Object-oriented programming (OOP)

is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields (often known as *attributes* or *properties*), and code, in the form of procedures (often known as *methods*). A feature of objects is an object's procedures that can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP, computer programs are designed by making them out of objects that interact with one another.[1][2] OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

Many of the most widely used programming languages (such as C++, Java, Python, etc.) are multi-paradigm and they support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming. Significant object-oriented languages
include Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, MATLAB, and Smalltalk.

1. Re-usability

It means reusing some facilities rather than building it again and again. This is done with the use of a class. We can use it ‘n’ number of times as per our need.

2. Data Redundancy

This is a condition created at the place of data storage (you can say Databases)where the same piece of data is held in two separate places. So the data redundancy is one of the greatest advantages of OOP. If a user wants a similar functionality in multiple classes he/she can go ahead by writing common class definitions for the similar functionalities and inherit them.

3. Code Maintenance

This feature is more of a necessity for any programming languages, it helps users from doing re-work in many ways. It is always easy and time-saving to maintain and modify the existing codes with incorporating new changes into it.

4. Security

With the use of data hiding and abstraction mechanism, we are filtering out limited data to exposure which means we are maintaining security and providing necessary data to view.

5. Design Benefits

If you are practicing on OOPs the design benefit a user will get is in terms of designing and fixing things easily and eliminating the risks (if any). Here the Object Oriented Programs forces the designers to have a longer and extensive design phase, which results in better designs and fewer flaws. After a time when the program has reached some critical limits, it is easier to program all the non-OOP’s one separately.

6. Better productivity

with the above-mentioned facts of using the application definitely enhances its users overall productivity. This leads to more work done, finish a better program, having more inbuilt features and easier to read, write and maintain. An OOP programmer cans stitch new software objects to make completely new programs. A good number of libraries with useful functions in abundance make it possible.

7. Easy troubleshooting

lets witness some common issues or problems any developers face in their work.

- Is this the problem in the widget file?
- Is the problem is in the WhaleFlumper?
- Will I have to trudge through that ‘sewage.c’ file?
- Commenting on all these issues related to code.

So, many a time it happens that something has gone wrong which later becomes so brainstorming for the developers to look where the error is. Relax! Working with OOP language you will know where to look for. This is the advantage of using encapsulation in OOP; all the objects are self-constrained. With this modality behavior, the IT teams get a lot of work benefits as they are now capable to work on multiple projects simultaneously with an advantage that there is no possibility of code duplicity.

8. Polymorphism Flexibility

Let’s see a scenario to better explain this behavior.

You behave in a different way if the place or surrounding gets change. A person will behave like a customer if he is in a market, the same person will behave like a student if he is in a school and as a son/daughter if put in a house. Here we can see that the same person showing different behavior every time the surroundings are changed. This means polymorphism is flexibility and helps developers in a number of ways.

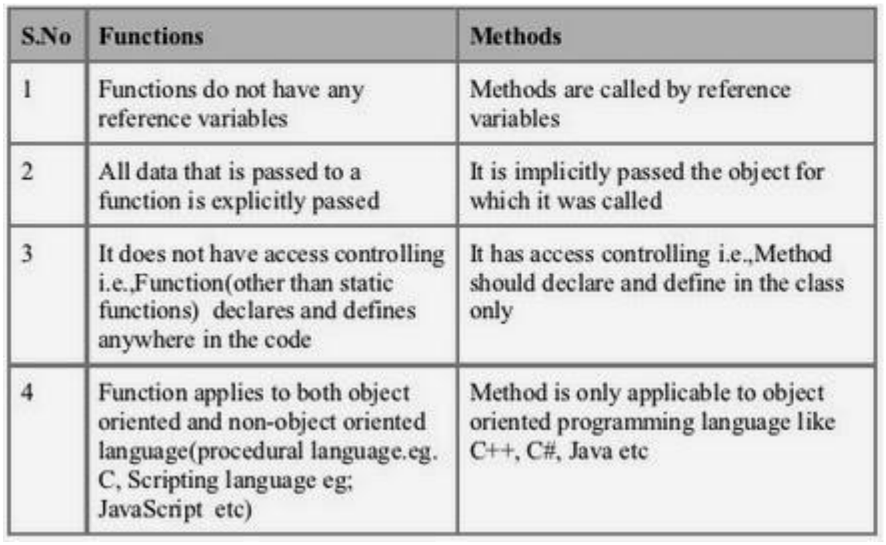
- It’s simplicity
- Extensibility

9. Problems solving

Decomposing a complex problem into smaller chunks or discrete components is a good practice. OOP is specialized in this behavior, as it breaks down your software code into bite-sized – one object at a time. In doing this the broken components can be reused in solutions to different other problems (both less and more complex) or either they can be replaced by the future modules which relate to the same interface with implementations details.

A general relatable real-time scenario – at a high level a car can be decomposed into wheels, engine, a chassis soon and each of those components can be further broken down into even smaller atomic components like screws and bolts. The engine's design doesn't need to know anything about the size of the tires in order to deliver a certain amount of power (as output) has little to do with each other.

Differeniate between function and method:



S.No	Functions	Methods
1	Functions do not have any reference variables	Methods are called by reference variables
2	All data that is passed to a function is explicitly passed	It is implicitly passed the object for which it was called
3	It does not have access controlling i.e.,Function(other than static functions) declares and defines anywhere in the code	It has access controlling i.e.,Method should declare and define in the class only
4	Function applies to both object oriented and non-object oriented language(procedural language.eg. C, Scripting language eg; JavaScript etc)	Method is only applicable to object oriented programming language like C++, C#, Java etc

Class:

Classes (**OOP**) In **object-oriented** programming, a **class** is a blueprint for creating objects (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods). The user-defined objects are created using the **class** keyword.

Object:

In **object-oriented** programming (**OOP**), **objects** are the things you think about first in designing a program and they are also the units of code that are eventually derived from the process. ... Each **object** is an instance of a particular class or subclass with the class's own methods or procedures and data variables.

Attribute:

In **Object-oriented** programming(**OOP**), classes and objects have **attributes**. **Attributes** are data stored inside a class or instance and represent the state or quality of the class or instance. ... One can think of **attributes** as noun or adjective, while methods are the verb of the class.

Behavior:

Behavior. A class's **behavior** determines how an instance of that class operates; for example, how it will "react" if asked to do something by another class or object or if its internal state changes. **Behavior** is the only way objects can do anything to themselves or have anything done to them.