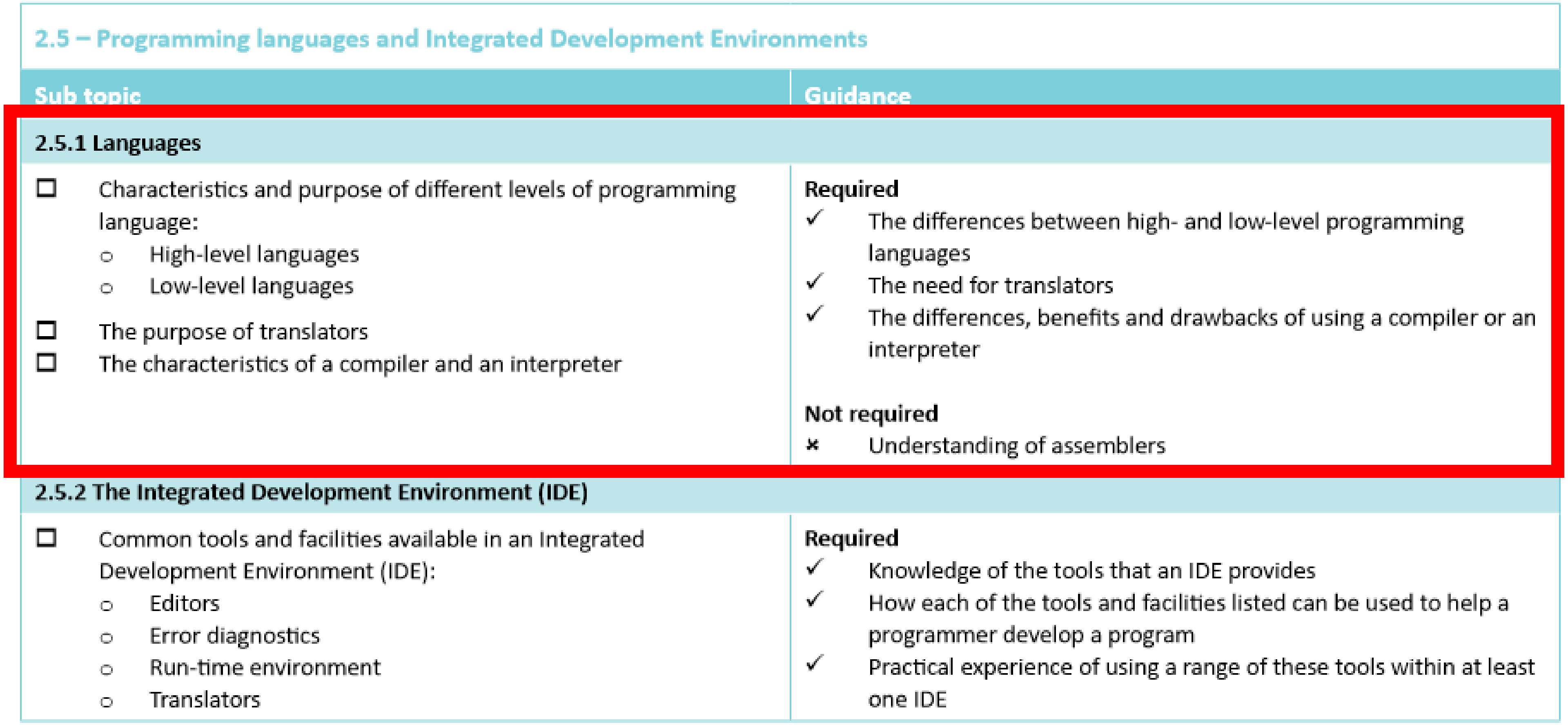
J277 - 2.5 Programming languages and IDEs



2.5 – Programming languages and Integrated Development Environments

Sub topic	Guidance
2.5.1 Languages	
☐ Characteristics and purpose of different levels of programming language: ○ High-level languages ○ Low-level languages ☐ The purpose of translators ☐ The characteristics of a compiler and an interpreter	**Required** ✓ The differences between high- and low-level programming languages ✓ The need for translators ✓ The differences, benefits and drawbacks of using a compiler or an interpreter **Not required** ✗ Understanding of assemblers
2.5.2 The Integrated Development Environment (IDE)	
☐ Common tools and facilities available in an Integrated Development Environment (IDE): ○ Editors ○ Error diagnostics ○ Run-time environment ○ Translators	**Required** ✓ Knowledge of the tools that an IDE provides ✓ How each of the tools and facilities listed can be used to help a programmer develop a program ✓ Practical experience of using a range of these tools within at least one IDE

