Java, Python and Javascript, a comparison

Rasmus Horntvedt and Tobias Åkesson
Author
Rasmus Horntvedt, Tobias Åkesson

Title
Java, Python and Javascript, a comparison

Supervisor
Kamilla Klonowska

Examiner
Daniel Einarson

Abstract
With the amount of programming languages currently available there is a high risk of confusion and doubtfulness in aspiring programmers of which to choose. It may be motivating for a beginner to choose “the perfect language” when starting, to avoid learning multiple languages. This thesis compares three popular languages on three separate aspects, their syntax, usefulness in different areas, and performance in terms of speed. Syntax wise the results varied with some aspects being very similar across all three languages to completely different in others. In terms of usefulness in specific areas the languages flexibility allowed them to develop applications in most fields, while being dominant in different areas. The speed comparison resulted in python being the slowest across all tests, with Java and Javascript (running inside Nodejs) competing for first place.

Keywords
Python, Java, Javascript, Comparison, Benchmark, Performance, Syntax, Programming language.
# Content

1. Introduction 4  
   1.1 Purpose 4  
   1.2 Research questions 5  
   1.3 Limitations 5  
   1.4 Structure RQ1 6  
   1.5 Structure RQ2 6  
   1.6 Background 7  

2. Method 8  
   2.1 Ethical acknowledgements 8  
   2.2 Research question 1 8  
   2.3 Research question 2 8  
   2.4 Research question 3 9  
      2.4.1 Testing system and setup 10  

3. Literature review 11  
   3.1 Python, Java and Javascript - Overview 11  
      3.1.1 Python 11  
      3.1.2 Java 12  
      3.1.3 Javascript 12  
   3.2 Related work 13  

4. Result 15  
   4.1 The differences between the languages (RQ1) 15  
      4.1.1 Variables 15  
      4.1.2 Comments 16  
      4.1.3 Separators 16  
      4.1.4 Indentations or curly brackets 16  
      4.1.5 Selection statements 17  
      4.1.6 Loops 17  
      4.1.7 Data structures 18  
         4.1.7.1 Array 18  
         4.1.7.2 Array Lists 19  
         4.1.7.3 List 19  
         4.1.7.4 Sets 19  
         4.1.7.5 Tuples 20  
         4.1.7.6 Stacks and queues 20  
         4.1.7.7 Dictionaries 21  
      4.1.8 Classes 21  
      4.1.9 Functions and Methods 22  
      4.1.10 Error handling 22  
      4.1.11 File management 23  
      4.1.12 Operators 24  
   4.2 What can the languages be used for? (RQ2) 24  
      4.2.1 Python usage areas 24  
         4.2.1.1 Python machine learning 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.1.2 Python automation</td>
<td>25</td>
</tr>
<tr>
<td>4.2.1.3 Python web development</td>
<td>26</td>
</tr>
<tr>
<td>4.2.2 Java usage areas</td>
<td>26</td>
</tr>
<tr>
<td>4.2.2.1 Java Android development</td>
<td>27</td>
</tr>
<tr>
<td>4.2.2.2 Java full stack development</td>
<td>27</td>
</tr>
<tr>
<td>4.2.3 Javascript usage areas</td>
<td>28</td>
</tr>
<tr>
<td>4.2.3.1 Javascript full stack development</td>
<td>28</td>
</tr>
<tr>
<td>4.2.3.2 Javascript mobile development</td>
<td>29</td>
</tr>
<tr>
<td>4.2.3.3 Javascript Desktop application development</td>
<td>30</td>
</tr>
<tr>
<td>4.2.4 Comparison on different usage areas between the languages</td>
<td>31</td>
</tr>
<tr>
<td>4.3 Which language is solving tasks the fastest? (RQ3)</td>
<td>34</td>
</tr>
<tr>
<td>4.3.1 Quicksort</td>
<td>34</td>
</tr>
<tr>
<td>4.3.2 Fizzbuzz game</td>
<td>35</td>
</tr>
<tr>
<td>4.3.3 Reading from file</td>
<td>36</td>
</tr>
<tr>
<td>5. Discussion</td>
<td>37</td>
</tr>
<tr>
<td>5.1 Discussion of research question 1</td>
<td>37</td>
</tr>
<tr>
<td>5.2 Discussion of research question 2</td>
<td>38</td>
</tr>
<tr>
<td>5.3 Discussion of research question 3</td>
<td>39</td>
</tr>
<tr>
<td>6. Future work</td>
<td>40</td>
</tr>
<tr>
<td>7. Conclusion</td>
<td>40</td>
</tr>
<tr>
<td>8. References</td>
<td>41</td>
</tr>
<tr>
<td>9. Appendix</td>
<td>54</td>
</tr>
<tr>
<td>9.1 A Syntax</td>
<td>54</td>
</tr>
<tr>
<td>9.2 B Benchmarking algorithms</td>
<td>61</td>
</tr>
</tbody>
</table>
1. Introduction

As a new programmer in the ever growing computer world it is difficult to pinpoint exactly what one needs out of a programming language. With a long catalog of languages it is difficult to make a decision of what programming language is the best, since there is no definitive answer to what “the best” is. The best could be in terms of third party support, ease of understanding, speed or functions etc. Therefore it is impossible to tell what the best is. However, it is possible to explain why one language would be preferred over the other in certain situations. It is still important to note that generalization and misinformation can give the impression of big differences in terms of any of the previously mentioned factors and can therefore skew the decision of what programming language to use. There are several popular programming languages, but the three chosen for this report is as follows: Java, Python and Javascript. Java is one of the most prominent and successful languages in the world [1]. Python has been around since the late 1980’s [2] but has in the last few years been steadily rising in popularity and is now competing with some of the most popular programming languages such as C and C++ [1]. Javascript, although the least popular language out of the three chosen [1], is still relevant thanks to the support it gets from different frameworks that it can implement, and developers who continue to add features to it like for example with Nodejs [3]. Tiobe was used as a way to measure the popularity of the languages. It should be noted that Tiobe uses search index and so it is not a clear correlation between popularity and the statistic being presented on the site [1]. The three languages are fundamentally different but allows us in a nuanced way to compare different aspects of the different languages with each other, and how much weight one should put on a specific aspect of a language.

