**Qno1: Define Object Oriented Programming Language?**

Object-oriented Programming, or OOP for short, is a [programming paradigm](http://en.wikipedia.org/wiki/Programming_paradigm) which provides a means of structuring programs so that properties and behaviors are bundled into individual *objects*.

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

Put another way, object-oriented programming is an approach for modeling concrete, real-world things like cars as well as relations between things like companies and employees, students and teachers, etc. OOP models real-world entities as software objects, which have some data associated with them and can perform certain functions.

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

The key takeaway is that objects are at the center of the object-oriented programming paradigm, not only representing the data, as in procedural programming, but in the overall structure of the program as well.

**Qno2: list down the benefits of OOP?**

* It provides a clear modularstructure for programs which makes it good for defining abstract datatypes in which implementation details are hidden
* Objects can also be reused within an across applications. The reuse of software also lowers the cost of development. More effort is put into the object-oriented analysis and design, which lowers the overall cost of development.
* It makes softwareeasiertomaintain***.*** Since the design is modular, part of the system can be updated in case of issues without a need to make large-scale changes
* Reuse also enables fasterdevelopment. Object-oriented programming languages come with rich libraries of objects, and code developed during projects is also reusable in future projects.
* It provides a good framework for code libraries where the supplied software components can be easilyadaptedandmodifiedbytheprogrammer. This is particularly useful for developing graphical user interfaces.
* BetterProductivityas OOP techniques enforce rules on a programmer that, in the long run, help her get more work done; finished programs work better, have more features and are easier to read and maintain. OOP programmers take new and existing software objects and "stitch" them together to make new programs. Because object libraries contain many useful functions, software developers don't have to reinvent the wheel as often; more of their time goes into making the new program.

**Qno3: Differentiate between function and method?**

**Python Method**

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

**Functions**

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

**Difference between method and function**

Simply, function and method both look similar as they perform in an almost a similar way, but the key difference is the concept of ‘Class and its Object.

Functions can be called only by its name, as it is defined independently. But methods can’t be called by its name only, we need to invoke the class by a reference of that class in which it is defined, i.e. method is defined within a class and hence they are dependent on that class.

**Qno:4 Define the following terms:**

1. **Class**
2. **Object**
3. **Attribute**
4. **Behavior**

## Classes

A class is a code template for creating objects. Objects have member variables and have behavior associated with them. In python a class is created by the keyword class.

An object is created using the constructor of the class. This object will then be called the instance of the class. In Python we create instances in the following manner

Instance = class(arguments)

## Objects

An object is an instance of a class. We can take the Shark class defined above, and use it to create an object or instance of it.

We’ll make a Shark object called Sammy

we initialized the object sammy as an instance of the class by setting it equal to Shark().

sammy = Shark()

sammy.swim()

sammy.be\_awesome()

The Shark object sammy is using the two methods swim() and be\_awesome(). We called these using the dot operator (.), which is used to reference an attribute of the object. In this case, the attribute is a method and it’s called with parentheses, like how you would also call with a function.

Because the keyword self was a parameter of the methods as defined in the Shark class, the sammy object gets passed to the methods. The self-parameter ensures that the methods have a way of referring to object attributes.

When we call the methods, however, nothing is passed inside the parentheses, the object sammy is being automatically passed with the dot operator.

**class** **Shark**:

**def** **swim**(self):

print("The shark is swimming.")

**def** **be\_awesome**(self):

print("The shark is being awesome.")

**def** **main**():

sammy = Shark()

sammy.swim()

sammy.be\_awesome()

**if** \_\_name\_\_ == "\_\_main\_\_":

main()

**Attributes:**

Pandas objects have a number of attributes enabling you to access the metadata

* shape: gives the axis dimensions of the object, consistent with and array
* Axis labels
  + Series: index (only axis)
  + Data Frame: index (rows) and columns

A class **attribute** is a **Python** variable that belongs to a class rather than a particular object. It is shared between all the objects of this class and it is **defined** outside the constructor function, \_\_init\_(self,...) , of the class.

Behavior:

Behave is one of the most popular BDD frameworks in Python. It is very similar to other Gherkin-based Cucumber frameworks despite not holding the official Cucumber designation. Behave has two primary layers:

1. Behavior specs written in Gherkin**.** feature files
2. Step definitions and hooks written in Python modules that implement Gherkin steps

As shown in the example above, Gherkin scenarios use a three-part format:

1. Given some initial state
2. When an action is taken
3. Then verify the outcome

Each step is "glued" by decorator to a Python function when behave runs tests.