J277 - 2.5 Programming languages and IDEs

Characteristics and purpose of different levels of programming language

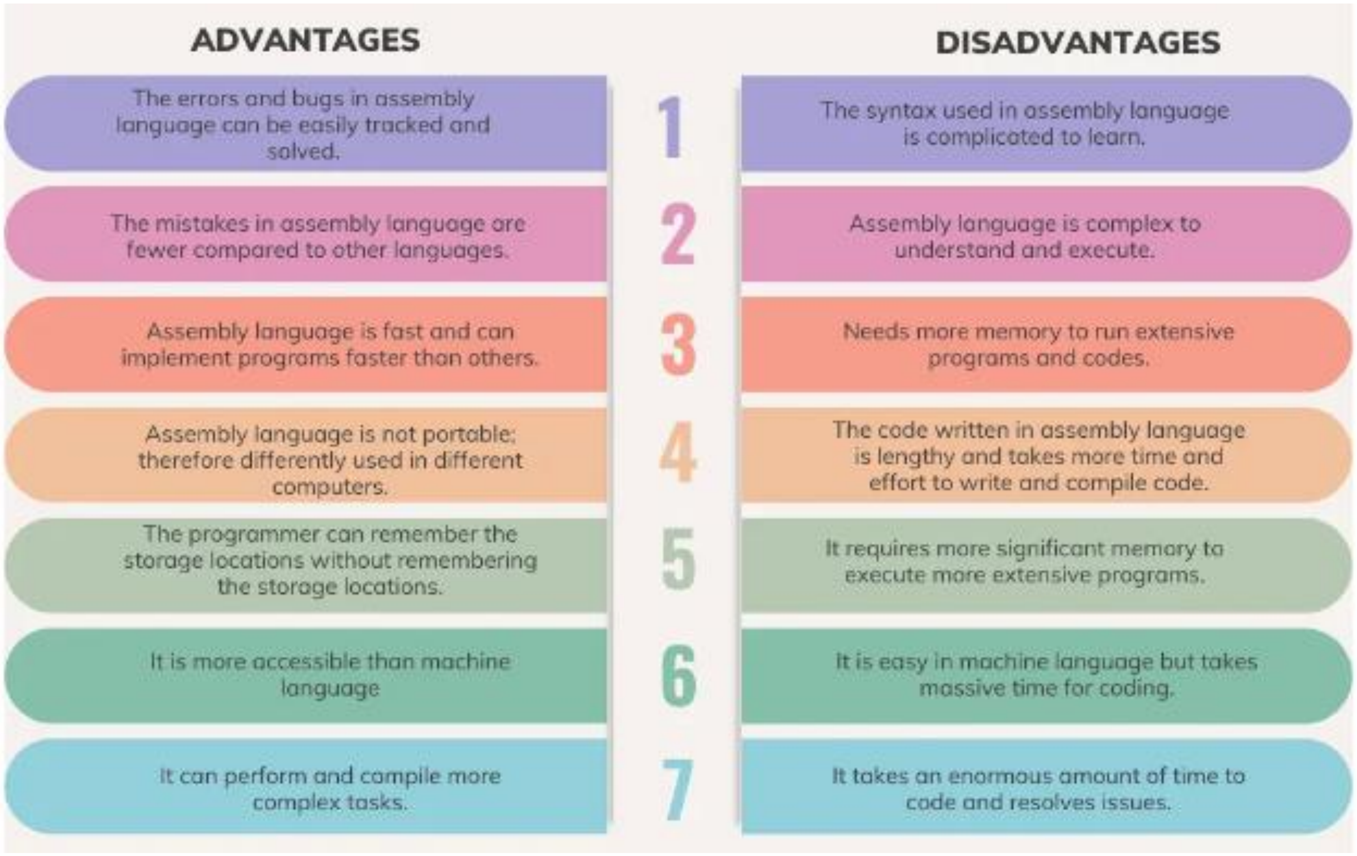
	High level language	**Low level language**
Example language:	Python	Assembly
Purpose:	Makes writing of computer programs easier by using commands that are similar to English language.	Used for embedded systems and device drivers where instructing the hardware directly is necessary.
Characteristic 1 Instructions to machine code:	One instruction translates to many machine code instructions.	One instruction translates to one machine code instruction.
Characteristic 2 Types of processors:	Code will run on different processors.	The code will work on one type of processor.
Characteristic 3 Data structures:	The programmer has lots of data structures to use.	The programmer works with the memory directly.
Characteristic 4 Ease of coding:	Code is quicker and easier to understand and write.	Code is much harder to understand and write.
Characteristic 5 Memory efficiency:	Less memory efficient.	More memory efficient.
Characteristic 6 Speed of execution:	Code is slower to execute.	Code is faster to execute.