1.1 Purpose

The purpose of this report is to analyze and compare these popular programming languages in three different ways. It is not meant to declare that one language is superior to the rest, but rather analyze desirable traits and strengths in all of them. The reasons for choosing this topic was to understand how different programming languages work and if it is easy to convert knowledge between languages. Considering we have mainly focused on Java during our studies, it was in our interest to develop an understanding of different languages and make a comparison between them. The thesis is primarily targeted towards novice coders, and programmers that has established some knowledge within at least one programming language and is interested in how other programming languages work. It requires some knowledge to understand every component of programming language since they are not explicitly explained in the thesis. It can also be hard to understand the
terminology used if the reader is a complete beginner. The report is clearly divided into three separate parts, each focusing on different aspects of the programming languages. The first part of the report centers around the differences between the chosen languages in terms of how easy they are to understand and use for development. This part also involves mentioning similarities. The second part of the report is about finding each language best use case and when it is preferred to be used. This can be interesting to know since some of the chosen languages are considered multi-purpose languages. The last part of the report is about comparing the speed and run time of each language and how they handle different computing tasks.

1.2 Research questions

Research question 1: What is the differences between Java, Python and Javascript in terms of syntax and ease of understanding?
Research question 2: In what situations is it preferable to use one of the languages over the others?
Research question 3: Which of the three languages are solving tasks the fastest?

1.3 Limitations

In order to not exceed the time limit, the authors needed to set some limitations. It would be unreasonable to try and compare all the languages and it was therefore necessary to narrow it down to only comparing three languages. It is important to note that there are languages which are more popular than Java, Javascript and Python, such as C and C++ [1]. The reason as to why these languages were not chosen was because they involved lower level concepts, which might overcomplicate the beginning for new programmers. There was also not sufficient time to compare additional languages. Another limitation that was needed to be done, was when deciding what to analyze when answering the first research question. Trying to break down the complete syntax of all components in three languages would not be plausible in the time that was given and so the authors focus on the more prominent parts of the languages. The authors acknowledges that libraries is a big part of a programming language and can be used to extend the features of said language. The time restraint and sheer amount of libraries available would make it impossible to compare and add valuable knowledge to the field of computer science. When answering research question 2 the authors needed to set some limitations on where the programming language exceeds. The authors picked the areas in which the programming languages are popular to use and then tried to explain why they may be popular in that specific field. Another limitation that was needed to be done was to decide which tasks should be tested for research question 3 since it is not reasonable to test a large amount of tasks. Instead we decided to focus on some tasks that does very different things.
1.4 Structure RQ1

The first research question needed extra limitations since there are many components to talk about in each programming language. We decided to focus on the parts that we found the most interesting and what we consider to be the most fundamental parts of a programming language. However, we also wanted to add some analysis of more complex features of programming languages and that is why we chose to add error handling and file management as part of the thesis. The programming language components that will be discussed in the thesis can be seen in figure 1.

![Figure 1. The covered components in research question 1](image)

1.5 Structure RQ2

In RQ2 we discuss which language to pick for certain situations. With “situations” we felt that it was necessary to explain what we mean and what we are going to look for. This research question was focused on specific applications. Which language should one pick for a certain application, for example if someone wants to develop a web application, which programming language should that person pick to get the necessary features. The aspects that we want to focus on is web applications, machine learning, mobile applications, scripting (small automation programs) and graphical desktop applications.
1.6 Background

Every language has its own books or articles made to help programmers understand how the language works and how to write programs for it. It is important to note that although there are articles which compare languages, there are far fewer academic publications that compares languages. Nevertheless, there has been studies done in the past which focuses on if it is advantageous to switch from java to python for the first year students at the university. The motivation was based on their idea that python in general is easier to get into compared to java, even though they are both considered high level languages [4].

The different aspects to compare in programming languages reach far and wide, and it is not possible to look at all the different sides in one thesis. Previous work includes “Analyzing Programming Languages’ Energy Consumption: An Empirical Study” which focuses on energy efficiency amongst 14 languages, the three languages that were chosen in this report are present in that article [5].

In terms of comparing languages, there are some previous work done from Stephen J. Humer & Elvis C. Foster which did a comparison between C++, Java and Python at the Keene state college. Their report focused mostly on the theoretical aspects of the languages and the syntax [2]. The aim of the research that we are conducting is to expand on that research by adding a benchmark test on the languages. We are also comparing different languages than the previously mentioned research paper, by replacing C++ with Javascript the report adds additional theoretical information about the languages.
2. Method

2.1 Ethical acknowledgements

There were no surveys or interviews conducted during the writing of this paper, and so the authors did not have to apply any specific rule to make sure that the participants privacy are hidden. When citing sources, it is of big importance that the authors and their work is credited, this is achieved by using the Vancouver reference system where all the sources can be found in the “References” chapter. The authors is also not interested in any kind of misleading of information and should therefore apply a scientific view and have sources to motivate all statements.

2.2 Research question 1

The research question “What’s the difference between Java, Python and Javascript in terms of syntax and ease of understanding?” is answered by reading each languages official documentation and related books. However, Javascript does not have its own official documentation due to the differences in implementation in all the browser engines. This thesis will therefore use Mozilla's documentation on Javascript [6]. Although, the documentation can only be used to answer how they differ in syntax. The authors performed a literature review in order to see what conclusions previous researchers had drawn when deciding what makes a language easy to understand. When the authors of this thesis have read about each parts of the programming languages, every bit of code was tested before a conclusion could be made. In order to test the code, the authors needed to use some software. All Java code has been written in the IntelliJ IDEA IDE using Java 8. The Python and Javascript code are written in the Visual Studio Code text editor. The Python version that was used is 3.7.2. When testing Javascript code it was running inside Nodejs.

