OCR Computer Science GCSE
2.5 – Programming languages and Integrated Development Environments
Advanced Notes

2.5.1 Languages

Programming languages are used to write instructions that computers can execute. They fall into two main categories:

- High-level languages (e.g. Python, Java, C#)
- Low-level languages (e.g. Assembly language, Machine code)

High-level languages

Designed for humans to read and write, with instructions similar to structured English (such as `print` and `while`). Most computer programs are written using high-level languages. Examples: Python, C#.

Advantages	Disadvantages
Easy for humans to understand and debug as the instructions are closer to English	Slower to execute than low-level languages
Programs written are portable between different hardware, since they can be translated into machine code for each specific type of processor	Must be translated into machine code, which can be less efficient than if it was originally written as machine code

Low-level languages

Closer to machine code (binary). Examples: Assembly language, Machine code

Advantages	Disadvantages
Faster and more efficient to execute	Hard to read and write
Gives more control over hardware, and direct control of the registers	Not portable – specific to one type of processor

Translators

Computers only understand machine code, so all programs written in high-level or assembly languages must be translated before they can be executed. Machine code is expressed in binary and is specific to a processor or family of processors.

Types of translators

There are two types of program translator: compilers and interpreters.

Compilers

A compiler can be used to translate programs written in high-level languages like C# and Python into machine code. Compilers take a high-level program as their source code, check it for any errors and then translate the entire program at once. If the source code contains an error, it will not be translated. Because compilers produce executable files for the machine they were compiled on, they are said to be platform specific.

Once translated, a compiled program can be run without needing to be recompiled. This is not the case with interpreters.

Interpreters

An interpreter translates high-level languages into machine code and executes it line-by-line. Interpreters do not generate machine code directly - they call appropriate machine code subroutines within their own code to carry out statements.

Rather than checking for errors before translation begins (as a compiler does), interpreters check for errors as they go. This means that a program with errors in can be partially translated by an interpreter until the error is reached.

When a program is translated by an interpreter, both the program source code and the interpreter itself must be present. This results in poor protection of the source code compared to compilers which make the original code difficult to extract.

Comparison of compilers and interpreters

Compiler	Interpreter
Checks source code for errors line-by-line before beginning translation	Translation begins immediately
Entire source code translated at once	Each line is checked for errors and then translated sequentially
No need for source code or compiler to be present when the translated code is executed	Both the source code and the interpreter must be present when the program is executed
Protects the source code from extraction	Offers little protection of source code

2.5.2 The Integrated Development Environment (IDE)

The need for an Integrated Development Environment

Developing programs can be difficult, especially having to write clear and maintainable code. IDEs exist to provide programmers with features which make writing code much easier, these include:

- Editors
- Auto-indentation
- Auto-suggestion
- Auto-correction
- Colour coding
- Line numbers
- Debugging tools
- Variable tracing
- Interpreters (type of translator)

Editors provide the platform for programmers to write and develop code, and contain features such as line numbering, colour coding, auto-indent, amongst others.

Error diagnostics help identify errors made when writing code. An IDE may highlight parts of code, such as missing brackets, speech marks or colons, as well as point out the line number on which the error is taking place. Some IDEs may also give possible ways of fixing the error.

Run-time environments allow programs to run on a computer it may not have been designed to run on. This is done by creating a virtual machine, which emulates a different computer system so that the program may run on it.

Translators are used to convert high-level languages, such as program code (Python) into machine code so that the program can be run by the processor.