**Imperative Programming**as thename suggests is a type of programming paradigm that describes how the program executes. Developers are more concerned with how to get an answer step by step. It comprises the sequence of command imperatives. In this, the order of execution is very important and uses both mutable and immutable data. Fortran, [Java](https://www.geeksforgeeks.org/java/), [C](https://www.geeksforgeeks.org/c-programming-language/), [C++](https://www.geeksforgeeks.org/c-plus-plus/) programming languages are examples of imperative programming.

* In this, programs specify how it is to be done.
* It simply describes the control flow of computation.
* Its advantages include ease to learn and read, the notional model is simple to understand, etc.
* Its main goal is to describe how to get it or accomplish it.
* Its type includes procedural programming, object-oriented programming, parallel processing approach.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Declarative Programming**as thename suggests is a type of programming paradigm that describes what programs to be executed. Developers are more concerned with the answer that is received. It declares what kind of results we want and leave programming language aside focusing on simply figuring out how to produce them. In simple words, it mainly focuses on end result. It expresses the logic of computation. Miranda, Erlang, Haskell, Prolog are a few popular examples of declarative programming.

* In this, programs specify what is to be done.
* It simply expresses the logic of computation.
* Its main goal is to describe the desired result without direct dictation on how to get it.
* Its advantages include effective code, which can be applied by using ways, easy extension, high level of abstraction, etc.
* Its type includes logic programming and functional programming.

The null pointer exception :-

is a **runtime exception**, which is thrown when the program tries to use an object reference that is set to the null value .0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

not case-sensitive languages

such as [ABAP](https://en.wikipedia.org/wiki/ABAP), [Ada](https://en.wikipedia.org/wiki/Ada_(programming_language)), [Fortran](https://en.wikipedia.org/wiki/Fortran), [SQL](https://en.wikipedia.org/wiki/SQL) (for the syntax, and for some vendor implementations, e.g. [Microsoft SQL Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server), the data itself)[[NB 2]](https://en.wikipedia.org/wiki/Case_sensitivity#cite_note-5) and [Pascal](https://en.wikipedia.org/wiki/Pascal_(programming_language)). There are also languages, such as [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)), [Prolog](https://en.wikipedia.org/wiki/Prolog), and [Go](https://en.wikipedia.org/wiki/Go_(programming_language))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Programming languages that support auto garbage collection :-

languages that recovery feature built into programming languages such as [C#](https://www.techtarget.com/whatis/definition/C-Sharp) and [Java](https://www.theserverside.com/definition/Java).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Python integers are arbitrary-precision integers, also known as bignums. This means that they can be as large as we want, and their sizes are only limited by the amount of available memory. Bignums are handy to work with because we don't need to worry about such things as integer overflows and underflows

**Stack:-**

A stack is a special area of computer’s memory which stores temporary variables created by a function. In stack, variables are declared, stored and initialized during runtime.

It is a temporary storage memory. When the computing task is complete, the memory of the variable will be automatically erased. The stack section mostly contains methods, local variable, and reference variables

* A stack is a linear data structure.
* High-speed access
* Space managed efficiently by OS so memory will never become fragmented.
* Local variables only
* Limit on stack size dependent on OS.
* Memory is allocated in a contiguous block.
* Variables cannot be resized
* A stack can be implemented in 3 ways simple array based, using dynamic memory, and Linked list based.

## Heap:-

The heap is a memory used by programming languages to store global variables. By default, all global variable are stored in heap memory space. It supports Dynamic memory allocation.The heap is not managed automatically for you and is not as tightly managed by the CPU. It is more like a free-floating region of memory.

* Heap is a hierarchical data structure
* Slower compared to stack
* Heap Space not used as efficiently. Memory can become fragmented as blocks of memory first allocated and then freed.
* Variables can be resized.
* Doesn’t have a specific limit on memory size.