INVESTIGATING THE MOST UNDERSTANDABLE PROGRAMMING LANGUAGE FOR BEGINNERS

OLIVER KERR (Computing)

Abstract
Learning to program can be an overwhelming experience for many people; it can be difficult just knowing where to begin. A fundamental decision for beginners is which of the many programming languages they should choose to learn first. This research exercise explores the learning potential for first time users of different programming languages and offers results that show first time programmers prefer languages with simple syntax.

Keywords
Programming, Computer, Science, C, Python, PHP, Java, JavaScript, Education, Novice Programmers

1 Introduction
In today’s modern society computers are becoming more ubiquitous. They are used for a wide variety of tasks from creating documents to instant messaging on smart phones and, more recently, tracking exercise statistics and receiving notifications on the go with wearable smart devices. These physical devices (the hardware) would simply be empty shells without the applications (software) giving them purpose.

Software is created by programming. There are numerous programming languages to choose from, some being more specialized for specific tasks and certain hardware. An example is Matlab, which is often used within statistical applications, whereas Prologis is a language mainly used for artificial intelligence. There are also many general-purpose languages, such as C and Java, which can be used in almost any type of application.

The purpose of this investigation is to try to understand how novices learn to program. Programming is a difficult subject to understand, as it requires the comprehension of abstract concepts that do not normally arise in other subjects (Lahtinen et. al. 2005). Programming is also a very diverse subject, which means that it can be demanding to learn; Winslow (1996) suspects that it can take around ten years before a novice programmer can be considered an expert. This investigation focuses on finding the most easily understood language for beginners to learn first, which will allow them to begin to understand the
fundamental concepts of programming, without being confused by the complex syntax of some of the more complex languages.

This will lead to a recommendation as to which language should be learnt first by anyone wanting to learn to program, regardless of age, which could then be applied in schools and colleges that are beginning to teach programming as a subject, thus, covering a wide range of ages is especially important.

2 Design

The hypothesis for this experiment is that first time programmers will prefer languages with simple syntax, being defined as those that are most like Standard English. To investigate this fully an Internet based survey was designed, in which participants are shown a function in five different languages – Python, C (C was chosen to represent C++, C# and Objective C due to their similar syntaxes), Java, JavaScript and PHP. These five languages have been chosen from the hundreds that are available as they are the top five most popular programming languages of 2014, according to the Institute of Electrical and Electronics Engineers (IEEE) (Cass 2014). A secondary hypothesis is that C and Java will have similar scores due to Java being based on the C language and, therefore, they have similar syntaxes.

An example of a simple function that is used to output text on a screen, and was included in the investigation, is shown below.

Option A (Python) – `print “Hello world”`
Option B (C) – `printf (“Hello world”);`
Option C (PHP) – `Echo “Hello world”;`
Option D (JavaScript) – `document.write (‘Hello world’);`
Option E (Java) – `System.out.println (“Hello world”)`

Participants are asked to choose which of the languages appears to be the simplest and most straightforward to understand. They then answer a multiple-choice question about the function, in order to evaluate their understanding of the code. At the end of the survey participants are asked to rank each of the languages on how easy they are to understand on a Likert scale: one being the easiest and five being the hardest.
The survey does not ask the participants to write any code for themselves. This is because the investigation is aimed at individuals who have no prior experience of programming and with no prior explanation of any programming constructs. When answering questions participants are not told if they answered correctly, as informing them may influence their responses to later questions, leading participants to perhaps rely on trial and error to answer questions, rather than trying to understand the code.

The Leeds Group from the ITiCSE conference in 2004, following their investigation into how students perform when completing programming related tasks, concluded that ‘the ability to read code is a precursor skill to the ability to write code’ (Lopez et. al. 2008). This conclusion justifies the purpose behind this investigation. In finding the programming language that is the simplest to understand, with little or no explanation for beginners, this will help the users to become better at both understanding, and later, in writing the code for themselves.

A pilot survey was carried out before beginning the main investigation. By testing out the survey on a small number of participants, it makes it possible to highlight any issues participants may have with the survey, i.e. badly worded questions. Correcting these issues will allow participants to focus more on trying to understand the code instead of being confused by the survey, resulting in more accurate results.

### 2.1 Survey Design

The survey was targeted towards individuals who have no programming experience; there was no explanation of the concepts of programming within the survey (with the exception of defining the mathematical symbols used, e.g. greater than, less than, etc. Each question covered a different topic of programming and was split into two sections.

Section one presents participants with a function written in the five different languages, of which they select the one that appears simplest and most understandable to them.

