**Advance Assignment - 01**

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

**Answer :**

* It is used to structure a software program into simple, reusable pieces of code blueprints (usually called classes), which are used to create individual instances of objects.
* It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.

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

**Answer :**

* The whole point of a namespace tool like the class statement is to support name inheritance.
* In Python, inheritance happens when an object is qualified, and involves searching an attribute definition tree (one or more namespaces).
* Every time you use an expression of the form object.attr where object is an instance or class object, Python searches the namespace tree at and above object, for the first attr it can find. Because lower definitions in the tree override higher ones, inheritance forms the basis of specialization.

Attribute tree construction :

* 1. **Instance attributes** are generated by assignments to self attributes in methods.
  2. **Class attributes** are created by statements (assignments) in class statements.
  3. **Superclass links** are made by listing classes in parentheses in a class statement header.

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

**Answer :**

|  |  |
| --- | --- |
| **Class Object** | **Instance Object** |
| 1) 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. | 1) When we call a class, it creates an instance object of that class from which the object has been created. |
| 2) Class objects provide default behaviour and serve as factories for instance objects. | 2) Instance objects are real objects in your python code process. |
| 3) Class object inherits attributes of its parents classes. | 3) Instance objects inherits attributes of the class object from which it is created. |
| 4) e.g.  class MyClass:  pass  above code will generate a class object and name it ‘MyClass’. From this class object, we will create instance objects. | 4) e.g.  obj=MyClass  the above statement creates an object and names it to obj which is an instance of MyClass. |

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

**Answer :**

The calling process is automatic while the receiving process is not (its explicit). This is the reason the first parameter of a function in class must be the object itself. Writing this parameter as self is merely a convention. It is not a keyword and has no special meaning in Python.

Q5. What is the purpose of the \_\_init\_\_ method?

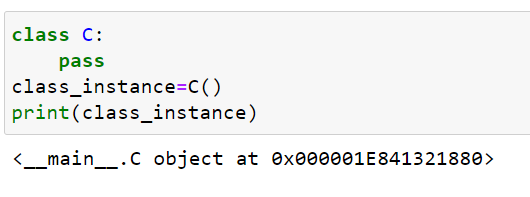
**Answer :**

* The \_\_init\_\_ method lets the class initialize the object's attributes and serves no other purpose. It is only used within classes.
* In short, \_\_init\_\_ allows you to initialize an instance of a class with specific data when the instance is created.

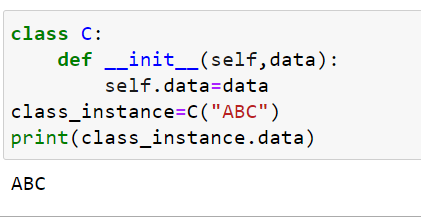
Q6. What is the process for creating a class instance?

**Answer :**

* Use the classname to create an instance of class. Call ClassName() to create a new instance of the class ClassName.



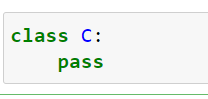
* To pass parameters to the class instance, the class must have an [\_\_init\_\_()](https://www.adamsmith.haus/python/docs/builtins.object.__init__) method. Pass the parameters in the constructor of the class.



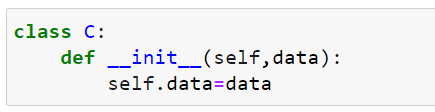
Q7. What is the process for creating a class?

**Answer :**

* A Class is like an object constructor, or a "blueprint" for creating objects. To create a class we use keyword “class”.



* All classes have a function called \_\_init\_\_(), which is always executed when the class is being initiated.



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

**Answer :**

* A superclass is the class from which many subclasses can be created.
* The subclasses inherit the characteristics of a superclass.
* The superclass is also known as the parent class or base class.