Q-1 What are oop concepts? Is multiple inheritance supported in java ?

Ans. Object-oriented programming (OOP) is method of structuring a program by bundling related properties and behaviors into individual objects. In this tutorial, you'll learn the basics of object-oriented programming in python.

Coceptually, objects are like the components of a system. Think of a programme as a factory assembly line of sorts. At each step of the assembly line a system component process some material, ultimately transforming new material into a finished product.

An object contains data, like the raw or preprocessed materials at each step on an assembly line, and behaviour, like the action each assembly line component performs.

Create a class, which is like a blueprint for creating an object

Use classes to create new objects

Model systems with class inheritance

Object-oriented programming is a programming paradigm that provides a mean of structuring programs so that properties and behaviors are bundled into individual objects.

For instance, an object could represent a person with properties like a name, age,and address and behaviours such as walking , talking,breathing, and running. Or it could represent an email with properties like a recipient list, subject, and body and behaviours like adding attachments and sending.

Put another way , object-orinented programming is an approach for modelling concreate, real- world things, like cars, as well as relations between things, like companies and employees, students and teachers , and so on .

Another common programming paradigm is procedural programming , which structures a program like a recipe in that it provides a set of steps, in the form of functions and code blocks, that flow sequnentially in order to compete a task.

How to define class in python

Class student:

Name=’komal’

Sub=”pyhton”

One significant difference between classes and interfaces is that classes can have fields whreas interfaces cannot. In addition, you can instantiate a class to create an object. Which you cannot do with interfaces. As explained in the section What is an object? An obeject stores its state in fields, which are defined in classes. One reason why the java programming language does not permit you to extend more than on class is to avoid the issuses of multiple inheritance of state which is the ability to inherit fields from multiple classes. For example suppose that you are able to define a new class that extends multiple classes When you create an object by instantiating

Q-2 Explain Inheritance in Python with an example? What is init? Or What Is A Constructor In Python?

Ans Inheritance allows us to define a class that inherits all the methods and properties from another class.

Parent class is the class being inherited from, also called base class.

Child class is the class that inherits from another class, also called derived class.

EX:

```
class Person:
  def __init__(self,                fname,                lname):
```

```
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

x = Person("John", "Doe")
x.printname()
```

Create a Child Class

create a class named Student, which will inherit the properties and methods from the Person class:

```
class Student(Person):
  pass
```

Add the __init__() Function

So far we have created a child class that inherits the properties and methods from its parent.

We want to add the __init__() function to the child class (instead of the pass keyword).

Note: The __init__() function is called automatically every time the class is being used to create a new object.

Example

Add the __init__() function to the Student class:

```
class Student(Person):
  def __init__(self, fname, lname):
    #add properties etc.
```

When you add the __init__() function, the child class will no longer inherit the parent's __init__() function.

Note: The child's __init__() function **overrides** the inheritance of the parent's __init__() function.

To keep the inheritance of the parent's __init__() function, add a call to the parent's __init__() function:

Ex:

```
class Student(Person):
  def __init__(self, fname, lname):
    Person.__init__(self, fname, lname)
```

Now we have successfully added the __init__() function, and kept the inheritance of the parent class, and we are ready to add functionality in the __init__() function.

Use the super() Function

Python also has a super() function that will make the child class inherit all the methods and properties from its parent:

Add a property called graduationyear to the Student class:

Ex:

```
class Student(Person):
  def __init__(self, fname, lname):
    super().__init__(fname, lname)
    self.graduationyear = 2019
```

Q-3 What is Instantiation in terms of OOP terminology?

Ans. What Is Instantiation In Terms Of Oop Terminology In programming, instantiation is the conception of a real-world example or the specific realization of an abstraction or model, e.g. B. a class of objects or a computer process. Instantiation means creating such an instance by, for example, identifying a particular difference of an

object within a category, giving it a name, and placing it in a physical location.

The potential confusion for people new to OO is that Instantiation is what happens to create an object, yet this is a class-to-class relationship no objects are involved. A parameterized class cannot have objects instantiated from it unless it is first instantiated itself (hence the confusing name for this type of relationship).

Instantiation is creating an instance of an object in an object-oriented software design (OOP) verbal. An instantiated object is given a name and is create in memory or on disk using the structure described in a class declaration.

What Does Instantiation Mean?

