Translators

Every instruction, written in whatever programming language, has to be converted into **machine code** in order to be executed by the CPU

- different CPUs have a different set of machine code instructions they understand

This requires the use of a **translator**.

Translators

Three types of translator:

- assembler
- compiler
- interpreter

Translators

Assemblers:

- used to turn assembly code into machine code
- 1:1 correspondence, so each line of assembly code = 1 machine code instruction
- all code assembled in one go
- efficient use of hardware and memory - so run quickly

Assemblers work with one set of processors

Translators

C and C++ are compiled languages

Compilers:

Used to turn a high level language into machine code (it **compiles** the code)

- breaks instructions down into small steps
- translates the **whole program** before running it - which takes time
- creates an executable file (in machine code) which can be run by an end user
- exe files run quicker as the translation is already done
- end user doesn’t require the compiler - just the exe file
- end user can’t see (or change, or copy, or steal) the source code

Translators

Python is an interpreted language

Interpreter:

Used to work through a high-level program creating code (it **interprets** the code)

- translates the program line by line as it is running it
- breaks each instruction down into small steps
- some of the program may work before a crash - useful for development testing
- no exe file - so end user requires the interpreter
- interpreter needs to run each time the program runs - takes longer then an exe files
- end user needs the interpreter
- end user **can** see (change, copy, or steal) the source code

Translators

Interpreter:

Does **not** directly create machine code.

Works through the code using machine code subroutines within its own code to carry out commands.

Translators

Assembler - converts assembly code into machine code

Compiler - fully converts high-level code into an executable machine code file

Interpreter - works through line by line converting high-level code whilst running the program