to the child class.**

**Example: class Family(Person): Here child class Family inherits attributes and methods from Superclass/Parent Person.**