J277 - 2.5 Programming languages and IDEs

Characteristics and purpose of different levels of programming language – LOW LEVEL LANGUAGE

100100
10000000011
101010101010
0000000001111

Low level language



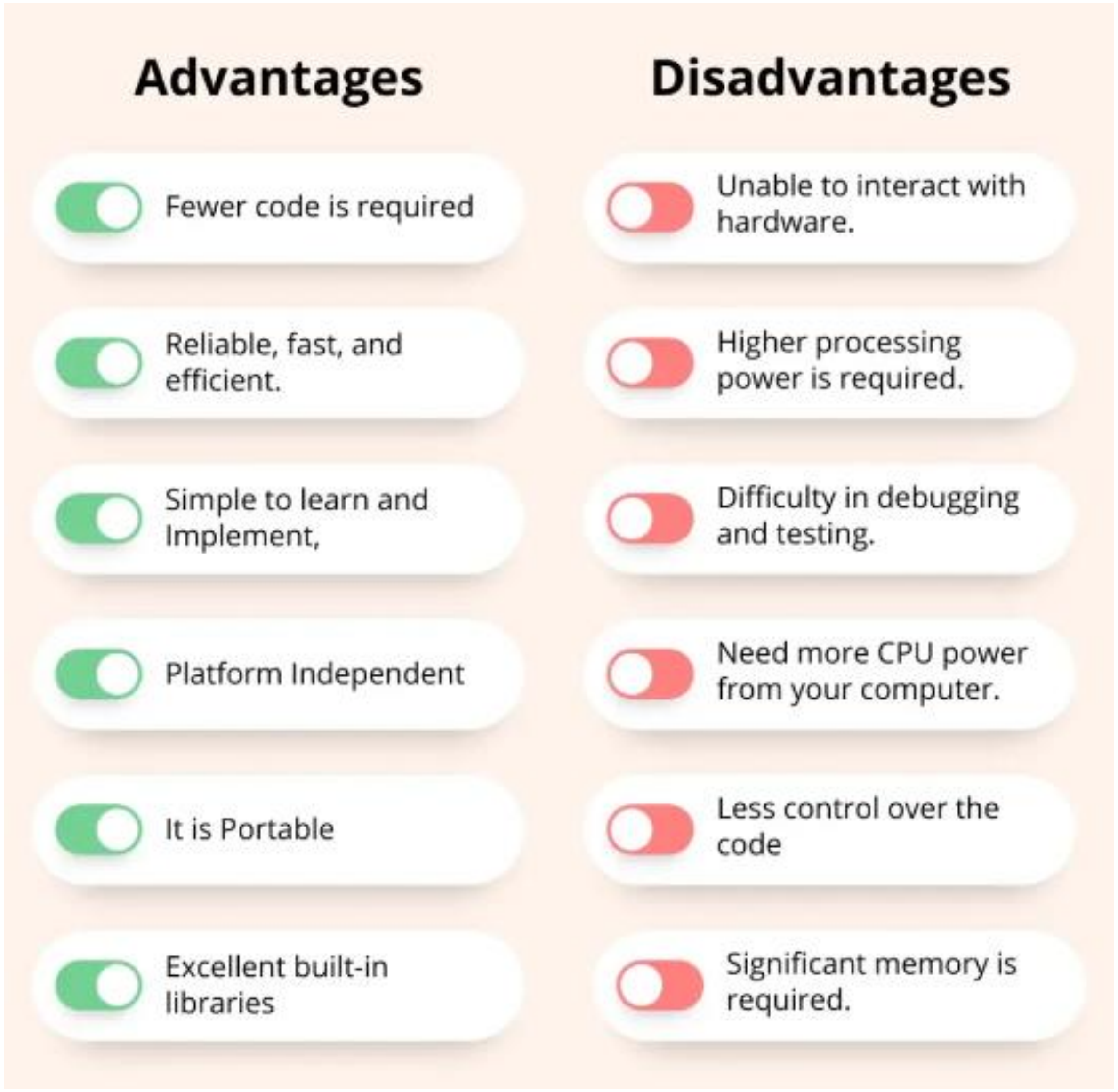
ADVANTAGES		DISADVANTAGES
The errors and bugs in assembly language can be easily tracked and solved.	1	The syntax used in assembly language is complicated to learn.
The mistakes in assembly language are fewer compared to other languages.	2	Assembly language is complex to understand and execute.
Assembly language is fast and can implement programs faster than others.	3	Needs more memory to run extensive programs and codes.
Assembly language is not portable; therefore differently used in different computers.	4	The code written in assembly language is lengthy and takes more time and effort to write and compile code.
The programmer can remember the storage locations without remembering the storage locations.	5	It requires more significant memory to execute more extensive programs.
It is more accessible than machine language	6	It is easy in machine language but takes massive time for coding.
It can perform and compile more complex tasks.	7	It takes an enormous amount of time to code and resolves issues.

