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

Ans: Object oriented programming (OOP) focusses on creating reusable code.

OOPS uses class and objects where class is the blueprint for the model, object is the actual entity which is built on that blueprint.

This OOPS helps us to break down the actual project into multiple problems, which can be easily maintained and modified,

Like if there is an application where we need to divide customer and employees. We can create 2 class, employee and customer can then divide them, by putting the modules for each of them separately.

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

Ans: inheritance searches goes bottom up approach,

So if there is a multilevel inheritance, it will check its nearest parent class and identify, if not available then goes one step up. Search continues until the attribute is found or it has reached the superclass.

In case of the multiple inheritance, it will check from left to right.

1st parent class, then 2nd parent class ,… continues till the last parent.

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

Ans

When we create a class, a class object is created. The class object stores the blueprint of the class. And does not actually store any information about the class

Instance object is derived from the class. When we call the class , an instance will be creates The instance object is the actual entity which represents the actual functionality.

|  |  |
| --- | --- |
| Class | Instance |
| Class is the blueprint of the object | Instance is the individual object based on that class |
| As this a blueprint, it does not consume any memory, save any data | Instance saves the data thus uses memory |
|  | From 1 class there can be multiple instances that can be created |

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

Ans: the first argument is the pointer argument , which is used to point to class itself.

We can access the attribute, methods by using the keyword.

We usually use self as the keyword, but it is not mandatory but more like the industry standard.

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

Ans: \_\_init\_\_ is unbuilt function.

It acts as a constructor for the class

all the paramters assigned for the \_\_init\_\_ should be initialized , when the object is created.

The object cannot be created, if there is \_\_init\_\_ and the required attributes are not available.

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

Ans: the instance of the class is object.

We call the name of the Class, and pass the attributes as the argument to the \_\_init\_\_ method of the class

E.g. :- etios= Car(‘Toyota’, ‘Diesel’)

Here the Etios object is created with attributes ‘Toyota’, ‘Diesel’ i.e. the company name and the Fuel type.

Q7. What is the process for creating a class?

Ans: the keyword ‘class’ is used to create a class. This is followed by the name of the class and “:”

E.g.

**class classname:**

**classvar= 1 # class variable**

**def \_\_init\_\_(self, key1): # constructor**

**self.key1= key1 # instance variable**

**def return key(): #instance method**

**return self.key1**

This will create a class Car.

Inside this class we can use \_\_init\_\_ which will be used as a constructor

Can define the global variables

Define method(function) which will perform some operation

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

Ans: a class using which multiple subclasses has been created is known as super class.

In simple terms, it is known as parent class.

Parent class arguments, methods acan all be used by the child class for its operations.

Eg if we create 2 class

|  |  |
| --- | --- |
| Class employee:  Hi= ‘hi I am employee’  Id=’this is my ID’  Manager=’manager’ | Class manager(Employee):  Reportees=[R1,R2,R3] |

In this case, employee is the super class, manager is the subclass.

The manager will inherit all the properties of the employee class (EmpID, managername) plus it will have properties of its own (Reportees name)