1 .Define Object Oriented Programming Language?  
  
Object-oriented programming (OOP) refers to a type of computer programming (software design) in which programmers define the data type of a data structure, and also the types of operations (functions) that can be applied to the data structure.  
In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.

2 .List down the Benefits of OOP?  
·        Through inheritance, we can eliminate redundant code and extend the use of existing classes.  
·        We can built programs from standard working modules that communicate with one another rather than, having to start writing the code from scratch. This leads to saving of development time and higher productivity.  
·        The principle of data hiding helps the programmers to built secure program that can’t be invaded by code in other parts of the program.  
·        It is possible to have multiple objects to coexist without any interference.  
·        It is possible to map objects in the problem domain to those objects in the program.  
·        It is easy to partition the work in a project based on objects.  
·        The data-centered design approach enables us to capture more details of the model in an implementable form.  
·        Object-oriented systems can be easily upgraded from small to large system  
·        Message passing technique for communication between objects make the interface descriptions with external system much simpler.  
·        Software complexity can be easily managed.

3.Differentiate between function and method?

Java is also an OOP language, but their is no concept of Function in it. But Python has both concept of Method and Function.

Python Method:  
1. Method is called by its name, but it is associated to an object (dependent).  
2. A method is implicitly passed the object on which it is invoked.  
3. It may or may not return any data.  
4. A method can operate on the data (instance variables) that is contained by the corresponding class.

Python Functions:  
1. Function is block of code that is also called by its name. (independent)  
2. The function can have different parameters or may not have any at all. If any data (parameters) are passed, they are passed explicitly.  
3. It may or may not return any data.  
4. Function does not deal with Class and its instance concept.  
3.Differentiate between function and method?

Java is also an OOP language, but their is no concept of Function in it. But Python has both concept of Method and Function.

Python Method  
1. Method is called by its name, but it is associated to an object (dependent).  
2. A method is implicitly passed the object on which it is invoked.  
3. It may or may not return any data.  
4. A method can operate on the data (instance variables) that is contained by the corresponding class

Functions  
1. Function is block of code that is also called by its name. (independent)  
2. The function can have different parameters or may not have any at all. If any data (parameters) are passed, they are passed explicitly.  
3. It may or may not return any data.  
4. Function does not deal with Class and its instance concept.

4.Define the following terms:  
1. Class  
2. Object  
3. Attribute  
4. Behavior  
Class:  
  
Classes are used to create new user-defined data structures that contain arbitrary information about something  
. In the case of an animal, we could create an Animal() class to track properties about the Animal like the name and age.  
It’s important to note that a class just provides structure—it’s a blueprint for how something should be defined, but it doesn’t actually provide any real content itself. The Animal() class may specify that the name and age are  
necessary for defining an animal, but it will not actually state what a specific animal’s name or age is.  
It may help to think of a class as an idea for how something should be defined.  
  
Object:  
  
While the class is the blueprint, an instance is a copy of the class with actual values, literally an object belonging to a specific class. It’s not an idea anymore; it’s an actual animal, like a dog named Roger who’s eight years old.  
  
Attribute:  
  
All classes create objects, and all objects contain characteristics called attributes (referred to as properties in the opening paragraph). Use the \_\_init\_\_() method to initialize (e.g., specify) an object’s initial attributes by giving them their default value (or state). This method must have at least one argument as well as the self variable, which refers to the object itself (e.g., Dog).  
While instance attributes are specific to each object, class attributes are the same for all instances—which in this case is all dogs.  
  
Behaviour:  
  
Objects in Python are generally classified according to their behaviors and the features that they implement. For example, all of the sequence types such as strings, lists, and tuples are grouped together merely because they all happen to support a common set of sequence operations such as s[n], len(s), etc.  
The behavior of each data type depends entirely on the set of special methods that it implements.  
User-defined classes can define new objects that behave like the built-in types simply by supplying an appropriate subset of the special methods described in this section. In addition, built-in types such as lists and dictionaries can be specialized (via inheritance) by redefining some of the special methods  
  
5.Write a code in python in which create a class named it Car which  
have 5 attributes such like (model, color and name etc.) and 3  
methods. And create 5 object instance from that class.  
  
class Car():  
    def\_\_init\_\_(self, model, name, color, make, speed):  
        self.m=model  
        self.n=name  
        self.c=color  
        self.m=make  
        self.s=speed  
         
      # create class methods  
    def start(self):  
        print ("Engine started")  
  
    def stop(self):  
        print ("Engine switched off")  
         
    def unlock(self):  
        print("Car unlocked")  
         
car\_a = Car("model",)  
car\_b = Car("name")  
car\_c = Car()  
car\_d = Car()  
car\_e = Car()  
print(car\_a.model)  
print([car\_b.name](http://car_b.name/))  
car\_c.stop()  
car\_d.unlock()  
car\_e.start()