J277 - 2.5 Programming languages and IDEs

Characteristics and purpose of different levels of programming language – HIGH LEVEL LANGUAGE

```
if(i<5)
{
printf("I am true block ");
}
else{
printf("I am false block");
}
```

High level language



Advantages	Disadvantages
Fewer code is required	Unable to interact with hardware.
Reliable, fast, and efficient.	Higher processing power is required.
Simple to learn and Implement,	Difficulty in debugging and testing.
Platform Independent	Need more CPU power from your computer.
It is Portable	Less control over the code
Excellent built-in libraries	Significant memory is required.

J277 - 2.5 Programming languages and IDEs

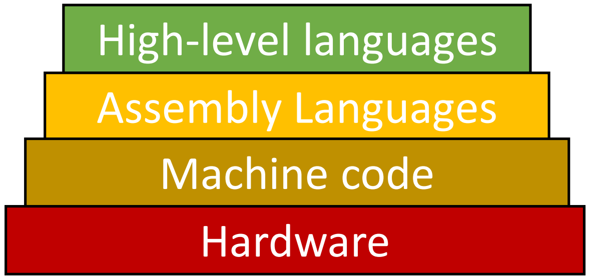
What is the relationship between machine code and assembly language?

Low-level language :	**Low-level languages** are languages that sit close to the computer's **instruction set**. An instruction set is the set of instructions that the processor understands. Two types of low-level language are: **Machine code & Assembly language.**
Machine code :	Machine code is the set of instructions that a CPU understands directly and can act upon. A program written in machine code would consist of only 0s and 1s - **binary**.

The difference between machine code and assembly languages:

- Assembly languages are written by programmers in assembly code.
- It is often used to develop software for embedded systems and controlling specific hardware components.
- They first have to be translated by an assembler into machine code before being run.
- Once run the actual CPU executes the machine code.
- Machine code is specific to each type of processor.
- The relationship between assembly code and machine code is one-to-one.
- This means that one line of assembly code translates into one specific line of binary machine code.