2.3 Research question 2

Research question 2 “In what situations is it preferable to use one of the languages over the others?” was answered by reading scientific articles, surveys, documentation and books about the subject. Another focus point was to research and find out in what type of applications these languages are used for and what frameworks it supports. The research was done by first looking at each language individually and then take their most common usage area and compare it to the same usage area in the other languages.
2.4 Research question 3

The third and last research question “Which of the three languages are solving tasks the fastest?” was answered by running different algorithms in the three chosen languages. Another test that was done, was that the authors performed some reading from files in each language to determine how fast they were at doing this task. The first algorithm used for this test was a simple arithmetic Fizzbuzz algorithm that checks if a given iterated number is divisible with 3 or 5 and prints fizz and/or buzz. The second one includes a recursive quicksort algorithm in order to sort a list of numbers. The motivation for doing these tests in particular was that these types of operations are very common. The code used in quicksort and fizzbuzz tests were from the rosetta code website[7, 8], and the reasoning for using code from this website was because of the fact that the authors of this thesis wanted as close to a fair comparison possible, with a similar structure and also code that has been reviewed and updated by several users over time. However, reading from file task was included in the same code as writing to file on rosetta code, and some of it did not work because the languages have changed so this task is not taken from rosetta code. The code that was from rosetta code needed some slight modifications, mainly adding timers so that a more precise time could be measured. The algorithms were run several times, and they were also executed on the same computer at the same day, with no other applications running in the background in order to eliminate as many outside factors that may cause the result to vary.

The quicksort algorithm testing process started with the authors generating three files with randomly generated numbers and then used these files for all languages. The files had an increasing amount of data entries(10000, 100000, 1000000) and were tested 150 times each (during the same execution in a loop) in order to get a fair average time. The authors wanted to avoid the risk of having imbalanced random datasets where one would be more ordered than the rest and therefore opted to use the same dataset across all languages. The programs would then read the file into a list and then measure the time when the algorithm started sorting to when it had finished. The different methods for getting the time in each programming language was giving different time precision so all timestamps were rounded to the nearest millisecond. The results were then used to calculate the average time. The average time was used to generate a graph displaying the time differences between the languages and also the increase in time for the larger datasets.

Fizzbuzz testing was simpler in terms of testing since there was no need to generate datasets and no file management needed to get the set. Instead the for loop used in the algorithm would iterate increasing amounts of times (1000000, 10000000, 100000000) and need to print more and more to the terminal. The testing was a bit different from testing quicksort, instead of running the fizzbuzz
sequence in a loop during the same execution, it was changed to being executed after each other. This was done five times and after that, the average time was calculated and added to a graph. The third and last task that was measured, was reading from file that was containing about 30MB of text and assigning that result to a variable. This testing was done like quicksort where the reading of the file was done 100 times (in the same execution) and the timestamp was saved and rounded to calculate the average later.

2.4.1 Testing system and setup

All the tests were done on a desktop computer with the following hardware:
CPU: Intel Core i7 -3770K running on 3.50GHz
GPU: MSI Nvidia GTX 980 with 4GB of memory
Motherboard: MSI Z77A - G43
RAM: 16 GB
Hard drive: Sandisk Extreme SSD with 120GB
OS: Windows 10, 64 bit
The testing algorithms were running with Java version 1.8.0_101, Python version 3.7.2 32 bit and Nodejs version V10.15.1.
3. Literature review

3.1 Python, Java and Javascript - Overview

3.1.1 Python

Python is a programming language that was developed during the later parts of 1980’s and the beginning of 1990’s by Guido van Russom in the Netherlands [2]. Python originates and are heavily influenced by other languages like for example Java, C, C++ and Smalltalk. Python is a dynamically typed programming language which means that the variables can change during the execution and the language does not force the programmer to define which datatype it is, the language will do that for the programmer. This is one of the reasons to why the language is often seen as a beginner friendly language since it does not require as much computer science knowledge to get started [2]. Python is an interpreted language instead of a compiled one and it has features for object oriented programming using classes [9, 10]. Python has grown over the years and has established a large community that uses it for a variety of different programs and is now one of the most popular programming languages which can be seen in figure 2. The programming language has a large support for most features that is required for a general purpose language. For example it supports usage of web frameworks, desktop application frameworks and has a wide range of libraries [2].

<table>
<thead>
<tr>
<th>Jan 2019</th>
<th>Jan 2018</th>
<th>Change</th>
<th>Programming Language</th>
<th>Ratings</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Java</td>
<td>16.904%</td>
<td>+2.69%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td>C</td>
<td>13.337%</td>
<td>+2.30%</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>↑</td>
<td>Python</td>
<td>8.294%</td>
<td>+3.62%</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>↓</td>
<td>C++</td>
<td>8.108%</td>
<td>+2.50%</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>↑</td>
<td>Visual Basic .NET</td>
<td>6.459%</td>
<td>+3.20%</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
<td>JavaScript</td>
<td>3.302%</td>
<td>-0.16%</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>↓</td>
<td>C#</td>
<td>3.284%</td>
<td>-0.47%</td>
</tr>
</tbody>
</table>

Figure 2. Statistics of the most popular programming languages in January 2019 [1]
3.1.2 Java

Java is a programming language that was developed in 1991 by James Gosling at Sun Microsystems and released in 1995 [2]. One of the big selling points when it comes to Java is its “write once run everywhere” feature that makes the programming language platform independent. When running a Java application, the compiler will, instead of generating platform dependent machine code, compile it to byte code. The byte code will then be executed in Java’s own virtual machine when running the application [2]. This means that the platform needs to have a JVM (Java virtual machine) installed in order to run the application, which makes it simple to run the same application on multiple platforms. Oracle is the current developer of the language and they provide different versions of the language that can be used for different things. For example, there is an embedded version of Java that is more lightweight and has other important features for embedded systems. There is also another enterprise edition that gives developers the latest features of Java [11]. Java is an object-oriented programming language that has some common object orientation features implemented such as classes and methods [12]. The language is considered to be multi purpose and can be used in a variety of different applications such as web, Android and desktop applications [2].

