Python Advanced Assignment-1

Q1. What is the purpose of Python's OOP?

Ans.

The purpose of Python's Object-Oriented Programming (OOP) is to organize code into reusable and modular components using objects and classes. It allows for encapsulation (bundling of data and methods), inheritance (reusing and extending code), and polymorphism (interchangeability of components), making it easier to manage, understand, and maintain code.

Q2. Where does an inheritance search look for an attribute?

Ans.

An inheritance search looks for an attribute first in the instance object, then in the class it was created from, and finally in all parent classes, following the method resolution order (MRO) until it finds the attribute.

Q3. How do you distinguish between a class object and an instance object?

Ans.

- Class object: The blueprint or template for creating instances. It defines attributes and methods shared among all instances.

- Instance object: An object created from a class. It represents a specific object with its own data (attributes) and can call the class's methods.

Q4. What makes the first argument in a class's method function special?

Ans.

The first argument in a class's method function is typically named `self`, and it refers to the instance of the class calling the method. It allows access to the instance's attributes and other methods within the class.

Q5. What is the purpose of the `__init__` method?

Ans.

The `__init__` method is a constructor used to initialize the instance's attributes when a new object is created. It sets up the initial state of the object.

Q6. What is the process for creating a class instance?

Ans.

To create a class instance, you call the class name as if it were a function, passing any required arguments that the `__init__` method accepts:

```
instance = ClassName(arguments)
```

Q7. What is the process for creating a class?

Ans.

To create a class, you use the `class` keyword, followed by the class name and a colon. Inside the class, you define attributes and methods:

```
class ClassName:
    def __init__(self, args):
        # Initialization code

    def method_name(self):
        # Method code
```

Q8. How would you define the superclasses of a class?

Ans.

Superclasses (or base classes) are the classes from which a class inherits. They are defined by specifying them in parentheses after the class name:

```
class SubClass(SuperClass1, SuperClass2):
    pass
```

This indicates that `SubClass` inherits from `SuperClass1` and `SuperClass2`.