



INTERPRETER VS COMPILER



@EPYTHONLAB





- Both are programs that help convert the high level language (Source Code) into machine codes to be understood by the computers.



@EPYTHONLAB





- **Computer programs are usually written on high level languages.**
- **A high level language is one that can be understood by humans.**



@EPYTHONLAB





- However, computers cannot understand high level languages as we humans do.
- Where as computers can only understand the programs that are developed in binary systems known as a **machine code**.



@EPYTHONLAB





- To start with, a computer program is usually written in high level language described as a **source code**.
- These source codes must be converted into machine language and here comes the role of compilers and interpreters.



@EPYTHONLAB



INTERPRETER

VS

COMPILER



- Translates one statement of a program at a time into machine code.

- Where as compiler scans the entire program and translates it into machine code at once.



@EPYTHONLAB



INTERPRETER

VS

COMPILER



- Takes very less time to analyze the source code, however the overall time to execute the process is much slower

- Takes a lot of time to analyze the source code, however the overall time to execute the process is much faster.



@EPYTHONLAB



INTERPRETER

VS

COMPILER



- Does not generate an intermediary code
- Highly efficient in terms of memory

- Always generate an intermediary object code.
- It will need further linking, hence more memory is needed.



@EPYTHONLAB



INTERPRETER

VS

COMPILER



- Keeps translating the program continuously till the first error is confronted.
- If any error is spotted, it stops working and hence debugging becomes easy.

- Generates the error message only after it scans the complete program and hence debugging is relatively harder while working with a compiler.



@EPYTHONLAB



INTERPRETER

VS

COMPILER



- **Example: Python and Ruby**

- **Example: C and C++ programming languages**



@EPYTHONLAB





Thank You!



@EPYTHONLAB