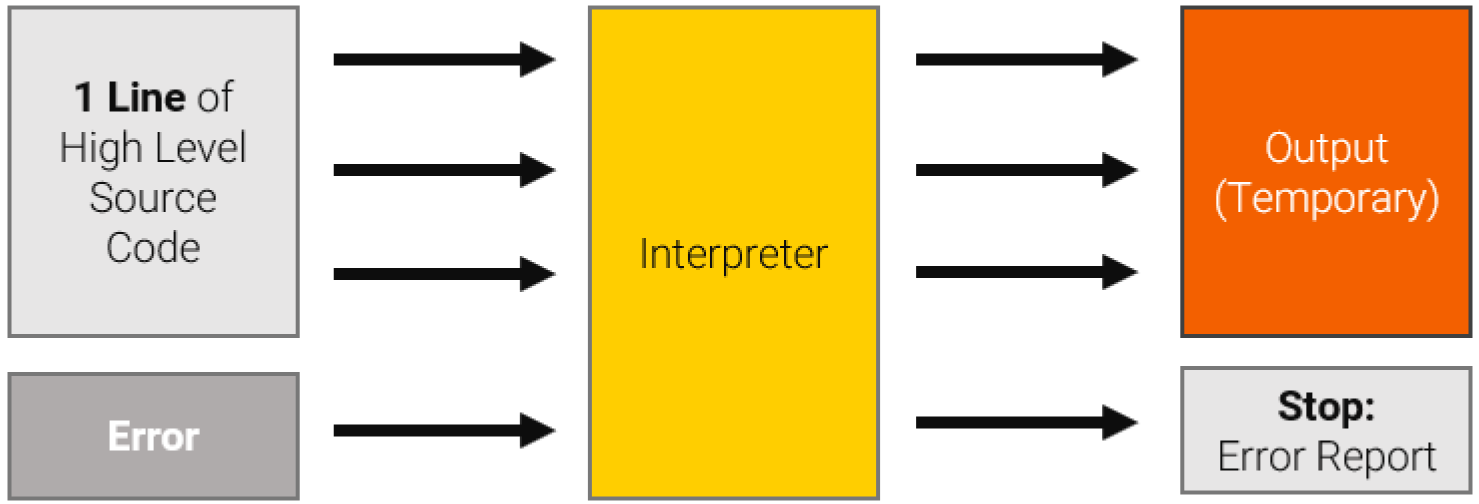
Hierarchy of languages:



J277 - 2.5 Programming languages and IDEs

The purpose of translators

Translators are needed to translate programs written in high level languages into the machine code that a computer understands. Tools exist to help programmers develop error-free code.

