ASSIGNMENT 1

QUESTION 1. What is the purpose of Python's OOP?

ANSWER :

1. OOP is a programming language feature that allows you to group variables and functions together into new data types, called classes, from which you can create objects.
2. By organizing your code into classes, you can break down a monolithic program into smaller parts that are easier to understand and debug.

QUESTION 2. Where does an inheritance search look for an attribute?

ANSWER :

Since, the objects subclass inherit attributes attached to objects in parent-class. The inheritance search takes place from child class or subclass to parent class, i.e.First find the first occurrence of attribute by looking in object, then in all classes above it, from bottom to top.

QUESTION 3. How do you distinguish between a class object and an instance object?

ANSWER :

Class Object	**Instance Object**
When we create a class in python then a class object is created so whenever python finds a class statement in the whole program then it creates a class object and assigns a name to that object i.e. class name. Since everything in python is an object so the class itself is an object and is the instance of metaclass. The class object comes from the **`'class'`** statement in code. whenever we encounter a class statement then a <class object> will be created. class object inherits the attributes of its parent classes.	When we call a class, it creates an instance object of that class from which the object has been created. The instance object comes from a call i.e. when we call the class. Actually, we are creating instance objects of that class. Instance objects are real objects in a python code process. The instance object has access to attributes of the class from which it is created. Instance object inherits the attributes of the class object from which it was created.

QUESTION 4. What makes the first argument in a class's method function special?

ANSWER :

- The first argument of every class method, including init, is always a reference to the current instance of the class. By convention, this argument is always named self. In the init method, self refers to the newly created object; in other class methods, it refers to the instance whose method was called.

QUESTION 5. What is the purpose of the `__init__` method?
ANSWER:

- To assign object attributes at creation time, we need the special Python object initialization method `__init__():`
- `__init__()` is the special Python name for a method that initializes an individual object from its class definition.
- . The `__init__()` method is where we commonly set the initial values of attributes therefore it is also called as an *initializer.*

QUESTION 6. What is the process for creating a class instance?
ANSWER:

```
class MyClass:
    pass
```

- **`x = MyClass()`** creates a new instance of the class and assigns this object to the local variable `x`.
- Creating a new class creates a new *type* of object, allowing new *instances* of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state.

QUESTION 7. What is the process for creating a class?
ANSWER :
There are two easy ways by which a class can be created in python.

```
class Books():
    pass
```

`or,`

```
class Books:
    pass
```

Class is created by using the **`class`** keyword followed by the name of the class.

QUESTION 8. How would you define the superclasses of a class?
ANSWER :
When we create a new class from an existing class (with or without some additions or changes), i.e., by putting the name of the existing class inside the parentheses of the newly created class then this existing old class is called a superclass or parent class or base class of the newly created class.

```
class A:
    print('superclass of B')

Class B(A):
    print('subclass of A')
```