1. What the different between

* 1.Procedural programming
* 2. Declarative programming
* 3. Imperative programming

Procedural programming is a type of programming paradigm that focuses on procedures or functions. The program is structured into procedures that take inputs and produce outputs. The program follows a step-by-step approach to solve a problem, and the procedures can call each other to achieve the desired result.

Declarative programming is a programming paradigm that focuses on what needs to be done, rather than how it needs to be done. It provides a way to describe the problem in a high-level language, and the computer determines how to solve it. This type of programming is used for complex problems that would be difficult to solve with procedural programming.

Imperative programming is a programming paradigm that focuses on describing how to solve a problem. It consists of a series of statements that change the state of the program. The program follows a step-by-step approach to solve a problem. It is used for simple problems that can be solved with a few statements.

1. How python can handle with number more than 14 bytes

Python uses a technique called "arbitrary-precision arithmetic" to handle large integers. This means that the size of the integer is not fixed at compile time, but can grow or shrink as needed at runtime. The arithmetic operations on these large integers are performed by the Python interpreter, which uses algorithms specifically designed for dealing with such large numbers.

1. What is non-pointer exception

A null pointer exception is a runtime error that occurs when a program tries to access a null or uninitialized object pointer, causing the program to crash. In other words, it is an exception that occurs when a program attempts to use a pointer that doesn't point to any memory address, or when a program tries to access a member or function of a null pointer.

1. Types of not case sensitive languages
2. SQL (Structured Query Language)
3. HTML (Hypertext Markup Language)
4. CSS (Cascading Style Sheets)
5. YAML (YAML Ain't Markup Language)
6. TOML (Tom's Obvious, Minimal Language)
7. What the different between Heap and Stack

In computer science, Heap and Stack are two common memory management techniques. Here are some of the main differences between them:

1. Data structure:

* Stack is a linear data structure that follows Last-In-First-Out (LIFO) order. It means the last element pushed into the stack is the first one to be popped out of it.
* Heap is a tree-like data structure that follows the order property, which depends on the specific type of heap (min heap or max heap).

1. Memory allocation:

* In Stack, memory allocation and deallocation are handled automatically by the compiler. When a function is called, its local variables and function parameters are pushed onto the stack, and when the function returns, they are popped off the stack automatically.
* In Heap, memory allocation and deallocation are managed manually by the programmer. The programmer has to allocate memory from the heap using functions like malloc(), calloc(), or new(), and free it explicitly when it is no longer needed.

1. Memory size:

* The size of the Stack is limited, and it is usually smaller than the size of the Heap. The exact size of the stack depends on the operating system and the amount of memory allocated to the program.
* The size of the Heap is dynamic and can grow or shrink as needed during program execution.

1. Memory access:

* Accessing memory in the Stack is faster than accessing memory in the Heap because the Stack uses contiguous memory allocation and direct pointer manipulation.
* Accessing memory in the Heap is slower than accessing memory in the Stack because the Heap uses dynamic memory allocation and indirect pointer manipulation.

In summary, Stack and Heap are two different memory management techniques that have their own advantages and disadvantages. Stack is simple and fast, but its size is limited, and memory allocation is handled automatically. Heap is dynamic and flexible, but memory allocation and deallocation are managed manually, and memory access is slower.

1. What is the programming languages contain auto garbage collection and not.

Programming languages with automatic garbage collection:

1. Java
2. Python
3. Ruby
4. C#
5. JavaScript
6. Go
7. Swift
8. Kotlin
9. Rust
10. Lua

Programming languages without automatic garbage collection:

1. C
2. C++
3. Assembly
4. Fortran
5. Ada
6. Pascal
7. COBOL
8. PL/I