ICT – Revision Session 3

1 of 19 – Welcome

Welcome to this session on revision (part 3).

By the end of this session, you will:

* Know what is meant by software
* Understand what operating systems are
* Have a basic knowledge of computer programming
* Know the difference between high level and low level programming languages

2 of 19 – Introduction to software

**What is software?**

Software is a broad term that covers a great number of computer programs that can be downloaded or installed onto a computer system.

Different types of software perform different jobs – which is why a computer system will typically have different software, or programs, installed and running at any given time.

When a computer user is looking to purchase a new type of software, they can choose between:

* Off-the-shelf software
* Custom-made software

3 of 19 – Reminder: off-the-shelf versus custom

The differences between off-the-shelf and custom software are as follows:

**Off-the-shelf software** refers to programs that have already been built, and are usually available from any technology retailer, meaning a computer user can buy this software quite easily.

This software is well tested, easy to purchase, and usually quite reliable. However, if a computer user has a very specific task to complete or a specific operation that they want to run on their computer, off-the-shelf software will not always be designed for these specific uses.

**Custom-made software** is software that is ordered and then developed with a specific need in mind. A computer user can approach a computer programmer and ask for software that performs X, Y, and Z tasks, and the software would be designed with these specific jobs in mind.

However, custom-made software is not as well-tested as off-the-shelf software, meaning there might be bugs or glitches in the system. Alongside this, it can also be quite an expensive and slow investment, as individuals have to wait weeks or months for the software to be written.

4 of 19 – Software installation

While software is an everyday part of using a computer system, adding new software or even updating older software is something that takes serious consideration. Users should consider a number of things, such as:

* Can the computer’s **hardware** handle the new or updated software?
* How **accessible** is the new software? Does it provide the right features?
* How **expensive** is the new software, or the software update?
* How **fast** is the new software? Is it likely to slow the computer down?
* Are there any **security** risks associated with this new software?

There are one or two issues that are associated with upgrading software too, which we will look at next.

5 of 19 – Upgrading software

Two things that users should consider, particularly when they are upgrading their software, is **incompatibility** and **loss of service**.

Incompatibility is something that occurs when the new software needs something that the computer system does not have. This could be enough storage space, enough free memory, or even fast enough hardware, meaning that the updated software will not run as well as the older version did. Users have to make sure that their system can provide whatever it is that the software needs.

Loss of service is another concern, which mostly impacts business users who are looking to update or upgrade their current software. Some software installations can take days to roll out to an entire network of computers. If the update or upgrade prevents users from accessing their computers for this amount of time, then that will have a negative impact on the company’s productivity and performance levels.

6 of 19 – Operating systems

Operating systems are installed in technology systems in order to keep an eye on how other elements – such as hardware, and other installed programs – are working. Operating systems (or, the OS) can be thought of as the programs that manage other programs.

Some of the main functions of an operating system are file management, hardware management, distributing resources, and security management.

Operating systems often come complete with a number of utility applications, too. Utility applications are small applications inside the OS that help to keep the computer system running smoothly. Examples of utility applications will be things such as anti-virus programs, firewalls, and disk defragmenters.

7 of 19 – Application software

Application software is the term used to describe applications – or rather, software – that has been written, developed, and completed. So this is the term that can be used to refer to the final application product.

Other examples of this, alongside utility applications, are productivity applications.

Productivity applications are small applications that can be added to a computer system in order to help users with day-to-day tasks.

Some common examples of productivity applications are word processors, which can be used for work (writing essays) or personal reasons (typing a letter to someone); another example would be a spreadsheet application, which can be used for monitoring finances or expenses.

8 of 19 – User interface

The user interface broadly refers to the parts of the computer that a user will interact with, referring to things like the layout of a desktop, for example, or how easy it is to navigate the file management system.

There are two main types of user interface available:

* A graphical user interface (GUI) which relies heavily on graphics, images, and icons, making it easy for a user to navigate their way around the computer system
* A command line interface (CLI) which relies heavily on users typing in their instructions, for the computer to then carry out a command, before the user must enter another instruction

A command line interface is thought to be quite an outdated software, which is why many computers now rely on a graphical user interface instead. GUI is faster and easier to navigate for many users, and it can also be added to over time, meaning new features are regularly made available for this type of system.

9 of 19 – Reminder: computer programming

Computer programming is the broad name given to the process of teaching a computer system how to do something new – i.e. a specialist programmer will program a new action or behaviour into a computer system, using specially written code that instructs the computer to perform certain instructions.