3.1.3 Javascript

Javascript was developed in 1995 by Brendan Eich who would continue to be one of the founders of mozilla.org in 1998 [13]. The language was technically trademarked from Sun Microsystems. When Netscape (Mozilla today) submitted Javascript for standardization to the ECMA the trademark problems occurred and Javascript was named “ECMAScript” instead [14]. Javascript is an interpreted language with object-oriented capabilities, such as the ability to create classes and objects [14]. In addition to being interpreted it is also dynamically typed which was explained in the Python paragraph above. Javascript is considered to be the programming language of the web due to the fact that it is used in the development of a lot of web pages. All modern web browsers have implemented the language due to the usefulness of it [14]. Java and Javascript are often associated with each other, however the languages are very different from each other [14], which will be explained in this thesis. Although initially being considered only a scripting language, Javascript has through the years continually gotten new implementations which has made it into a valid and solid general purpose language [14]. With the help of runtime environments such as Nodejs, programmers are given more tools for the same language. Nodejs is is made to help build backend servers in Javascript [3].
3.2 Related work

When looking at related work and articles, a lot would indicate that Python seems from a beginners point of view to be the best or simplest to understand in terms of syntax. This claim is based from schools and teachers who have written articles and performed tests about having or changing to Python as their introduction language for first year students, with the reasoning that Python is the simplest way to introduce programming as of now to novice programmers. One of these articles would be “Teaching an Object-Oriented CS1 — with Python” [4] which documents the experience of introducing python to students who attended the course CS1. The authors concluded the the experiment was a success, claiming:

“We have been very pleased with the use of Python, finding that it affords a clear, coherent, and consistent presentation of object-oriented programming. Our initial batch of Python CS1 students are progressing through the curriculum, and anecdotally the change has been successful.” [4]

However, some articles also emphasize that learning Java is beneficial and will be taught later on in the course as a way to make preparing for the later courses easier. Which is the case with the article “Enjoying Python, Processing, and Java in CS1” [15]. The article revolves around setting up an introductory programming course that is as simple to get into as possible, but with enough programming to know how to be able to continue with the program. The course starts with Python as the writers consider it the best choice to introduce to new students. The course then progresses into Processing and Java, and to finish the course, a project of the students choice. Here the authors reported that 17 students were happy with the course, 4 would prefer to focus solely on Java and the last two would skip Python or Processing [15]. The article “An Empirical Investigation into Programming Language Syntax“ [16] takes a look at different aspects from general purpose languages syntax which involves asking programmers and non-programmers alike questions about syntax word choices and how intuitive they are. The information conveys a mixed result with some words being very fitting for what they do such as “if” which is basically checking a condition and doing something if that condition is true, to syntax that does not conform to the top chosen words of the questionnaires, which could be “array” [16]. It is important to note that the top word choices vary heavily at times between the programmers and non-programmers and therefore makes answering the question of how intuitive the syntax word choices really are. Even if the writers of the articles mentioned in this literature review consider Python to be one of, if not the simplest programming language to start with [15, 4], “An Empirical Investigation into Programming Language Syntax“ [16] suggests that there may still be simpler languages to be created which has a more intuitive syntax word choice [16]. However there is a question to be asked: If a programming language would conform to the preferred
syntax word choices of most novices, how difficult would it be to transition from one of these languages to a more traditional language such as C++ or Java?

Regarding the technical differences between the chosen languages, some researchers did a study on the energy efficiency of a wide range of programming languages. In the research paper “Analyzing Programming Languages’ Energy Consumption: An Empirical Study” [5] the researchers did an analysis and comparison between different programming languages to see how much energy they use when doing small tasks and how they perform in comparison to each other. This was done to give developers more knowledge in terms of energy efficiency in programming and to determine which programming language to use for particular instances. One interesting discovery that was made in the article was that for the tests they made, compiled programming languages tended to have a lower total energy consumption compared to interpreted languages [5].

As one might notice, there are many ways to compare languages, it could be to compare their energy efficiency like was previously discussed or make a comparison on their performance and how fast they execute programming tasks. However, the authors did not find any article that focused on the same research questions and with the same languages.
4. Result

4.1 The differences between the languages (RQ1)

To answer research question number one, the authors are reviewing what the major differences are by going through every part of the programming languages (according to the limitations) and writing down how they are similar and how they differ. In the discussion chapter, an analysis will be found where we discuss the key findings in the research. All the figures that are mentioned in this chapter can be found in appendix A.

4.1.1 Variables

Java, Javascript and Python are heavily dependent on the use of variables. Variables are used to save different types of data that can be used in different places in the program.

The way that variables work in python is straightforward, there is no need to write the data type before the variable name and the assignment of data [17]. Since Python is a dynamically typed language, each variable can change data type dynamically during run time [2]. A variable that holds text can suddenly represent a number instead. Python do not allow the programmer to assign the data type, it is not possible to create a variable with no content, you need to assign a value to it. However, it is possible to work around this problem by for example setting a variable name equal to an empty string. Figure A1 shows how to create different variables and also how “number2” can change its data type dynamically.

Javascripts variables are similar to Python in how they do not need to declare a specific data type. It is possible to change the data a variable can hold by assigning it at another point in the program (shown in figure A2) [18]. As can be seen in the figures, the way that variables work and is declared is very similar in Javascript and Python.