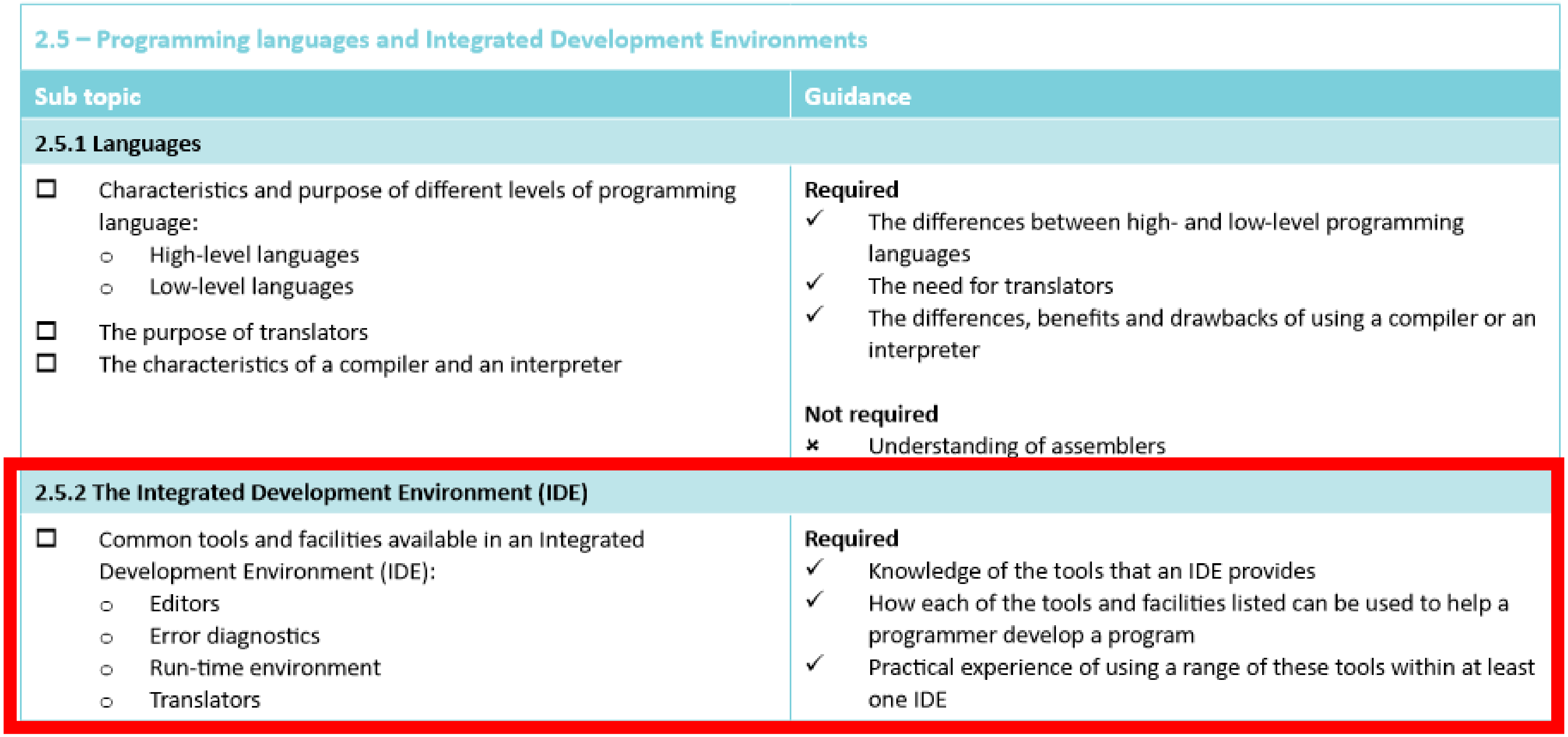


J277 - 2.5 Programming languages and IDEs

The characteristics of a compiler and an interpreter

	Compiler	Interpreter
Description:	Translates source code from a high level language into object code and then into machine code to be processed by the CPU.	Translates source code from a high level language into machine code to be processed by the CPU.
Feature:	The whole program is translated to machine code before it is run.	The program is translated line by line as the program is running.
Ease of writing code:	Easier to write code as it is close to English language, but the program will not run with syntax errors in the code.	Easy to write code as it is close to English language. The program will run and stop when it finds a syntax error.
Impact of changing code:	Needs to be recompiled.	Does not need be recompiled and it is easy to try out commands.
Designed for a specific type of processor:	Yes	No
Need for translation software at run-time:	No	Yes
Speed of code execution:	Quick	Slow
Optimised code:	Yes	No
Source code is kept secret:	Yes	No

J277 - 2.5 Programming languages and IDEs



2.5 – Programming languages and Integrated Development Environments

Sub topic	Guidance
2.5.1 Languages	
☐ Characteristics and purpose of different levels of programming language: ○ High-level languages ○ Low-level languages ☐ The purpose of translators ☐ The characteristics of a compiler and an interpreter	**Required** ✓ The differences between high- and low-level programming languages ✓ The need for translators ✓ The differences, benefits and drawbacks of using a compiler or an interpreter **Not required** ✗ Understanding of assemblers
2.5.2 The Integrated Development Environment (IDE)	
☐ Common tools and facilities available in an Integrated Development Environment (IDE): ○ Editors ○ Error diagnostics ○ Run-time environment ○ Translators	**Required** ✓ Knowledge of the tools that an IDE provides ✓ How each of the tools and facilities listed can be used to help a programmer develop a program ✓ Practical experience of using a range of these tools within at least one IDE

J277 - 2.5 Programming languages and IDEs

Common tools and facilities available in an integrated development environment (IDE)

Writing large programs can be a complex task. To help the programmer write clear, maintainable code, various tools exist.

Feature	Description
Text editor	Allows you to add and edit code as well as to insert comments.
Runtime environment	Runs your program by converting your source code into machine code in order for it to be executed by the CPU.
Syntax checking	Checks for any potential syntax errors in line with the rules of the language you are writing in. This helps to avoid common syntax errors appearing at the point when code is executed.
Keyword highlighting	Colour codes command words, variables, and data types to make your code more readable and easier to debug.

Feature	Description
Debugging tools	Tools that help you to detect and locate errors so you can fix them.
Break point	A debugging tool that enables you to stop the program execution at a specific point to enable you to see the values of the variables. Some IDEs also allow you to step through the code line by line to trace the values of the variables.
Memory inspector	Displays the contents of memory so that you can see how it is being used by the program in order to help debug problems such as memory leak.
Threading	Debugging tool that allows you to see the threads currently running. The inspector allows you to suspend, resume, and see the status of each thread being executed by your program.