WALKING DEBATE: PROGRAMMING LANGUAGES

DIGITAL CONTENT CREATION > 3.4 PROGRAMMING

TARGET GROUP	AGE GROUP	PROFICIENCY LEVEL	FORMAT	COPYRIGHT	LANGUAGE
All	Adults, Elderly citizens, Teenagers	Level 3	Activity sheet	Creative Commons (BY-SA)	English, French

A moving debate to understand and above all to de-dramatize programming and its languages. Programming does not have to be scary and it can be accessible to anyone !

General Objective	Knowledge acquisition
Preparation time for facilitator	less than 1 hour
Competence area	3 - Digital content creation
Time needed to complete activity (for learner)	0 - 1 hour
Name of author	Aurelie Corvot
Resource originally created in	French

WORKSHOP DIRECTIONS

1 Introduction

What is a walking debate? This is a debate in which participants will play a physical role. They will move around depending on their response to a given statement (agreement/neutral/disagreement). For example, to respond 'yes', they may take one step forward. For a neutral response, they could stay where they are and for a disagreement, a step backwards. Simple right? Prior to the statement being made and while participants are responding through movement, they should not speak in order to not influence each other. After the decisions are made, invite the participants to express their view on the question asked by explaining why they moved the way they did. Easy!

Facilitation tips: The statements listed below are examples tested by our moderators. We therefore know that they are good examples for debate encouragement. This shouldn't stop you from coming up with your own ideas! However, we would advise you to choose statements that don't have a clear or straightforward response (in order to encourage the debate) and particularly to test them beforehand (with colleagues, friends, family) to ensure that they work well for this type of activity. You can use this debate as a way to introduce activities based on programming or adapt it to turn it into a quiz to be used after a relevant activity to test participants' knowledge.

2 Statements – Debate on programming language

1. Choosing a language is a bit like choosing a weapon on the battlefield. We can be into massive spiky maces or thin swords. Neither is better than the other – the importance is choosing based on the initial requirements and using them artfully and appropriately.

2. You should choose the right language from the start. There is a debate around the most appropriate language for starting to learn how to program and the language most suitable for children. If we consider languages used in real life, it is always better to start with a language whose syntax is easy to learn. Python and Javascript are two good examples and can be used easily on all devices and all browsers. But it is impossible to pick *the* right language, since each child is different and, whereas some will easily pick up a language, others will find that same one difficult to learn. Every child should choose their favourite language – or not study programming at all, which is not so bad either. Also, it is hard to predict what language(s) will be the most popular when the new generation grows up.

3. It is required to learn several programming languages. The learning of a programming language should be done in relation to one's needs. Everyone will learn in different ways. It is therefore important to ask ourselves the right questions. It is clear that if you program just for fun you will certainly not want to go through the trouble of learning several languages. Ask yourself: what is your goal in programming? What is your project? What is your ambition? Are you a hobby programmer or an expert?

4. All IT languages are programming languages. There are many IT languages. Many are programming languages, but not all. This confusion is common with languages used for creating websites. For example, HTML is under no circumstances a programming language. It simply allows users to statically edit documents displayed in web browsers.

5. A good developer should know how to program in all languages. To be a good developer doesn't only mean being able to program well. You should also be able to effectively anticipate, listen and adapt. Listening obviously to best satisfy clients' demands. Adaptability for efficiency. And anticipation to see future needs coming, that is to say to change the way you work in order to make it easier for the people and projects that will follow.

6. Programming is the domain of particularly talented, mathematically minded people. This is a preconception from people who have never tried programming or who do not understand it. It is obviously false. Especially since if you use maths in programming it will usually be basic maths known by everyone. Clearly though, you would need to have good general foundations in IT to be able to learn and practice programming.

7. You need expensive equipment to begin programming. Contrary to what we might think, you don't need to invest in top quality equipment to start coding. Obviously having more advanced tech will give you more power and fluidity in your work, but a basic computer will be enough to start. There are exceptions to this rule however. If you want to make applications for iOS (iPhone or iPad), you will need a Mac and these are expensive machines. You may also need extra material in some cases, for example if you want to program using Arduino where you need the Arduino chip to test what you've done. The cost for these is not enormous (around 20-30 euros) but these purchases will have to be taken into account.