Figure A3 shows Javas need to specify what data type a variable needs to have in order to make use of it [19]. It is possible to parse the value of a variable into another data type in another variable using methods like “valueOf”[20]. However, it is not as simple as it is in javascript or python where there is no need to make use of functions in order to change what data a variable contains (shown in figure A4).
4.1.2 Comments

Python makes use of # for single line commenting (see figure A5) and does not support a formal block comment feature [21]. However, the authors discovered that it is possible to simulate a block comment by converting the desired code into a string. This is not an elegant solution and should preferably not be used. Figure A6 shows how to simulate a block comment.

Java and Javascript both make use of the // for single line commenting and /* */ for block commenting [22, 23]. Figure A7 and A8 shows Javas and Javascripts single line and block commenting respectively.

4.1.3 Separators

Python has no symbol to declare a line break and instead uses whitespace to see if a break has occurred, that is if a newline token has been generated. Python has the ability to line join rows of code [24]. The situations where line breaks can be used is in if statements that have several factors to check in order to make it more readable (shown in figure A9).

Java uses semicolons to separate each statement from each other. Each statement needs to stop with a semicolon like in figure A10 otherwise it will give an error message [25]. Javascript uses both semicolons and line break as statement separators. This means that there is no need to separate each statement with a semicolon, it is enough to use only line break. However, if there are more than one statement on the same row, then it will be required to have a semicolon to separate them [23]. Figure A11 shows how line breaks and semicolons can be used.

4.1.4 Indentations or curly brackets

Python does not use curly brackets to encapsulate code, instead it uses whitespace indentations to determine if code belongs to for example a function or an if statement. The indentation feature is implemented using a stack that uses indent and dedent tokens [24], the indentation level in the stack is compared for every row in the program, if the levels between the top stack token and the row the program is currently reading are the same, there is no change to be made. If the level is larger it will be pushed onto the stack. If the level is lower, the program will pop elements from the stack until it finds the correct level number. For each level added or removed from the stack an indent or dedent token will be generated respectively [24]. Figure A12 shows an example of how indentation is used, the first print method will only execute its task if the condition is met, whereas the second one will always happen.
Both Java and Javascript make use of curly brackets to encapsulate code. The curly brackets will determine whether or not a statement belongs to an if statement and/or whatever else. It is important to note that curly brackets are not necessary for statements that only have one line directly under it as shown in figure A13 and A14, the line of code in the if statement will execute if the condition is met [26, 27].

4.1.5 Selection statements

Selection statements are statements that give the programmer the ability to control the code and execute code depending on different variables and parameters. Typically one of the most common control flow mechanisms are if - statements, which can be seen in figure A15. These statements have some additions, such as else if (elif in Python) which checks if a conditions is true and if it is it will not check any other else if - statement related to the original if statement. The second addition is the else-statement, if none of the other conditions are true the code in the else block will execute [28, 26, 27]. Another useful selection statement is the switch case which is used more as a way to check character and/or integer keyboard commands although it has other uses. Rather then having to write several if-statements, switch cases tries to reduce the amount of repetitive code and offer more readability.

Java and Javascript both encapsulate code in if statements with the use of curly brackets whereas python makes use of indentations. The same selection statement logic applies across all the languages and there is no major difference between the three [26, 27, 28].

When it comes to switch cases, there is no built in functionality for Python [28]. Java and Javascript implements the same syntax and functionality in their switch cases [29, 30], which can be seen in figure A16 and A17.

4.1.6 Loops

Loops are a standard feature in programming languages with the functionality to be able to iterate n amount of times through data structures, numbers etc. Loops are heavily used in all parts of programming and of course Java, Javascript and Python have all implemented them. The structure and syntax however is somewhat different for all of them. Java and Javascript basic for loops follow a similar structure. A variable is initiated with a value in the first statement followed by a condition checking if the variable mentioned previously is smaller than another number or length of a data structure. After the condition the variable will be incremented after every iteration [31, 32]. These loops can be seen in figure A18 and A19. Python does not need to define the start value as Java or
Javascript does and instead makes use of the range function to define how many times the loop should happen. The basic for loops in Python can be seen in figure A20. For loops that does not make use of the range function in Python resemble the enhanced for loops in Java, were it creates an item and iterates through a data structure. It is then possible to use the item as it has been assigned the value of the element of the array the loop is currently in. An example of an enhanced for loop in java can be seen in figure A21. An interesting addition to Pythons for loop is the addition of an else statement, something that does not exist in Java or Javascript [31, 28].

While loops are present in all three programming languages and they work the same way, it is possible to set a condition and continue looping until the condition is met. While loops could also be used in situations when a developer wishes for the loop to continue forever such as a while true loop. Java and Javascript both have the additional features of do while loops, which means that the loop will execute one time regardless if the conditions is true or not. Python does not have any do while loops [28, 32, 33].

There are control statements that can be utilized in order to manage code in a simpler way. These statements are: Break, continue, return. Break simply makes it possible to break out of a loop before it is finished iterating. Continue has the capability to make the loop go to the next iteration immediately. Return can be used in situations were something in an if statement or loop is found, it is possible to return that variable immediately. These statements exists in all three languages [28, 34, 35, 36].

Python has another addition with the pass statement, it is a simple null operation meaning that it can be a good placeholder for code that has not been implemented yet [37].