The functions will vary in complexity as questions progress. Despite this being a relatively simple function there is a significant amount of variation between how it is written in each language. The
differences include the use of brackets and semi colons, as well as keywords that may either relate to the real world purpose of the function (such as Python’s ‘Print’ statement which is used to output text to the screen), or may be more abstract, which could influence participants’ understanding of each of the languages.

The second part of the questions tests the participants’ understanding of the code by presenting them with a multiple-choice question relating to the purpose or output of the code.

The topics covered in the survey include: variable declaration – assigning a variable a name and value, i.e. a = 20; simple output statements – displaying a line of text on the screen; mathematical calculations and simple and complex ‘IF’ statements – used to decide whether the program should follow one set of instructions or another. Loops, which are used to repeat a set of instructions until a condition has been met, were originally intended to be included in the survey; however, the pilot survey revealed that participants found these concepts very difficult to understand without detailed explanation so they were removed.

The results from the pilot survey have not been included in the overall findings of this investigation to ensure an accurate and reliable result. However, the trends within the pilot survey are the same as those found in the main investigation. Despite removing loops from the survey there are still enough topics covered to develop an understanding of how people who have never programmed before attempt to try and understand code.

Once a novice has grasped these simple topics they will be able to transfer them to other languages and programming techniques, such as Objected Oriented (OO) programming. OO programming is a commonly used approach which has an emphasis on objects – a combination of variables, functions and data structures, and classes – a set of instructions for a particular object. The OO paradigm allows for greater efficiency through code re-usability – when a program refers back to a specific function instead of rewriting it. However, first time programmers often have trouble forming mental representations of how a program works (Wiedenbeck and Ramlington 1999), meaning first time programmers will likely find it easier to grasp concepts used in procedural programming methods.
2.2 Participant Selection Methodology

The participants for the investigation were split into 3 age groups: 11-14, 15-18 and 19+, thus allowing for any trends between younger students (Key Stage 3), older students (Key Stage 4 & 5) and adults to be discovered.

The participants were selected using quota sampling. Quota sampling is based on selection by non-random means, making it useful for investigations that need to be completed in a short amount of time or when access to the whole population is difficult, but still allows for generalization of the results (Walliman 2010).

36 participants were involved in the investigation, with 12 people in each group and an equal male to female ratio. By using quota sampling to select participants from people the researcher knows, generalized conclusions can be drawn regarding the most understandable language for individuals who have never programmed before, as well as highlighting any trends that may occur due to the age of the participants. Quota sampling allowed for as fair as possible sampling process; there is a slight possibility for bias as specific individuals are being targeted and may not fully represent the views of the entire population due to the limited sample size and time available to conduct the research.

Participants completed the survey in controlled conditions and were observed to ensure there was no possibility for cheating which could create invalidations. Before each participant began the survey they were shown a brief of the investigation and asked to confirm they had no prior programming experience, as well as consenting for their results to be used in the investigation anonymously. No participants who were involved in the pilot survey were included in the final investigation.

3 Results

Table 1 shows the participant’s ranking of each programming language, using a Likert Scale, with 1 being the easiest and 5 being the hardest to understand. To make analysing results easier each language was given a score based on their ranking. Scores were calculated by assigning each rank a set number of points – 5 points for rank 1 (easiest to understand), to 1 point for rank 5 (hardest to understand). The percentage of participants was then multiplied by the number of points for each rank and a mean was then calculated from these to be used as the score for that language. This resulted in the higher the score the easier participants found it to understand.
<table>
<thead>
<tr>
<th>Option</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>69.4%</td>
<td>19.4%</td>
<td>2.8%</td>
<td>0.0%</td>
<td>8.3%</td>
<td>4.4</td>
</tr>
<tr>
<td>Java</td>
<td>11.1%</td>
<td>27.8%</td>
<td>41.7%</td>
<td>19.4%</td>
<td>0.0%</td>
<td>3.3</td>
</tr>
<tr>
<td>C</td>
<td>16.7%</td>
<td>30.6%</td>
<td>25.0%</td>
<td>5.6%</td>
<td>22.2%</td>
<td>3.1</td>
</tr>
<tr>
<td>JavaScript</td>
<td>2.8%</td>
<td>5.6%</td>
<td>25.0%</td>
<td>58.3%</td>
<td>8.3%</td>
<td>2.4</td>
</tr>
<tr>
<td>PHP</td>
<td>0.00%</td>
<td>16.7%</td>
<td>5.6%</td>
<td>16.7%</td>
<td>61.1%</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Table 1. Participant rankings of how easy it is to understand each language; 1 = easiest, 5 = hardest. Percentages rounded to 1 d.p.

