ICT Level 2 – Key Terms 4

1 of 18 – Welcome

Welcome to this session on key terms. In this session we will go back over some of the key terms.

In this session you will:

* Revisit a number of key terms from recent sessions
* Consider their specific meanings
* Know how to use them in relation to software and programming

2 of 18 – Introduction to key terms

**What sort of key terms will we be looking at?**

In earlier sessions we thought about **the types of, and uses for, specific software**, and we looked at different **programming concepts** that can be used to design this software. During these sessions a number of key terms and important concepts were introduced, such as:

* Off-the-shelf software
* Operating systems
* High and low-level programming languages

There are a number of different types of high and low-level programming languages available.

3 of 18 – Software: a reminder!

**Remember:** when someone is purchasing software they have to choose between **off-the-shelf** software and **custom-made** software. Each type of software comes with its own good and bad points, which users also need to remember.

Off-the-shelf software is easy to get hold of and it usually has been well tested for bugs and glitches before being made available. However, if a computer user needs software for a very specific task or job, then off-the-shelf software might not be able to meet that specific need.

Custom-made software can meet specific needs as it is purpose-built with a certain aim or task in mind, which is one of the advantages to using it. However, it is not as well tested as the off-the-shelf alternatives and it can take a long time (months, in some cases) to get this software up and running (not to mention the extra money required for this product to be made).

4 of 18 – Operating system: what is it?

**Remember:** the operating system (sometimes called the OS) is a piece of software within a computer, and its main job is to control parts of the hardware.

Nowadays, computer users have the option of interacting directly with their operating system, using things like a graphic user interface (which you might remember from an earlier session, too).

The main day-to-day job of the operating system is to act as a manager to other software and programs (sometimes called applications) that are installed on the computer.

The operating system can authorise a number of different functions, and it can send messages to other areas containing instructions – or, if it makes for a smoother running order on the computer, the OS can distribute some of its work to other computer parts as well.

5 of 18 – Utility applications: what are they?

**Remember:** utility applications are a type of supportive software. These applications are typically used to help another application, or to help a computer function (which means helping out the operating system sometimes).

There are two different types of utility applications available to computer-users; those that are nonessential (meaning they are not a requirement) to the operating system, although they could be added to help.

Alternatively, the second type is something that is specialised but limited in terms of what it can achieve on the computer system.

While their importance can vary from one application type to another, the overall purpose remains the same, as all types are used to help your computer perform better.

6 of 18 – Utility applications: some examples

You might remember the terms **disk defragmenters**, **firewalls**, and **anti-virus** programs from an earlier session. These are all examples of utility applications that can be added to a computer to not only help it work better, but also to protect it from unfriendly software, or malware.

Utility applications can be used for a number of other computer-related tasks too, such as:

* Backing up – which is particularly useful if software crashes and needs to be returned to an earlier state for a user to access certain information
* Encryption
* Formatting data
* Deleting data

7 of 18 – Command line interface: what is it?

Think back to our earlier talk of operating systems and user interface. You might have recognised the term graphical user interface (GUI), and this is usually discussed alongside command line interface (CLI), as both of these interfaces are systems that allow a computer user to interact – or rather, interface – with a computer system.

A command line interface is a little outdated now, as graphical user interfaces tend to offer users more options and make it easier for users to navigate their systems.

Before GUI, computer users instead had to use CLI systems which worked by asking users to type in their instructions or commands, which the computer would then respond to, before another instruction was typed in by the user – and so on.

As you can imagine, it was a little slower than our modern methods, but CLI was an impressive piece of technology when it was first released.

8 of 18 – Hierarchical structure: what does this mean?

The hierarchical structure within a computer system is a framework that maps out how different pieces of software interact with each other.

Hierarchical structures start with the most important parts at the top, and then branch off and down into other areas or, in this case, other computer parts. The list below shows the hierarchical structure within a computer system (starting with the most important parts at the top of the list):

1. **High-level** and **low-level** programming languages
2. **Machine code** – programs are mixed with machine code
3. **Computer** **hardware**
4. **User interface** – machine code is performed by the CPU, which controls the hardware, and displays the proper user interface

These are a particularly useful tool to use when a computer-user is trying to determine which parts of their computer are interacting – and also what jobs these parts are, or at least should be, completing.

By ranking different computer components – or ordering them, you might say – from top to bottom in the list, it is also easier to track problems within the computer system.

The structure outlines clear parts that are performing different jobs and giving the overall system different information, so it is a good tool for organising your computer and then checking each component to ensure everything is performing as it should be in each step of the structure.