9. Programmers are solitary nerds and introverts. A programmer is a **normal person** like everyone else. They leave their houses, have fun, share, meet others, explore the world around them and most of all they think about the conceptual sides of things they encounter. Do you live your life like any other human on earth? I have good news for you: you have the right disposition to be a programmer.

10. Programming is a discipline requiring masculine skills. The pioneer of programming was a woman called Ada Lovelace. However, IT in general is for some reason seen as a male domain. Many women have a negative image of IT – much like most people who have no familiarity with it – mostly as a result of a preconception that programmers are solitary and introverted. IT careers are now starting to be pursued by more women and more and more women are now starting to learn to program.

3 Going further: some information on the history of programming languages

Year: 1842-1843 Language: First programming language Ada Lovelace is known for creating the first computer program, while working on an ancestor of the modern computer: Charles Babbage's proposed 'mechanical general-purpose computer'.

Year: 1957 Language : Fortran FORmula TRANslation is the oldest language still in use. Created by John Backus, it was developed to make high level mathematical, scientific and statistical calculations. It is still used in aeronautical and automobile industries as well as by governments and research institutions.

Year: 1959 Language: Cobol Common Business Oriented Language is behind the majority of commercial transactional systems processing credit cards, ATMs, and telephonic, hospital, governmental, automobile, and road signage payment infrastructures. Cobol's development team, directed by Dr. Grace Murray Hopper, wanted to create a single convenient language for all commercial transactions.

Year: 1964 Language: Basic Developed by students at the university of Dartmouth, this used symbolic and simple all-purpose instructions and was designed to a simplified language for people without a lot of technical or mathematical knowledge. A modified version, written by Bill Gates and Paul Allen, became Microsoft's first product. It was sold to MITS who used it for their Altair computer.

Year: 1969 Language: C '**C**' was developed between 1969 and 1973 by Dennis Ritchie at Bell Labs for use with the operating system Unix. It was named 'C' because its characteristics were taken from a prior language 'B'. C became powerful enough that most of the Unix kernel was rewritten in C. This was one of the first operating system kernels to be implemented in a language other than the one in which it was implemented (B).

Year: 1970 Language: Pascal This was named after Blaise Pascal, credited for inventing the first mechanical calculator in 1641. Niklaus Wirth created Pascal as a teaching tool designed to encourage good programming practices and it has evolved for use for commercial purposes.

Year: 1983 Langage: C++ From Bell Labs, Bjarne Stroustrup modified C to C++ and created what most consider to be the most popular programming language of all time. It has been in the top ten most used languages since 1986.

Year: 1987 Language: Perl Larry Wall, a Unix programmer, created Perl after attempting to extract data for a report and finding that Unix was incapable of doing what he wanted. The **Practical E xtraction Report Language** was described by its inventor as a language to make report processing easier.

Year: 1991 Language: Python Monty Python was the inspiration for the name. Guido Van Rossum developed Python to solve problems with the language ABC. He continued to serve as the language's lead developer until 2018.

Year: 1993 Language: Ruby Yukihiro 'Matz' Matsumoto named Ruby after the birthstone of July. He developed it through an amalgamation of his favourite languages: Perl, Smalltalk, Eiffel, Ada and Lisp.

Year: 1995 Language: PHP Rasmus Lerdorf developed PHP to replace Perl scripts used to maintain his personal homepage. Today, PHP has become an integral part of the internet's architecture being used on over 20 million websites.

Year: 1995 Language: Java A team of developers at Sun Microsystems directed by James Gosling created Java to manage decoding systems for interactive television. It was designed to have as few implementation dependencies as possible. It functions today on more than 1.1 billion PCs across the world and many websites cannot function with it.

Year: 1995 Language: JavaScript Java and JavaScript are unrelated and work very differently. Javascript was originally developed by Brendan Eich of Netscape under the name Mocha. It uses a syntax influenced by that of C. Although meant to work on browsers, it is now used in server-side website deployments, usually via Node.js. Ajax for example runs on JavaScript.

Year: 1995 Language: Ruby on Rails Ruby on Rails, or simply Rails, was extracted by David Heinemeier Hansson from his work on the project management tool Basecamp. Hansson released Rails as open source in July 2004 but did not share the full commit rights until February 2005. Version 6.0 was released in 2019.