code4sp
coding for social promotion

Co-funded by the
Erasmus+ Programme
of the European Union

3.1.:
Code4SP Training Material Package

WP3:
Code4SP Training Materials

Prepared by:

CD
DR

Project Information

Project Acronym: Code4SP

Project Title: Coding for Social Promotion

Project Reference: 621417-EPP-1-2020-1-PT-EPPKA3-IPI-SOC-IN

Project website: www.code4sp.eu

Authoring Partner: CodeDoor

Document Version: 2

Date of Preparation: 22/03/2022

Document History			
Date	Version	Author	Description
25/02/2022	1	CodeDoor	Draft
22/03/2022	2	CEPROF	Revision

Table of contents

Topic Information

Topic:

1. Computer Programming and its basic concepts

Prerequisites:

Basic computer literacy, basic software installed, and basic knowledge of working with files.

Workload:

5 hours.

Description:

In this topic, we give a brief introduction to computer programming and its basic concepts. We handle hardware and software concepts as well as how computers tore data. In the end we take a lot at how a program works and what type of programming languages are out there.

Learning outcomes:

- Recognize the concept of HyperText Markup Language (HTML) in the family of document description languages.
- Distinguish between the structure, content and styles of a page.
- Use HTML in the construction of pages for the web.

Material required:

- Computer or laptop
- Internet connection

Lesson Scenario:

The total time for this topic is 5 hours, and it will be up to the trainer/coach to decide how much time to dedicate to teaching each subtopic. In order to make the most of all the time available, we propose the use of the training materials produced by the project

(PPT presentations), which were designed with an effective use of time in mind. These presentations are composed of the following elements:

- Development of the subtopic and main ideas to retain;
- Proposed Activities/Exercises.

That said, if the trainer/coach follows the logical sequence of the PPTs, he/she will certainly be able to complete the session within the stipulated time limit. These presentations can also be made available to learners for individual study.

Subtopics:

1.1. Introduction to computers and programming

1.2. Hardware and Software Concept

1.3. How computers store data

1.4. How a program works

1.5. Programs and Program Languages

Additional resources:

- Khan Academy: useful resources computer topics in various languages
- Marc Andreessen: https://future.a16z.com/software-is-eating-the-world/

1.1. Introduction to computers and programming

Computers are machines that can be programmed to carry out a sequence of instructions. Programming is the process of designing those instructions. Programs are written in a particular language which provides a structure for the programmer and uses specific instructions to control the sequence of operations that the computer carries out. „Software is eating the world“ (Marc Andreessen, https://future.a16z.com/software-is-eating-the-world/) so make your learners aware of the fact that nowadays nearly everything is computerized and programmed (see also: Internet of Things).

What are some of the ways you use computers?

Let your learners do a brainstorming or create a mind map on how they are actually using computers. There are many ways you can use computers in your everyday life. Here are a few of the most common ways:

- To surf the internet
- To check your email
- To do research for a project
- To play games
- To watch movies or TV shows
- To listen to music
- To work on a document or spreadsheet
- To read the news

What other devices are computers?

Computers are not just the devices that sit on our desks or in our pockets. They are also the devices that run our cars, planes, and boats. They are in our televisions, refrigerators, and even our watches. In short, computers are everywhere.

What software have you used?

Let your learners do a brainstorming or create a mind map on what software they have already used. Try to make them aware that every Software was written by some developers. Even the Software that resides on the smartphones, cash-dispensers or even televisions they use everyday.

1.2. Hardware and Software Concept

Definition of hardware and software

Hardware refers to the physical components of a computer system, while software refers to the instructions and data that make the computer work.

The main components of a computer and its functions

A computer has four main components: the central processing unit (CPU), the memory, the input devices, and the output devices. The CPU is the part of the computer that performs the calculations and controls the other parts. The memory is where the computer stores the data it is working on. The input devices are the devices that the user uses to enter data into the computer, such as the keyboard and the mouse. The output devices are the devices that the computer uses to display the results of its calculations, such as the monitor and the printer.

1.3 How computers store data

Concept: the binary system

A binary system is a way of representing information using two symbols: 0 and 1. Binary is the simplest form of information representation and is used in computer systems to store and process information. Binary is a convenient way to represent information because it is very simple and can be processed by computer systems very easily.