4.1.7 Data structures

4.1.7.1 Array

Arrays are similar in all three programming languages, however they are implemented a bit different in Python and Javascript because of the fact that these two languages are not dependent on a specific data type [38, 39]. Python has the ability to import the array functionality which enforces an array to have a specific data type [38, 39]. Java by default requires the array to be bound to a specific datatype when initialized [40]. There are some fundamental differences in how arrays are treated in the three languages. Javascript has no way of enforcing a data type into the array making it more similar to a list in the other languages. Java also does not allow changing the size of the array whereas the other two do [39, 40]. How Java and Javascript creates arrays can be seen in figure A21 and A22. The Python implementation can be seen in figure A23.
4.1.7.2 Array Lists

Array lists are like arrays except that they change size depending on how many elements it contains [41]. The standard implementation of arrays in Python are already dynamically sized which means that they do not need a specific implementation of array lists [38]. Javascript does not support an implementation of array lists, however with the use of the built in methods it is possible to use a normal array like an array list [39]. Java has its own built in arraylist that can be used by importing the package and then create an object of it [41]. An array list in Java can be used like in figure A24.

4.1.7.3 List

Lists works in a similar fashion across the three programming languages, they are all dynamically sized and have the possibility to contain different data types (Java is bound to a specific data type in the same list) [42, 43, 39] Note that Javascript claims that their arrays are list like objects [39]. However, Java has the ability to create objects out of different types of lists, such as arraylist, linked list and stacks etc [42]. The reason for this is because the list class is abstract and can therefore not be declared [42]. If the programmer wants a list more similar to how they act in Javascript and Python, it is possible to create a list object out of an arraylist[42].

There are methods that acts differently that result in a similar outcome, which makes comparing the depth and possibilities of lists in the three languages difficult, as an example Python has “count” method which returns how many times an element appears in a list whereas Javas “contains” and Javascripts “include” returns a boolean if the list contains the specified parameter [39, 43, 42]. How lists are created in Java can be seen in figure A25.

4.1.7.4 Sets

The basic sets works the same way in all three languages, it will automatically remove duplicate entries. Java, Javascript and Pythons set objects have similar methods, this includes being able to add, remove, get values at indexes, get the total size of the set and so on. Python adds additional mathematical operations for sets which are union, intersection, difference and symmetric difference. Javascript and Java does not have these operators built in to their sets [44, 45, 46].

Union: All elements occur in both sets. Intersection: When elements occur in both sets. Difference: Elements only exists in a specified set, but not the other. Symmetric difference: Elements that exists in either set, it can not exists in both [43]. Figure A26, A27 and A28 show the basic implementations of sets in the three languages.
4.1.7.5 Tuples

Tuples are data types that are initiated and can have several values associated with them. The big difference between the tuple and list is that the elements in the tuple can not be altered. The output from tuples also consists of parentheses, a reason for this is because of the fact that tuples can be nested and needs to be displayed correctly. This changes how indexing works, if there are two nested tuples in a tuple, when printing an index it will print the entire tuplet which is located at that index. Figure A29 and A30 shows the code to basic tuplet operations and the output from the code [47]. Tuples is a data structure that only officially exists in Python out of the three languages.

4.1.7.6 Stacks and queues

Java has built in support for stacks and queues that can be used by importing the stack or queue class and creating an object out of it. The classes has the standard methods that are used for stacks and queues which is: push, pop, peek for stacks and add, remove and peek for queues [48, 49]. Figure A31 and A32 shows how to create and use a stack and queue in Java.

Python has no official stack. However, because of the way lists are implemented it can easily be handled as one with the use of the “append” and “pop” methods. It works well because there is no shifting involved which is a problem for a python programs that tries to make use of lists when creating queues. Although it is possible to make a Python list into a queue, it is not efficient and therefore should be avoided in favour of the dequeue function that can be imported, dequeue gives the ability for fast pops from the start of the list. Figure A33 and A34 shows the basic functionality of stacks and queues in python. Python has no built in peek function to display the head of the stack or queue [43].

Javascript has a similar implementation of stacks and queues as Python where it does not have the data structures built in to the language or the standard library. However, with the help of the methods for arrays the programmer can easily manipulate the array to work as a stack or queue [39]. The collection can be used as a stack by using the push and pop methods for adding or removing elements. How this is done and how similar it is to Python can be seen in figure A35. The collection can be used as a queue by using a method called shift instead of pop (see figure A36). Shift is used to remove the first index of the array which is similar to the way that a standard dequeue function is working.
### 4.1.7.7 Dictionaries

Dictionary is a useful data structure that Python has built in support for, in other languages like Java it is called hashmaps [43, 50]. Dictionaries differs a bit from normal arrays by not having each element indexed by numbers, instead it uses something called keys that is associated to each element. Dictionaries can be useful in many situations, especially if the programmer wants to access elements of the dictionary often but does not know where in the collection it will be saved. How dictionaries can be used and created in Python can be seen in figure A37. In figure A37 it is shown how to use dictionary as a collection of telephone numbers and how to use it for saving different information about the same person. Note that the elements are accessed by passing the key value instead of a number.

Java uses hashmaps instead of dictionaries but it can be used in the same way. However, the programmer needs to define which data type will be the key and the value [50]. This is one way the programming languages differentiates. A typical implementation of a hashmap in Java can be seen in figure A38.

Javascript has no official support for something called dictionaries or hashmaps but with the help of the object class one can simply use that to simulate a dictionary [51]. How that is implemented in Javascript can be seen in figure A39. Like Python, Javascript gives the programmer the ability to have different data types in the same collection.