In computer science, instantiation (verb) and instantiation (noun) refer to the creation of an object (or "instance" of a particular class) in an object-oriented programming (OOP) language. An instantiated object is called and created in recollection or on disk by reference to a class declaration. It is also limited or defined by its attributes. Resources are allocated, assigned somewhere in the overall set of codebases, and programmers play a role in its construction.

Object instantiation uses different methods and terminology in other programming environments: for example, in C++ and similar languages, What Is Instantiation In Terms Of Oop Terminology class means creating an object, whereas, in Java, it means creating an object. Initializing a class creates a specific category. The results are the same in both languages (executables), but the way to get there is slightly different.

Technologists may talk about instantiation differently depending on their environments and the protocols and methods they use when working with active classes to create objects. For example, starting a virtual server involves virtualizing each server's predefined features

(disk space, allocated RAM, operating system type, installed software).

In general, “instantiating” an object means creating it as an instance of a class. But there is technical information on how to do this. Terminology can be confusing. Linking “example” to “example” helps, but the jargon may seem rather cloudy for those who don’t actually work with object-oriented programming. The key is to fully understand what a class is. And also, What an object is and how they differ, how a thing works in code. What it contains and what it means to create an object.

What Is Instantiation In Object-Oriented Programming?

In object-oriented programming (OOP), an object is an instance of a class. Everything in a class has a specific set of associate variables or properties, props or means of accessing those variables. And functions or methods. In OOP languages, a class is like a model in which variables and methods are defined, and every time a new instance of the class is created (instantiated), an object is created, hence the term object-oriented.

These properties (or variables) can be thought of as data describing the objects created from a class. For example, a class can represent a new employee. In this case, the properties of this class can include title, role, responsibilities, salaries, etc.

An instance of an object can be state by giving it a exceptional name that can be use in a program. This process is called instantiation. A class can also be instantiate to create an body, a real class instance. The thing is an executable file that can be run on a computer.

One use of instantiation is in data modelling and programming prior to object-oriented programming. Another to transform an abstract object into a real (full of data) item. As done by creating an entry in a

database table. (which can thought of as a kind of class model for objects).

What Is Instantiation In C++?

In C++, creating a new instance of the class is call instantiating. Memory is allocate for this object and the constructor of the class is execute. Programmers can instantiate objects on the mountain with a new keyword or on the load as a inconstant declaration. Every time an object of this class is instantiate, the compiler calls special class members.

Instantiation is when a new instance of the class is created (an object). In C++ when an class is instantiated memory is allocated for the object and the classes constructor is run. In C++ we can instantiate objects in two ways. On the stack as a variable declaration, or on the heap with the new keyword.

Q-4 What is used to check whether an object o is an instance of class A?

Ans. Example

Check if the number 5 is an integer:

x = isinstance(5, int)

Definition :

The isinstance() function returns True if the specified object is of the specified type, otherwise False.

If the type parameter is a tuple, this function will return True if the object is *one* of the types in the tuple.

Syntax

isinstance(*object*, *type*)

Parameter Values

Parameter	Description
object	Required. An object.
type	A type or a class, or a tuple of types and/or classes

Example

Check if "Hello" is one of the types described in the type parameter:

```
x = isinstance("Hello", (float, int, str, list, dict, tuple))
```

Try it Yourself »

Example

Check if y is an instance of myObj:

```
class myObj:
  name = "John"


y = myObj()


x = isinstance(y, myObjc
```

The issubclass() function, to check if an object is a subclass of another object.

Q-5 What relationship is appropriate for Course and Faculty?

Ans.

In python, Multilevel inheritance is one type of inheritance being used to inherit both base class and derived class features to the newly derived class when we inherit a derived class from a base class and another derived class from the previous derived class up to any extent of depth of classes in python is called multilevel inheritance. While inheriting the methods and members of both base class and derived class will be inherited to the newly derived class. By using inheritance, we can achieve the reusability of code, relationships between the classes. If we have three classes A, B, and C where A is the superclass (Parent), B is the subclass (child), and C is the subclass of B (Grandchild)

Syntax of Multilevel Inheritance

Let us say we have three classes with names as the base, derived1, and derived2. The base class has its members and functions as below:

```
class base:
    //member of base class
   // functions of base class
    Pass
class derived1(base):
   // members of derived1 class + base class
  // functions of derived1 class + base class
   pass
```

```
class derived2(derived1):
   // members of derived2 class + derived1 class + base class
  // functions of derived2 class + derived1 class + base class
    Pass
```

In the above class example, we can see that the class base is the base class and class derived is inherited from the class base and class derived2 is inherited from the class derived1.