Assignment 1

# Q1

The purpose of Python's Object-Oriented Programming (OOP) is to provide a way to structure and organise code by creating objects that encapsulate data. OOP allows for the creation of reusable and modular code through the use of classes and objects. It promotes concepts such as encapsulation, inheritance, and polymorphism, which help in writing clean, maintainable, and extensible code.

# Q2

Inheritance search, looks for an attribute in a specific order known as the Method Resolution Order (MRO). It starts by searching for the attribute in the current class, then in its parent classes, and continues up the inheritance hierarchy until it finds the attribute or reaches the top-level class (usually object). If the attribute is not found at any level, a NameError is raised.

# Q3

In Python, a class object is the actual class itself, which acts as a blueprint or template for creating instances. It defines the attributes and behaviour that the instances of the class will have. An instance object is a specific object that was created from a class. It has all of the attributes and methods of the class, as well as any additional attributes or methods that were assigned to it after it was created.

The main difference between a class object and an instance object is that a class object cannot be used to perform any actions. It is simply a blueprint. An instance object, on the other hand, can be used to perform actions, because it has all of the attributes and methods of the class.

# Q4

The first argument in a class's method function, conventionally named self, is used to refer to the instance of the class that the method is being called on. It is a reference to the instance itself and allows accessing its attributes and methods. This argument is usually named self, but it can be named anything else. The self argument is used to access the attributes and methods of the object that the method is being called on.

# Q5

The \_\_init\_\_ method, also known as the constructor, is a special method in Python classes. It is automatically called when an instance of a class is created. The purpose of the \_\_init\_\_ method is to initialise the object's attributes and perform any necessary setup or initialization tasks. It allows us to specify the initial state of an object by accepting arguments and assigning them to instance variables. The \_\_init\_\_ method is commonly used to set up the initial state of objects before they are used.

# Q6

The process for creating a class instance, also known as object instantiation, involves the following steps:

Define a class: First, we need to define a class using the class keyword. The class serves as a blueprint or template for creating instances.

Create an instance: To create an instance of a class, we call the class as if it were a function, passing any necessary arguments specified by the class's \_\_init\_\_ method. This is done by appending parentheses to the class name, like my\_instance = MyClass().

Initialise the instance: The \_\_init\_\_ method of the class is automatically called when we create an instance. It initialises the instance by setting its initial state and any instance variables.

Use the instance: Once the instance is created and initialised, we can use it to access its attributes and invoke its methods using dot notation, such as my\_instance.attribute or my\_instance.method().

## Q7

The process for creating a class involves the following steps:

Use the class keyword: Start by using the class keyword followed by the name of the class we want to create. Class names conventionally use CamelCase.

Define the class attributes and methods: Within the class block, define the attributes (variables) and methods (functions) that the class will have. These attributes and methods describe the behaviour and properties of objects created from the class.

Optionally, define the \_\_init\_\_ method: If we want to perform any initialization when an instance of the class is created, define the \_\_init\_\_ method within the class. This method takes self as its first parameter and is used to set up the initial state of the instance.

Use the class: After defining the class, we can create instances of the class by calling it as a function, passing any necessary arguments. Instances can be stored in variables and used to access the attributes and methods of the class.

# Q8

The superclasses of a class refer to the classes from which the current class directly inherits. In Python, the superclasses are defined in the parentheses following the class name when defining the class.

For example:

class SubClass(SuperClass1, SuperClass2):

# Class body

In this example, SubClass is the class being defined, and it directly inherits from SuperClass1 and SuperClass2. These superclasses provide the attributes and methods that the subclass can inherit and use. The subclass can override or extend the inherited attributes and methods, or it can define its own unique attributes and methods.

The superclasses form an inheritance hierarchy, where the subclass inherits the attributes and methods of its superclasses. If a requested attribute or method is not found in the subclass, the inheritance search continues through the superclasses until the attribute or method is found or until the top-level class (usually object) is reached.