Storing numbers, characters

ASCII

Computer systems store text and numbers in a variety of ways, each with its own benefits and drawbacks. The most common way to store text is as ASCII (American

Standard Code for Information Interchange) characters. In ASCII, each character is represented by a number, from 0 to 127. This number is called the character's ASCII code. When a computer stores text as ASCII characters, it simply stores the ASCII codes for each character in the text.

Unicode

Another way to store text is as Unicode characters. Unicode is an international standard that defines a unique number for every character in every language. When a computer stores text as Unicode characters, it stores the Unicode code for each character in the text.

UTF-8

UTF-8 is a character encoding that can store text and numbers in a single Unicode character. This encoding is widely used on the internet because it can encode all characters in a variety of languages. UTF-8 is a variable-length encoding, which means that it can encode characters of different sizes. The smallest encoding is 1 byte, and the largest encoding is 4 bytes. This encoding is backwards compatible with ASCII, which means that ASCII text will be encoded in UTF-8 using 1 byte.

Numbers

The most common way to store numbers is as binary integers. In binary, each number is represented by a string of 0s and 1s. For example, the number 12 can be represented as: 01001000 The number 12 can also be represented in hexadecimal, which is a base 16 numbering system. In hexadecimal, each number is represented by a string of hexadecimal digits. For example, the number 12 can be represented as: C When a computer stores a number in binary or hexadecimal, it stores the number's integer value.

Other types of data

Computers are often referred to as digital devices. The term digital can be used to describe anything that uses binary numbers. Binary data is data that is stored in binary, and a digital device is any device that works with binary data. In this section we have discussed how numbers and characters are stored in binary, but computers also work

with many other types of digital data. For example, consider the pictures that you take with your digital camera. These images are composed of tiny dots of color known as pixels. (The term pixel stands for picture element.) Each pixel in an image is converted to a numeric code that represents the pixel's color. The numeric code is stored in memory as a binary number.

The music that you play on your CD player, iPod or MP3 player is also digital. A digital song is made up of small pieces of music called samples. Each sample is turned into a binary number, which can be stored in a computer's memory. The more samples that a song has, the more like the original music it will sound when it's played back. A CD quality song has more than 44,000 samples per second!

1.4. How a program works

There are many different types of computer programs, but they all have the same basic components: a user interface, a processor, and memory. The user interface allows the user to input information and instructions into the program, the processor carries out the instructions, and the memory stores the program and the data it processes. Most computer programs are written in a high-level programming language, which is a language that is designed to be easy for humans to read and write. However, the processor can only understand machine code, which is a series of ones and zeroes. So, before a program can be run, it must be converted into machine code. This is done by a program called a compiler. The compiler reads the program and converts it into machine code. It then stores the machine code in a file called an executable. When the user runs the program, the executable is loaded into memory and the processor carries out the instructions.

The fetch-decode-execute cycle

The fetch-decode-execute cycle is the basic process that a computer uses to carry out instructions. The cycle begins when the computer fetches an instruction from memory. It then decodes the instruction to determine what it is supposed to do. Finally, it executes the instruction. The cycle then repeats, fetching the next instruction from memory.

From machine language to assembly language

As programming in machine language, which consist only of binary code is too complicated for a human being, assembly language was created. Assembly language is a low-level programming language for a computer, microprocessor, or other programmable device, in which the programmer uses assembly language instructions to control the operation of the device. Assembly language is specific to a certain microprocessor or family of microprocessors. It consists of a series of mnemonic codes, symbolic names for the operations that the microprocessor can perform, and

the operands (data) upon which these operations are to be performed. Assembly language is converted into machine code, a form of binary code that is specific to a particular type of computer and can be understood by the computer's processor. Machine code is the only form of code that the processor can directly execute.

Even assembly language programming is easier than machine language programming it was not handy enough to produce fast and easy to read source code. Therefore high-level programming languages (such as C# or python) where created.

Today there are many high-level programming languages. Some of the more common ones are Java, Python, and Ruby. High-level programming languages are easier to use than low-level programming languages. They allow you to focus on the task at hand, rather than on the details of the computer. This makes them ideal for creating applications and programs. High-level programming languages also tend to be more forgiving than low-level programming languages. If you make a mistake when writing code in a high-level language, the compiler will usually be able to correct it for you. This can save you a lot of time and frustration when coding.

