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

**ANS:-**

1. **Provides a clear program structure and a clean code**
2. **Facilitates easy maintenance and modification of existing code.**
3. **Since the class is sharable, the code can be reused.**
4. **Since the class is sharable, the code can be reused and many other such Advantages.**

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

**ANS:-**

**Attribute fetches are simply tree searches. The term inheritance is applied because objects lower in a tree inherit attributes attached to objects higher in that tree. As the search proceeds from the bottom up, in a sense, the objects linked into a tree are the union of all the attributes defined in all their tree parents, all the way up the tree.**

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

**ANS:-**

**Objects and Instances are almost similar and are often used interchangeably but with a small difference. Object is a generic term , it is physically present but remains undiferrentiated. Instance is something that gives them a separate identity.**

**Object is the physical entity for which memory is allocated. Object contains many instances.**

**Instance : An instance is also the physical manifestation of a class that occupies memory and has data members.**

**e.g. There is Class car, when I create c = car(), c is object. When we create different object with different specifications(car name, type, company) such as i10, i20, Creta, audi7 are called as instances which actually exists.**

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

**ANS:-**

**Self represents the instance of the class. By using the “self” keyword we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.The reason you need to use self. is because Python does not use the @ syntax to refer to instance attributes. Python decided to do methods in a way that makes the instance to which the method belongs be passed automatically, but not received automatically: the first parameter of methods is the instance the method is called on.**

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

**ANS:-**

**The task of init method is to initialize(assign values) to the data members of the class when an object of class is created. It contains collection of statements that are executed at time of Object creation. It is run as soon as an object of a class is instantiated. The method is useful to do any initialization you want to do with your object.**

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

**ANS:-**

**To create instances of a class, you call the class using class name and pass in whatever arguments its init method accepts.**

**Q7. What is the process for creating a class?**

**ANS:-**

**The class statement creates a new class definition. The name of the class immediately follows the keyword class followed by a colon.Explained below**

***#Example for questtion 6 & 7***

**class Student:**

**def \_\_init\_\_(self,name):**

**self.name = name**

**def \_\_str\_\_(self):**

**return f"This is {self.name} (an instance of a class Student)"**

**student1 = Student("Snehal Suryawanshi")**

**print(student1)**

**This is Snehal Suryawanshi (an instance of a class Student)**

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

**ANS:-**

**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.**

**class A:**

**def method\_A(self):**

**return "this is method of class A"**

**class B(A):**

**def method\_B(self):**

**return "this is method of class B"**

**class C(A):**

**def method\_B(self):**

**return "this is method of class C"**

**class D(A):**

**def method\_B(self):**

**return "this is method of class D"**

***# Class B, C, D are subclasses and Class A is superclass***

**a = A()**

**b = B()**

**c = C()**

**d = D()**

**print(a.method\_A())**

**print(b.method\_B())**

**print(b.method\_A())**

**print(c.method\_A())**

**print(d.method\_A())**

***# objects of class B, C and D are able to access method of #class A because Class A is superclass.***

**this is method of class A**

**this is method of class B**

**this is method of class A**

**this is method of class A**

**this is method of class A**