



# OCR GCSE Computer Science



Your notes

## Programming Languages & Integrated Development Environments (IDEs)

### Contents

- \* Levels of Programming Languages
- \* Translators, Compilers & Interpreters
- \* Tools & Facilities in IDEs



Your notes

## Levels of Programming Languages

- Since the invention of the computer, people have needed to learn **how to communicate** with them **using programming languages**
- Early computers were **complex** and instructions would have to be in **written in binary code**, 0s and 1s
- This process was **slow**, taking days to program simple tasks
- Over time, new generations of programming languages have enabled people to become **faster** and **more efficient** at writing programs as they **resemble human language**
- Generations of programming languages can be split in to two categories:
  - **Low-level**
    - **First generation**
    - **Second generation**
  - **High-level**
    - **Third generation**

## Low-Level Languages

### What is a low-level language?

- In GCSE Computer Science, a low-level language is a programming language that **directly translates to machine code** understood by the processor
- Low-level languages allow **direct control over hardware** components such as **memory** and **registers**
- These languages are written for **specific processors** to ensure they embed the correct machine architecture

### First generation

- **Machine code** is a first-generation language
- Instructions are **directly executable by the processor**
- Written in **binary code**

### Second generation

- **Assembly code** is a second-generation language
- The code is written using **mnemonics**, abbreviated text commands such as **LDA (Load)**, **STA(Store)**



Your notes

- Using this language programmers can write **human-readable programs** that correspond **almost exactly to machine code**
- **One assembly language instruction** translates to **one machine code instruction**
- **Needs to be translated** into machine code for the computer to be able to execute it

Advantages	Disadvantages
Complete control over the system components	Difficult to write and understand
Occupy less memory and execute faster	Machine dependent
Direct manipulation of hardware	More prone to errors
	Knowledge of computer architecture is key to program effectively

## High-Level Languages

### What is a high-level programming language?

- A high-level programming language uses **English-like statements** to allow users to program with easy to use code
- High-level languages allow for clear **debugging** and once programs are created they are **easier to maintain**
- High level languages were needed due to the **development of processor speeds** and the **increase in memory capacity**
- **One instruction** translates into **many machine code instructions**
- **Needs to be translated** into machine code for the computer to be able to execute it
- Examples of high-level languages include:
  - **Python**
  - **Java**
  - **Basic**
  - **C+**



Your notes

Advantages	Disadvantages
Easier to read and write	The user is not able to directly manipulate the hardware
Easier to debug	Needs to be translated to machine code before running
Portable so can be used on any computer	The program may be less efficient
One line of code can perform multiple commands	



Your notes

## Translators, Compilers & Interpreters

# Translators, Compilers & Interpreters

## What is a translator?

- A translator is a program that **translates** program **source code** into **machine code** so that it can be executed directly by a processor
- **Low-level languages** such as **assembly code** are translated using an **assembler**
- **High-level languages** such as **Python** are translated using a **compiler** or **interpreter**

## What is a compiler?

- A compiler translates high-level languages into machine code **all in one go**
- Compilers are generally used when **a program is finished** and has been checked for syntax errors
- Compiled code can be **distributed (creates an executable)** and run **without the need for translation software**
- If compiled code contains any errors, after fixing, it **will need re-compiling**

Advantages	Disadvantages
Speed of execution	Can be memory intensive
Optimises the code	Difficult to debug
Original source code will not be seen	Changes mean it must be recompiled
	It is designed solely for one specific processor

## What is an interpreter?

- An interpreter translates high-level languages into machine code **one line at a time**
- Each line is executed after translation and if **any errors are found**, the **process stops**
- Interpreters are generally used when a program **is being written** in the development stage
- Interpreted code is more **difficult to distribute** as **translation software is needed for it to run**



Your notes

Advantages	Disadvantages
Stops when it finds a specific <b>syntax error</b> in the code	Slower execution
Easier to debug	Every time the program is run it has to be translated
Require less RAM to process the code	Executed as is, no optimisation



### Worked Example

A computer program is written in a high-level programming language.

