Q

What does a translator do?

A

A translator translates/converts source code into machine code (binary)

Q

Why are translators required?

A

Translators are required because without them, a CPU will be unable to execute a program's instructions. CPU's can only execute machine code, they don't understand anything else. If you supply a CPU with source code, it will be unable to make sense of it! We therefore need translators to translate source code into machine code

Q

What 3 types of translator are there?

A

1. Compiler
2. Interpreter
3. Assembler

Q

What is the difference between a COMPILER and an INTERPRETER?

A

A compiler will translate source code into machine code in 'one go', producing an executable file, which can run independent of the translator.
An interpreter will translate, then execute source code, line by line. It therefore needs to be present in order for the program to run.

Q

What is the job of an assembler?

A

The 'assembler' translates assembly language into machine code.

Q

What is the difference between machine code and source code?

A

Machine code is binary code (0's and 1's. It is the only thing that a CPU can process. Source code is what is produced when writing in a high-level language (like python).

Q

What is the difference between assembly language and source code?

A

Assembly language is low level code (close to machine code) and made up of the few instructions that the CPU can carry out directly (e.g. ADD, SUB etc.). Each piece of assembly code represents a binary instruction.
Source code is high level (close to the English language) and has many different possible instructions. It makes use of keywords and requires either a compiler or interpreter to translate the code into a form that the CPU can understand.

Q

Why do people code programs using high level languages as opposed to machine code?

A

People write code in high level languages because they are closer to the English language and therefore more natural to use for humans. Machine code on the other hand consists of 0s & 1s and it is therefore very difficult for a human to remember the binary equivalents of the various instructions that would be required to write a program.

Q

Why do some developers choose to write in LOW LEVEL languages?

A

Because each piece of low-level code (i.e. assembly language) represents a single instruction, a programmer can control precisely the actions of the CPU and how it uses the RAM. This means that programmers can write highly efficient programs, which will run faster and uses memory more resourcefully.

Q

What does IDE stand for?

A

Integrated Development Environment

Q

Explain 3 features of an IDE.

A

Any 3 from:
Source Code Editor – allowing the writing and editing of code Interpreter – allows source code to be translated into machine code one line at a time for testing
Automation Tools – automate tasks such as finishing off key words and indenting on your behalf
Debugger – identifies logic and syntax errors and shows where they are.
Compiler – converts entire source code into machine code so it can be run as its own individual program file.
Auto-Documentation – stores lists of variables, modules, functions calls etc which are documented for other programmers.

Q

How can breakpoints and steppers help a developer debug their code?

A

To find logic errors, it can be useful to watch the program execute its code, one line at a time. This can be done using breakpoints and stepping. A breakpoint is a marker added to the code, which stops the program running when it reaches the marker. At this point stepping can be carried out which is where the code is executed one line at a time which helps the programmer check the actual logic of the code.

Q

What is 'auto-documentation'?

A

Auto-Documentation:
Stores lists of variables, modules, and functions calls etc., which are documented for other programmers.

Q

What is a 'debugger'?

A

Debugger:
Identifies logic and syntax errors and shows where they are.

Q

Which translator is useful at finding syntax errors and how does it help?

A

An interpreter is more useful at finding syntax errors because as it translates code line by line, it will be able to run the program until it finds an error and therefore can highlight the moment the error occurs more clearly to the programmer.

Q

Which translator enables a program to run the fastest? Explain your answer.

A

Although a compiler will take longer to initially translate a program's code, the complied program will run faster than an interpreter will, as an interpreter translates the code line by line at run-time, which is a more time-consuming process.