OOPS CONCEPT

-By Ch.Siri chandana

INTRODUCTION

In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. 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.

OOPS CONCEPTS IN PYTHON

- Objects
- Class
•Class
- Polymorphism
- Encapsulation
- Inheritance
- Data Abstraction

CLASS :

- A class is a collection of objects.
- A class contains the blueprints or the prototype from which the objects are being created.
- It is a logical entity that contains some attributes and methods.
- Classes are created by keyword class.
- Attributes are the variables that belong to a class.
- Attributes are always public and can be accessed using the dot (.) operator.
- Eg.: Myclass.Myattribute

CLASS DEFINITION SYNTAX:

```
class ClassName:
# Statement-1
.
.
.
# Statement-N
```

OBJECTS

- The object is an entity that has a state and behavior associated with it.
- It may be any real-world object like a mouse, keyboard, chair, table, pen, etc. Integers, strings, floating-point numbers, even arrays, and dictionaries, are all objects.
- More specifically, any single integer or any single string is an object.
- The number 12 is an object, the string “Hello, world” is an object, a list is an object that can hold other objects, and so on.

An object consists of:

- **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
- **Behavior:** It is represented by the methods of an object. It also reflects the response of an object to other objects.
- **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

INHERITANCE

- Inheritance is the capability of one class to derive or inherit the properties from another class.
- Inheritance is a way of creating a new class for using details of an existing class without modifying it.
- The newly formed class is a derived class (or child class). Similarly, the existing class is a base class (or parent class).

```
# base class
class Animal:

    def eat(self):
        print( "I can eat!")

    def sleep(self):
        print("I can sleep!")

# derived class
class Dog(Animal):

    def bark(self):
        print("I can bark! Woof woof!!")

# Create object of the Dog class
dog1 = Dog()

# Calling members of the base class
dog1.eat()
dog1.sleep()

# Calling member of the derived class
dog1.bark();
```

ENCAPSULATION

- Encapsulation is one of the key features of object-oriented programming.
- Encapsulation refers to the bundling of attributes and methods inside a single class.
- It prevents outer classes from accessing and changing attributes and methods of a class. This also helps to achieve **data hiding**.
- Encapsulation involves bundling the data (attributes) and methods (functions) that operate on the data within a single unit, i.e., a class.
- This concept restricts direct access to the internal details of an object, promoting information hiding and protecting the integrity of the data.
- Encapsulation enhances code maintainability and security by establishing well-defined interfaces for interacting with objects.

POLYMORPHISM

- Polymorphism is another important concept of object-oriented programming. It simply means more than one form.
- That is, the same entity (method or operator or object) can perform different operations in different scenarios.
- In OOP, polymorphism is achieved through method overloading and method overriding, providing a means for objects to exhibit different behaviors based on their specific types or contexts.

ABSTRACTION

- Abstraction is the process of simplifying complex systems by focusing on essential features while ignoring unnecessary details.
- In OOP, abstraction involves creating abstract classes and interfaces that define a common set of methods without specifying their implementation.
- It allows developers to work with high-level concepts, promoting a clearer understanding of the software's structure and functionality.