Python

**1. \*args and \*\*kwargs:**

* **\*args (Positional Arguments)**:
  + \*args allows a function to accept any number of positional arguments. The arguments are passed as a tuple to the function.
* **\*\*kwargs (Keyword Arguments)**:
  + \*\*kwargs allows a function to accept any number of keyword arguments. The arguments are passed as a dictionary to the function.
* **Positional Arguments**:
  + These are arguments that are passed to a function in the correct positional order.
* **Keyword Arguments**:
  + These are arguments passed by explicitly stating the parameter name when calling the function.
* **Default Arguments**:
  + These are arguments that have a default value. If no value is passed during the function call, the default value is used.

**2. Lambda Functions:**

* **lambda**:
  + A lambda function is a small anonymous function that can have any number of arguments but only one expression. It’s typically used for short, throwaway functions.
  + Syntax: lambda arguments: expression
* **filter()**:
  + The filter() function is used to filter elements of an iterable (like a list) based on a condition (a function that returns True or False). It returns a filter object (which can be converted to a list).
* **map()**:
  + The map() function applies a function to all items in an iterable (like a list) and returns a map object (which can be converted to a list).
* **reduce()**:
  + The reduce() function (from functools module) applies a function cumulatively to the items of an iterable, from left to right, so as to reduce the iterable to a single value.

**3. Classes and Object-Oriented Concepts:**

* **Class**:
  + A class is a blueprint for creating objects (instances). It can have methods (functions) and attributes (variables).
* **Inheritance**:
  + Inheritance allows one class to inherit the attributes and methods of another class.
  + **Single Inheritance**: A class inherits from one parent class.
  + **Multilevel Inheritance**: A class inherits from a class that is itself derived from another class.
* **Polymorphism**:
  + Polymorphism allows different classes to implement the same method in their own way. This can be achieved through method overriding (in inheritance) or method overloading.
  + **Method Overriding** (In inheritance):

**4. Encapsulation:**

* **Encapsulation** is the bundling of data (attributes) and methods (functions) that operate on the data into a single unit, called a class. It also involves restricting access to certain attributes or methods by using **private** and **protected** access specifiers.
* **Private Members**: Denoted by a double underscore (\_\_). These can't be accessed directly from outside the class.
* **Protected Members**: Denoted by a single underscore (\_). These are meant to be used by subclasses or within the class, but they can be accessed externally (although it’s discouraged).

**5. Polymorphism:**

* **Polymorphism** allows objects of different classes to be treated as objects of a common superclass. It also allows for method overriding, where a method in a subclass has the same name as a method in the superclass, but behaves differently.
* The speak() method is implemented differently in both the Bird and Dog classes, but both are treated as objects that implement the same method.

**6. Abstraction:**

* **Abstraction** is the concept of hiding the complex implementation details of a system and showing only the necessary parts of it. In Python, we can achieve abstraction using abstract classes and methods.
* An **abstract class** is a class that cannot be instantiated on its own and must be inherited by other classes. It can contain abstract methods, which are methods that have no implementation in the abstract class but must be implemented in the subclasses.
* The Animal class is abstract and defines an abstract method speak(), which must be implemented in any subclass like Dog.

**7. Constructor (\_\_init\_\_ Method):**

* The **constructor** is a special method in a class that gets called when an object of the class is created. It's used to initialize the attributes of the object.

**Method Overriding:**

* **Method overriding** is when a method in a child class has the same name and parameters as a method in the parent class, but with a different implementation.