Key Words, Operators, and Syntax: an overview

There are many high-level programming languages available today. Each has its own unique set of keywords, operators, and syntax. In order to be effective with a high-level programming language, it is important to be familiar with the specific keywords, operators, and syntax used by that language. Some of the most common keywords used in high-level programming languages include: *if, then, else, while, for, do, break, continue*. These keywords are used to control the flow of program execution. Operators are symbols that represent operations that can be performed on values. The most common operators include: + *(addition), - (subtraction), * (multiplication), / (division), and % (modulus)*. These operators can be used to calculate the results of expressions. The syntax of a programming language is the set of rules that govern how code must be written in order to be interpreted by the compiler or interpreter. The syntax of a high-

level programming language is typically more forgiving than the syntax of a lower-level language. This can make it easier for beginners to learn how to program.

Compilers and Interpreters

Computer compilers and interpreters are important tools for software developers. A compiler takes code written in one language and converts it into code that can be run on a different machine. An interpreter takes code written in one language and runs it as it is, without compiling it first. Compilers are typically used for languages that have a lot of structure, like C or Java. Interpreters are typically used for languages that are more flexible, like Python or Ruby. Compilers usually produce faster code than interpreters. However, interpreters are typically more portable, meaning they can run on more types of machines. Which tool to use depends on the situation. If speed is important, a compiler is a better choice. If portability is important, an interpreter is a better choice.

1.5. Programs and Program Languages

Programming languages are used to create programs, which are used to control the behavior of a machine, typically a computer. A programming language provides a structure for the programmer to give instructions to the machine, and a way to communicate those instructions to other programmers. There are many programming languages in use today. The most popular ones are C, Java, Python, and JavaScript.

Types of languages

There are dozens of programming languages in use today, but they can be broadly classified into five categories:

- **Low-level programming languages:** These programming languages are very close to the hardware and are used to program microprocessors and other low-level devices. They are not easy to learn and are not popular for general-purpose programming. Examples: Assembly language, C programming language, and Low-level assembly language.
- **High-level programming languages:** These programming languages are designed to be easy to learn and use. They are popular for general-purpose programming. Examples: Java, C++, and Python.
- **Scripting languages:** Scripting languages are designed to be easy to use and are popular for scripting purposes. Examples: Python, Ruby, and JavaScript.
- **Domain-specific languages:** Domain-specific languages are designed for a specific task or industry. They are not easy to learn and are not popular for general-purpose programming. Examples: MATLAB, SQL, and FORTRAN.
- **Object-oriented programming languages:** These programming languages are based on the object-oriented programming paradigm. Examples: Java, C++, and Python.

From a high-level program to an executable file

When a computer program is written in a high-level language, it is first translated into a lower-level language, which is more easily understood by machines. The lower-level language is then compiled into an executable file, which can be run on a computer.

IDEs (Integrated Development Environments)

An Integrated Development Environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE typically consists of a source code editor, build automation tools, and a debugger. The source code editor allows the programmer to write code, while the build automation tools automate the process of compiling that code into a form the computer can run. The debugger allows the programmer to step through the code, examining the state of the program at each point in its execution. IDEs are often used in conjunction with a version control system, which allows different programmers working on the same project to share and merge their changes seamlessly.

The common elements in programming languages

Computer programming languages share a number of common elements, despite their differences. All programming languages have a way of representing instructions to the computer in a form that the computer can understand. This is usually called code, or source code. Programmers use code to create software programs and applications. All programming languages also have a way of organizing instructions so that they can be reused, modified, or shared with other programmers. This is usually called a library or module. Libraries and modules allow programmers to create complex programs by building on the work of other programmers. Finally, all programming languages have a way of conveying information to the user about what the program is doing and how it is performing. This is usually called output or debug information. Output and debug information helps programmers understand and fix problems with their programs.

Procedural and object-oriented programming

There are two main types of programming: procedural and object-oriented. Procedural programming involves a step-by-step process, while object-oriented programming involves creating objects that interact with one another. Procedural programming is often seen as simpler than object-oriented programming. It is easy to learn the steps required to complete a task, and it is easy to change the order of those steps without affecting the outcome. However, procedural programming can be less efficient because it can be difficult to reuse code that has been written for a specific task. Object-oriented programming is more complex than procedural programming, but it allows for more flexibility and reuse of code. Objects can be created for specific tasks and then reused as needed. In addition, object-oriented code is often easier to read and understand than procedural code. However, object-oriented programming can be more difficult to learn and may be less efficient than procedural programming.