### 4.1.8 Classes

Classes are a staple in object oriented programming all chosen languages namely Python, Java and Javascript have them [52, 53, 54]. During the tests of writing a nested class inside another class, the authors discovered that it was not possible to write the class code after the code trying to access it. After the class has been initiated the programmer is free to call whatever method they wish in the (according to access modifiers) class. Python and Javascript supports importing classes like Java does, but there is for the most part only similar things one can do with them. An interesting thing to mention is the fact that Python can inherit multiple classes [52]. The authors also discovered that Javascript has the possibility to create a constructor and when the object is made, it is not required to give any input into the constructor. The syntax will not encounter an exception, but rather when the programmer tries to use a variable that is initiated in the constructor it will return that the variable is undefined, which is accurate (shown in figure A40). Figure A41, A42, A43 and A44 shows how to access a class in Python, Java and Javascript and also how to export a class in Javascript.
4.1.9 Functions and Methods

All three languages has the possibility to create methods which can be used to execute a block of code at a specific time [55, 56, 57]. Like with classes, Python has to create its methods before a call to them can be made. Javascript works around this problem by using hoisting in its variable and function declarations, which basically mean that all the declarations are put into memory when the program is compiling [58]. There are additional differences with the one being how to handle return methods. Java requires the user to specify in the creation of the method if there should be something returned once the method is finished, it is also required to declare which datatype should be returned as well [56]. Python and Javascript does not have this limitation and can therefore return any variable [55, 57]. The same principle applies to parameters/arguments where Java needs to specify which datatypes are expected. It is important to note that Java has the functionality to implement generic methods which basically means it will handle whatever the user enters into the method. So the ability to specify a data type is not necessarily a bad thing but rather gives the option to streamline code and to set guidelines for it. How function/methods are structured and called is shown in figure A45, A46 and A47.

4.1.10 Error handling

Errors and exceptions can happen during execution of a program. Many programming languages like Java separates errors from exception and give them different meaning [59, 60]. Typically an exception is something that can be handled during execution and errors are other problems that should not be handled [59, 60]. In order to be able to handle this problems, the programmer must understand how an error is catched.

Python has two types of problems that can occur before or during execution, syntax errors and exceptions. Syntax errors occur when the Python parser does not understand the code that has been written [61]. Exceptions are problems that typically can happen during run time and can be for example that the programmer is dividing something with zero, these are things that can happen even if the code is written correct and follows the language standard [61]. All exceptions can be of different types depending on what problem occurred, these types are built in to Python to distinguish between what exception occurred and to help the programmer know what happened [61]. An exception can be defined as a subclass to the BaseException class [62]. To programmatically handle an exception the programmer can use the “try” and “except” keyword [61]. In figure A48 the try - except clause has been implemented. It works in the following way, in the try indentation it will execute the code and if any error occurs (in this case it is a “TypeError”) it will call the except section and execute that indentation and skip the rest of the part in the try clause. However, if no exception happens in this
section, it will simply skip the except part and continue. If another exception happens in the try clause that is not handled, it will stop the execution and show a message [61]. The error handling part can also have multiple excepts after each other (see figure A49 to handle different types of exceptions [61]). Compared to Python, Javas way of writing error handling code is very similar. In figure A50 the code shows that the try block is just like Python, this is where you put the code that might trigger an exception [63]. After the try block, the programmer is required to put a “catch” block, this function is similar to “except” in Python which was previously discussed [64]. The programmer needs to pass it the exception that it should handle and its possible to add multiple catch blocks after each other [64]. There is also a way of always executing some code after every try - catch part using the keyword “finally” [65]. How try - catch can be used in Java can be seen in figure A50. In the third programming language, Javascript, the error handling is also very similar (see figure A51). It uses the same keywords as Java but it is used in a somewhat different manner. Just like the other two languages, Javascript uses a try block where the programmer can place the code that may trigger an exception [66]. However, when it comes to catching the exception, the programming language does not require the programmer to specify which exception it should handle, all the exceptions can be handled in the same block [66]. Javascript also supports the use of a finally block which works in the same way as in Java [66].

4.1.11 File management

When it comes to writing to files in different programming languages, the way of doing it can be a bit different. The file management capabilities for Javascript used in this chapter is the Nodejs features for doing these tasks instead of normal Javascript. In Nodejs one can use the “require” keyword to import the file system feature. This object can then be used to call the “writeFile” function which lets the programmer to give the file a name, some information that will be in the file, and pass it a function [67]. How this is done in practice can be seen in figure A52. When it comes to reading from a file, the way it works is similar to writing to files except that another method is used, namely “readFile” [67]. Javascript makes use of callback functions meaning that after the function to read or write to the file is made, the control will be reassigned to the Nodejs execution environment. This means that there is no waiting for the I/O command to complete before starting another task [67]. Java has a more varied way of writing to files since it needs multiple objects to use the features of reading and writing. There are many ways of writing to file in java using different objects. All these ways of writing to file can be used in different situations and should be picked depending on what task the programmer wants to solve. In this example, the authors is going to mention one way to write to, and read from files using a filewriter. A programmer can use this functionality by creating a filewriter object that is connected to a file. When the object has been created, it will be available to use for writing to a file using the
write method \[68\]. How this is done, can be seen in figure A53. Just like writing to a file can be done in many ways, also reading from a file can be done differently. In this case, a bufferedreader is used. In figure A53, a bufferedreader is created with a filereader as a parameter. When the object is created, it is used to read every line from the file \[69, 70\]. Python has another way of reading and writing to files, and it works in the following way:

The open method will be called and a file object is made, the arguments for the most part consists of the file name and what type of operation the programmer wishes to do with the file. These operations consists of r (read), w (write), x (create file if it does not exist), a (append to the end of the file), b (binary), t (text mode) and + (update a disk file) \[71\]. The keyword “with” is useful as it closes when the file task is completed. File objects can also call the close function to close the file manually. File objects in python have functions for different operations on the file. Such as the ability to read specific lines of the file or the file as a whole \[71\]. How this is done in Python can be seen in figure A54.

4.1.12 Operators

There are several operators that are standardized in Java, Javascript and Python. A lot of the operators are similar and there is different groups of operators such as Assignment (“a=b” and “a+=b”), comparison (“a==b” and “a!=b”), arithmetic (“a%b”, “a++”), bitwise operators (“a&b”, “a|b”), bitwise shift operators (“<<”, “>>”) and logical operators (“&&”, “||”) \[72, 73\]. Python does however have some interesting additions such as matrix multiplication with the “@” symbol \[74\].

