**What is difference between High Level Language Vs Low Level Language?**

**Ans.**  1. **Definition and Purpose:**

* + **High-Level Languages (HLL):**
    - Designed to be easy for humans to read and write.
    - Abstract away most of the hardware details, allowing focus on problem-solving and logic.
    - Examples include Python, Java, and C++.
  + **Low-Level Languages (LLL):**
    - Closer to machine code and hardware.
    - Provide minimal abstraction, offering direct control over hardware.
    - Examples include Assembly language and machine code.

1. **Syntax and Readability:**
   * **High-Level Languages:**
     + Use natural language elements and clear syntax.
     + Easier to learn, understand, and maintain.
     + Suitable for complex applications and large codebases.
   * **Low-Level Languages:**
     + Use mnemonic codes and hexadecimal or binary numbers.
     + More challenging to read and write.
     + Ideal for system programming and performance-critical applications.
2. **Abstraction and Control:**
   * **High-Level Languages:**
     + Offer high abstraction from hardware.
     + Manage memory, processes, and hardware through built-in functions and libraries.
     + Prioritize developer productivity and ease of use.
   * **Low-Level Languages:**
     + Provide low abstraction, with direct access to memory and CPU instructions.
     + Offer fine-grained control over hardware resources.
     + Prioritize performance and resource efficiency.
3. **Performance and Efficiency:**
   * **High-Level Languages:**
     + Typically slower due to additional layers of abstraction.
     + More resource-intensive but improve developer efficiency.
     + Suitable for applications where development speed is more critical than execution speed.
   * **Low-Level Languages:**
     + Faster execution due to minimal abstraction.
     + More efficient use of resources.
     + Suitable for performance-critical applications like operating systems and embedded systems.