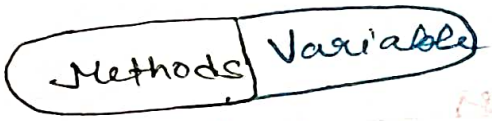
What are the main principles of OOP?

Encapsulation – This principle states that all important information is contained inside an object and only select information is exposed.

- The implementation and state of each object are privately held inside a defined class. Other objects do not have access to this class or the authority to make changes.
- They are only able to call a list of public functions or methods. This characteristic of data hiding provides greater program security and avoids unintended data corruption

A class is an example of encapsulation as it encapsulates all the data that is member functions, variables, etc.



Abstraction – Abstraction is a fundamental concept in Python programming that allows us to simplify complex concepts and focus on the essential details. It involves hiding unnecessary details and exposing only the relevant information to the users.

Inheritance – Inheritance allows us to define a class that inherits all the methods and properties from another class.

Parent class - Parent class is the class being inherited from, also called base class.

Child class - Child class is the class that inherits from another class, also called derived class.

Polymorphism - Polymorphism in Python is the ability of one object to take on multiple forms. This is done by creating multiple classes inherited from single base class. Each class can then be used interchangeably as they all share the same interface. This allows for a great degree of flexibility when it comes to programming.

Class and Objects

Class

A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together.

Syntax

```
class ClassName:
    # statement
```

Creating a Python Class

```
class Dog:          # ClassName
    sound = "bark"
```

Object

An object is an instance of a class. A class is like a blueprint while an instance is a copy of the class with actual values. It's not an idea anymore, it's an actual dog, like a dog of breed pug who's seven years old.

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.

Example of Python Class and Object

```
Class Dog:
    # A simple class
    # attribute
    attr1 = "mammal"
    attr2 = "dog"
    # A sample method
    def fun(self):
        print("I'm a", self.attr1)
        print("I'm a", self.attr2)

# Driver code
# Object instantiation
Rodger = Dog()

# Accessing class attributes
# and method through objects
print(Rodger.attr1)
Rodger.fun()
```

Output

```
mammal
I'm a mammal
I'm a dog.
```