It is clear to see that Python was chosen as the most understandable language, with almost 70% of participants ranking it first, giving it a mean score of 4.4. Java and C were scored 3.3 and 3.1 respectively, it was expected that the two languages would have similar results, as Java takes much of its syntax from the C language. PHP scored the lowest of the 5, only achieving 1.8.

The data collected from the survey identified that Python was the most understood language in all of the questions, followed by either Java or C, based upon their mean score. The majority of participants also answered the multiple choice questions correctly, showing that even with no prior programming experience it is still possible for individuals to understand basic programming concepts.

4 Analysis and Discussion

The data collected during the experiment clearly describes a trend amongst all participants that Python is the most preferential of the languages that were included in the investigation. Almost 70% of participants ranked Python as their top choice; it is suspected that this is due to its similar syntax to Standard English. PHP was the lowest scoring language, over 60% of participants ranked it as the hardest to understand.

When comparing a piece of code in both Python and PHP side by side, it is clear why a first time programmer may be put off using PHP and be drawn to Python. For example, variable declarations in PHP use dollar signs – $a = 20; whilst in Python the same declaration would simply be - $a = 20. These
may only be minor differences but could have a profound impact on an individual’s understanding of a language. Lahtinen et. al. (2005) stated that the biggest problem with novice programmers is not their ability to understand the basic concepts of programming, but rather learning to apply them appropriately. This may link to the results from this investigation, as participants were able to understand concepts they already held a real world meaning for. This was evident within the results amongst participants when they understood Python output statement – *print*, rather than the PHP *echo* statement as they were able to deduce the meaning of the function.

After participants completed the survey an informal discussion took place to allow them to voice their opinions. A number of participants noted the similarities between algebra and programming, especially in Python, due to its simplistic syntax and transformation rules. This finding is supported by Rogalski and Samurçay (1990) who stated that students who came from mathematical backgrounds found abstract concepts easier to understand and therefore required less teaching.

It was also stated that when a student begins to learn programming they commonly refer to everyday objects to allow them to visualise what the program is doing (Rogalski & Samurcay 1990). These real world objects can come from mathematics, physics, or even basic logic. This supports the theory that languages such as Python that have a relatively simple syntax and use self-explanatory function names, which are relatable to real world examples, are more straightforward for first time programmers to understand.

Duncan (2002) maintains that it is possible to divide programming students into three categories. The first is for students who are not able to grasp basic programming concepts. The second category includes students who have the ability to understand the abstract concepts of programming but require help from teaching staff, whilst the third category includes students who are able to understand programming concepts with little or no input from teachers. These categories highlight the divide in novice programmers and are often attributed to a person’s ability to apply what they have learnt in other subjects such as mathematics as discussed above.

Programming is a complex process (Çakiroğlu 2013) that requires the use of a variety of different cognitive abilities in order to develop a mental representation of what the program is doing
(Wiedenbeck and Ramlington 1999). Due to this, it is important for first time programmers to spend more time learning to understand the concepts of programming, rather than focusing on the syntax of a particular language. Once the basic concepts are understood they can be applied to other languages with more complex syntaxes, as novice programmers will have already developed an understanding of the concepts of the language, and can therefore, focus on learning the more complex syntax.

5 Further Research

One of the limiting factors of this investigation is the small sample size. This is due to the time constraints for collecting data. For future work it would be preferable to continue the investigation with a larger sample size to see if the results are correlated. This could be achieved by sending the survey to schools across the country.

There is also scope for further research into different teaching methods to help students who are struggling to understand programming concepts. One possible long term research topic could investigate the impacts of teaching primary school students visual programming, such as Scratch, and evaluate how it impacts on their learning of normal programming languages, in addition to related topics such as algebra and logical reasoning. This proposal would be supported by the government’s initiative to make programming a part of the curriculum for all schools in the UK.

6 Conclusion

Drawing from the results collected by this investigation, the simpler syntax of Python makes it the easiest of the languages for first time programmers to understand, therefore, allowing the hypothesis to be accepted. There appears to be no significant variation in preferences due to the age of participants, further supporting the recommendation that all first time programmers would benefit from learning a language such as Python, which is perhaps why it is taught in schools. By using Python, basic programming concepts can begin to be understood, without being confused by complex syntax or meaningless keywords. The knowledge gained can then be applied to other more complex languages such as C.

References