Computer programming can cater for a number of different user needs. Software can be written and programmed in to complete any number of jobs, whether it is for entertainment purposes – such as installing a new game onto a computer system – or whether programs are written with more practical or work-related purposes in mind.

There are a number of different options for how a computer programmer can piece their programs together, and this decision is usually influenced by the type of program that needs to be made.

10 of 19 – Programming options

Using imperative code – this is code that allows a computer programmer to break down a program into step-by-step processes – a programmer will choose the most appropriate programming style for the program that they are trying to develop.

They will often choose from:

* Procedural programs, which are programs made with a clear start and end point, although there can be multiple end points written into these programs now, too
* Event-driven programs, which are programs that are designed to respond to a certain event or action, e.g. a mouse click will trigger an action from the program
* Object-orientated programs, which work in a slightly more complicated way by grouping programs together like they are objects, rather than steps to be completed (do not worry too much about these for the time being though)

11 of 19 – Programming languages

Not only does a computer programmer have to decide on the structure that their program will follow, but they also have to decide which language they will use to write the program.

There are a number of different programming languages available and some are more or less appropriate than others, depending on the type of program that is being written, and also depending on the system that the program is likely to be run through (a personal computer or a MAC, for example).

Broadly speaking, programmers will choose between high level programming languages and low level programming languages (although there are number of individual language types that can be found under these main titles, too).

When making this decision, the programmer’s experience and expertise is something else that has to be considered, as some languages are easier than others.

12 of 19 – High-level programming languages

High level programming languages were originally designed in order to simplify the programming process, as they are thought to be slightly easier to use. Their user-friendly reputation comes from the fact that these languages are quite close to human speech – which not only makes them easier to use, but also easier to fix, should something go wrong.

Common examples of high level programming languages are:

* Visual Basic
* Java
* C++

While these language types (and a number of others that fall under the high level heading) are easier to use to begin with, they can become quite time-consuming. This is largely because computer systems do not understand these languages in their natural form, so high level programming languages have to be translated before they can be introduced to a computer.

13 of 19 – Low-level programming languages

Low level programming languages are a little more difficult to use, largely because they closely resemble machine code (remember this is the language that is used and understood by computer systems).

Programming in low level languages can take a little more time and expertise to get right; however, many low level languages benefit from the use of an assembler, which translates the language into machine code as it is written, making the process of program writing a little faster overall.

Low level programming languages have a reputation for making fast-acting programs, as they are thought to work slightly quicker. However, a common problem with these language types is that, because they are so intricate, they can be particularly difficult to fix if something goes wrong with them.

**Remember:** low level languages often require a certain number of specialist skills, which can take much longer to master than the high level language alternatives.

14 of 19 – Question 1

What are the disadvantages to custom-made software?

Choose all that apply:

1. It can take a lot of time to make
2. It may not fit a user’s needs
3. It can be quite expensive
4. It is not as well-tested for glitches
5. It is readily available from technology outlets

The correct answers are A, C and D, it can take a lot of time to make, it can be quite expensive, and it is not as well-tested for glitches.

15 of 19 – Question 2

Indicate whether the following statements are true or false.

When installing new software users must consider their hardware, security risks, and one or two other concerns.

True

False

The correct answer is: True

When a user is simply upgrading or updating software there is no risk associated with this.

True

False

The correct answer is: False

The operating system is an example of hardware that is used inside every computer.

True

False

The correct answer is: False

Utility applications and productivity applications are designed to help with everyday computer use.

True

False

The correct answer is: True

16 of 19 – Question 3

From the list below, what are the real programming options?

Choose all that apply:

1. Procedural programs
2. Start-and-finish programs
3. Event-driven programs
4. Object-orientated programs
5. Action-driven programs

The correct answers are A, C and D, procedural programs, event-driven programs and object-oriented programs.

17 of 19 – Question 4

High level and low-level programming languages have their own individual advantages.

Categorise the advantages below according to if they belong to **high-level** or **low-level** programming languages.

Takes less time to translate

Easy to use

Close to human speech

Takes less time to learn

Easy to fix

Closer to machine code

Makes faster programs

The correct answers are:

Easy to use, close to human speech, easy to fix and takes less time to learn are advantages of **high-level** programming languages.

Takes less time to translate, closer to machine code and makes faster programs are advantages of **low-level** programming languages.

18 of 19 – End

Well done. You have completed this session on revision (part 3).

In this session we have covered:

* What is meant by software
* What operating systems are
* The basics of computer programming
* The difference between high level and low level programming languages

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