9 of 18 – High-level programming language: a reminder!

**Remember:** when programmers are writing a new program, programming languages have to be used. These are converted, or translated, at a later point into machine code. Remember too that machine code is the language understood and used by computer systems.

High-level programming languages are often used by programmers because they are **quick** and **easy** to get used to. They are also **closer to our natural human language** than other alternatives (like low-level programming languages, for example).

High-level programming languages are also known to be very powerful, and so can be used to write a whole host of different types of programs.

However, it is also important to remember that high-level programming languages might make for a more basic piece of software (again, this is in comparison to low-level options).

10 of 18 – Low-level programming language: a reminder!

Low-level programming languages can be a **little more difficult** for programmers to use, as these languages are **closer to computer language**. However, this does mean that the translation time is cut down, in comparison to high-level programming languages.

Another tool that can be used to make low-level programming languages a little easier to use is an **assembler**. This is a tool that can translate low-level languages straight into machine code.

However, there are some low-level programming languages that can be immediately understood by a computer system before a translation has even taken place.

While they do have many good points, it is important to remember that low-level programming languages can be **intricate,** and they require a **lot of detail** to get the finished work just right.

11 of 18 – Mnemonics: what are these?

When a programmer is using an assembler – remember, this is the technology used to convert low-level programming languages into machine code – they have the option of using the assembler not only to translate their work, but also to scan it for any problems.

When a programmer is writing code for software, they will use mnemonics. These are abbreviations that signal certain instructions – for example, if a programmer were to use the mnemonic MOV, this would indicate a move of some sort in the software.

Each mnemonic will translate to a specific instruction in the machine code, before this is loaded into the computer system – which is why they are so important, as they have a significant influence over whether software runs correctly or not.

Fortunately, assemblers are now able to scan for errors in these mnemonics, meaning they can be corrected before the machine code is completed.

12 of 18 – Question 1

Software can either be off-the-shelf or custom-made; both types have their disadvantages. What are the **disadvantages** of using custom-made software?

Choose all that apply:

1. It is not as well tested as others
2. It might not be useful for specific tasks
3. It can take months to build
4. It is more expensive
5. It is a little cheaper

The correct answers are A, B and C, it is not as well tested as others, it might not be useful for specific tasks and it can take months to build.

13 of 18 – Question 2

What does the operating system of a computer do?

1. It acts as a manager to other pieces of software by authorising and distributing different functions
2. It directly interacts with the user by using voice control features, but it cannot really control anything else within the machine

The correct answer is A, it acts as a manager to other pieces of software by authorising and distributing different functions.

14 of 18 – Question 3

Indicate whether the following statements are true or false.

Utility applications are supportive software that is never specialised but can sometimes be useful.

True

False

The correct answer is: False

Anti-virus programs and backing up data are two useful examples of what utility programs can do.

True

False

The correct answer is: True

CLI is the abbreviation for Computer Line Interactions.

True

False

The correct answer is: False

CLI is now thought to be an outdated type of technology.

True

False

The correct answer is: True

15 of 18 – Question 4

Why are hierarchical structures useful for computer software?

1. Users can see what software is least important at the top of the structure
2. Users can see what software is responsible for what jobs
3. Users can see what software needs to be upgraded
4. Users can see what hardware is running slowly

The correct answer is B, users can see what software is responsible for what jobs.

16 of 18 – Question 5

Using the following choice of words; **understand**, **system**, **low-level**, **programming**, **computers**, **easier** and **human**, fill in the blanks for the paragraph below:

When programmers are designing new software, they tend to use one of two different types of **blank** languages: high-level or **blank**. High-level programming languages are a little **blank** to use as they are quite close to our natural **blank** speech, whereas low-level programming languages are closer to the language of **blank**. However, this does not mean that low-level programming languages are easier for a computer to **blank** when the software is added to the **blank**.

The correct paragraph should read:

When programmers are designing new software, they tend to use one of two different types of **programming** languages: high-level or **low-level**. High-level programming languages are a little **easier** to use as they are quite close to our natural **human** speech, whereas low-level programming languages are closer to the language of **computers**. However, this does not mean that low-level programming languages are easier for a computer to **understand** when the software is added to the **system**.

17 of 18 – End

Well done. You have completed this session on key terms.

In this session we have:

* Revisited a number of key terms from recent sessions
* Considered their specific meanings
* Observed how to use them in relation to software and programming

If you have any questions about any of these topics, make a note and speak to your tutor for more help.