(a) State why the computer needs to translate the code before it is executed. [1]

(b) Either a compiler or an interpreter can translate the code. Describe two differences between how a compiler and an interpreter would translate the code. [2]

#### How to answer this question

- (a) what time of language does a computer understand?
- (b) the keyword is 'how'

#### Answer

(a)

- To convert it to binary/machine code
- The processor can only understand machine code

(b)

- Compiler translates all the code in one go...
- ...whereas an interpreter translates one line at a time
- Compiler creates an executable...
- ...whereas an interpreter does not/executes one line at a time
- Compiler reports an error at the end...
- ...whereas an interpreter stops when it finds an error



Your notes

## Tools & Facilities in IDEs

# Tools & Facilities in IDEs

## What is an IDE?

- An Integrated Development Environment (IDE) is **software** designed to make **writing high-level languages** more **efficient**
- IDEs include tools and facilities to make the **process of creating/maintaining code easier**, such as:
  - Editor
  - Error diagnostics
  - Run-time environment
  - Translators

## Editor

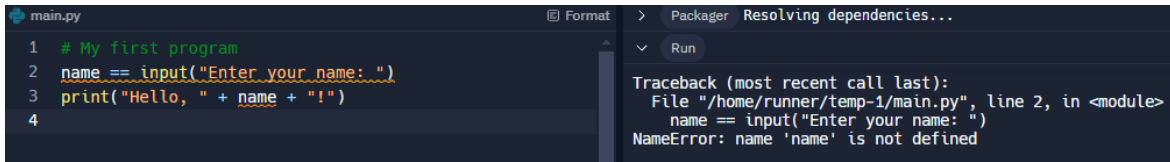
```
main.py > ...  
1 # My first program  
2 name = input("Enter your name: ")  
3 print("Hello, " + name + "!")  
4
```

- An editor gives users an environment to **write, edit and maintain** high-level code
- Editors can provide:
  - **Basic code formatting tools** - changing the font, size of the font and making text bold etc
  - **Coloured keywords** - using colour to make it easier to identify keywords, for example 'print', 'input' and 'if' in Python
  - **Code editing** - auto-completion and auto-correction of code, bracket matching and syntax checks
  - **Commenting code** - allows sections of code to be commented out easily to stop it from being run or as comments on what the program is doing

## Error-diagnostics

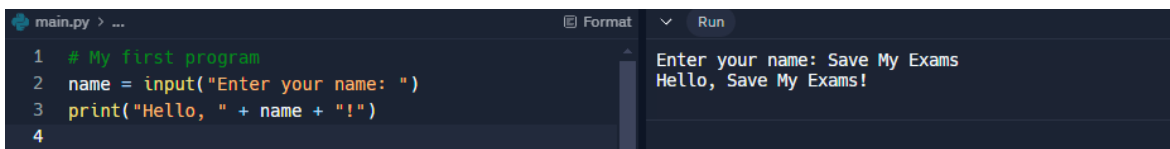


Your notes



- Tools that help to **identify, understand** and **fix errors** in code, such as:
  - Identifying errors** - highlight particular areas of code or provide direct error messages where the error may have appeared e.g. indentation errors etc
  - Debugger** - provide a 'step through' command which provides step by step instructions and shows what is happening to the code line by line, useful for finding **logic errors**

## Run-time environment



- Gives users the ability **to run** and see the **corresponding output** of a high-level language

## Translator

- Built in to **compile** or **interpret** code without the need for an extra piece of software



### Worked Example

James creates his programs using an Integrated Development Environment (IDE).

Describe two tools or facilities that an IDE commonly provides [4]

#### How to answer this question

- Name two tools or facilities
- For each tool or facility, give a reason why its helpful

#### Answer

- Editor**
  - ...to enable program code to be entered/edited
- Error diagnostics/debugging**





Your notes

- ...to display information about errors (syntax/run-time) / location of errors
- ...suggest solutions
- **Run-time environment**
- ...to enable the program to be run
- ...check for run-time errors / test the program
- **Translator / compiler / interpreter**
- ...to convert the high-level code into machine code / low-level code / binary
- ...to enable code to be executed / run



## Examiner Tips and Tricks

There are 4 main tools/features of an IDE you need to know.

**Editor, error diagnostics, run-time environment and translator**

If a question asks for 2 tools/features, do not give multiple examples from the same tool/feature, e.g. coloured keywords and commenting are both examples of the same tool/feature (**editor**)