OOPs Definition

Definition

Python supports both procedure-oriented and object-oriented features.
Object-Oriented Programming (OOPs) is a programming paradigm that uses classes and objects to look similar to real life. OOPs has four central concepts: Encapsulation, Abstraction, Inheritance, and Polymorphism. OOPs mainly works upon two keys is Attributes and Functionality.

Classes

Classes consist of Data and Functions. The Data in class is known as Members and functions of the class are known as methods.

Syntax:

```
Class className:
    Statement1_Class
    Statement2_Class
    Statement3_Class
    Statementn_Class
```

Example:

```
Class student:
    No = 100
    Sub = “Maths”
Print(student.No, student.Sub)
```

Objects

Using Python Programming, we can create objects in two ways that are as follows:

- Attribute Referencing
- Instantiation

Attribute Referencing

In this method, class properties can be accessed using the class name itself.

Syntax:

className:propertyName

Example:

```
Class student;
        No_of_admission = []
student.No_of_admission = 100
print(student.No_of_admission)
```

Instantiation

While Instantiation, an empty object is created, and this object is made, and this object is assigned with the given object name.

Syntax:

Object_name:class_name()

After creating the class instance, the properties of the class are accessed using following

Object_name:propertyname

Example:

```
Class student;
        No_of_admission = []
```

Calling.student();

Calling.No_of_admission = 100

Print(Calling.No_of_admission)

Encapsulation

Encapsulation means wrapping data and the functions in a single place.

Data Abstraction

The data abstraction is the process by which only a few necessary features are highlighted while the internal details are suppressed. For example, while using mobile, the user uses all the features without knowing the internal hardware and software. Thus Data Abstraction reduces the complexity associated with data encapsulation.

Polymorphism

Using polymorphism, a single entity can act in multiple forms. For example, a shape can be both a circle and a triangle.

Inheritance

Inheritance allows a class to derive the properties of another class. For example class1 has data1 & function 1 where class 2 has data1 & function 1 of class 1 and class3 has data1 and functions 1 of class 1.