4.2 What can the languages be used for? (RQ2)

The languages chosen in this thesis are all general purpose languages and can be used for a multitude of different things \[2, 12, 14\]. However, one language may not be best for every type of application and the authors has therefore conducted a literature review which can be found in this chapter. The chapter revolves around previous research done in the field and what people have found useful with each language while using them.

4.2.1 Python usage areas

Python is a general purpose programming language that has many areas where it can be used. On their website they list a wide variety of frameworks, libraries and tools which can be used for developing different type of applications. According to a survey by Jetbrains, the most used areas for python was its scripting, machine learning and web development prospects \[75\]. The authors will therefore
investigate these areas more closely in this chapter to motivate why they might exceed at these specific fields.

4.2.1.1 Python machine learning

As was mentioned in the literature review and introduction to Python, the language has become one of the most popular language over the years. Some sources indicate that it is because of AI and machine learning [76]. AI and Machine learning is the process of feeding an application a large amount of data in which it uses to learn patterns, match data and performs tasks based on this information. The reason why Python is considered to be one of, if not the best machine learning language is because of the flexibility and support it holds, where it uses parts of both general purpose programming languages and task specific scripting languages in order to create robust and easy to code machine learning applications. Python also supports use of several libraries in order to load, visualize, and perform statistical analysis on data, etc. These commonly used libraries for Machine learning and AI includes “scikit-learn” “NumPy” and “SciPy” which all have their own features. Numpy is a popular library that gives the programmer more extensive features for doing mathematical operations in Python. SciPy is another library that is often used in machine learning which contains a wide collection of functions that can be useful in scientific computing. For example it has features for signal processing and statistical distribution. A third very prominent library is the scikit-learn tool that is built on top of the previously mentioned libraries (SciPy and NumPy) which has a collection of machine learning algorithms that can be used for free [77].

4.2.1.2 Python automation

The process of making small programs to automate tasks which would take a long time for humans to do manually can be of great value for not only software companies, but also for private persons and small to big corporations who work in other areas outside software development. One of the most popular applications to develop using Python, is small automation scripts [75]. This might be because it is easy to learn compared to other languages and often requires less code which was discussed in RQ1. Tasks such as renaming files, sending out a large amount of emails to friends and/or coworkers is time consuming for a human to do manually and could be done in a matter of seconds after a basic implementation has been done. Therefore it is not as important to write optimized code because it will still save lots of time if the programmer is fast at writing the code, especially if the program is saved and can be used repeatedly. An example of this could be in the case of scraping information from websites, the process of doing this is very simple through the use of python and its many libraries. One of these libraries is called “beautiful soup”, that can be used to parse HTML and XML pages. A built in feature that comes with Python is “webbrowser” which gives the programmer a way to open
websites programmatically [78]. Combining different libraries and tools can make Python very powerful and extend the amount of features provided by the standard libraries.

Another situation where automation can be used is when organizing files on the hard drive. This can be done with the help of “shutil” which comes with Python and can easily be imported to the program. Shutil has features for copying, deleting and managing files in python using only a few lines of code [78]. Since Python generally requires less code, it adds to how fast you can write the program and make changes to it in the future.

### 4.2.1.3 Python web development

Python has a web development framework called django, which in the stack overflow survey is among the most popular web frameworks [79]. According to another survey done by JetBrains, Pythons code is often used together with Javascript, HTML and CSS which indicates that the language is popular among web developers [75]. Django can be used as both a frontend and/or backend application when developing web sites. It is similar to the MVC (model view controller) but has some differences, django uses MTV(model template view) which performs the controllers tasks by the framework who then sends request to the correct view. With knowledge in python and some basic understanding of the framework, it is possible to develop a webpage using django [80, 81].

Django is used in some popular applications on the web like for example Pinterest and Instagram, which proves that the framework works great for large applications [81]. However, Django is not the only web framework for Python, it also supports other micro frameworks like flask and bottle [82, 83]. A micro framework is more lightweight compared to a full framework such as django and is therefore useful in smaller web applications which does not require all features of a more full framework [83].

### 4.2.2 Java usage areas

Java is, just as Python, a general purpose language. However, the language has more of a tradition of being considered a general purpose language compared to the previous mentioned Python, which has to many developers been seen as a scripting language. The different usage areas of the language has changed over the years. It was first developed to be used in embedded systems, but was applied more in applet applications for the browser during the 90’s [12]. Java has developed over the years and are right now in a version called Java 8 [84]. The language is nowadays used in many different applications. One of the common ways to apply Java today is in Android applications [84]. However, the language is now seen as a “second language” in android development since Google has introduced
Kotlin as its official language. Another area where Java is used, is in large enterprise applications on the web with the help of the enterprise edition [11].

4.2.2.1 Java Android development

When it comes to android app development, Java is among the most popular languages due to the fact that Google decided to pick it up as the primary language for Android app development when it first was released in 2007 [85]. There is no clear evidence as to why Google decided to pick Java for developing apps for Android, but multiple sources indicate that it may be because of the platform independency that the JVM provides [85, 86]. When using Java for developing Android apps, one is using Google's own Android API instead of the original Java API. The virtual machine that the applications are running on is called “Dalvik” and was developed by Google to optimize the execution of apps on a mobile device [86]. However, Google has later introduced Kotlin as the official language for Android app development which has led to some competition between the two languages. Kotlin is designed to work together with Java and all of its frameworks/libraries, it is also possible to run the applications on the same JVM [87]. Since Google has decided to use these languages, there is no real competition between Java, Python, and Javascript when it comes to Android development. However, Javascript is very popular when it comes to web development and its possible to run Javascript code in the browser on a mobile phone, but there is still a big difference between Android applications and web applications.

Android applications are being developed to solve and simplify tasks in all fields and areas of the world, such as health related topics. There are ways to use a smartphone in order to recognize how well ones lungs function and if there are any symptoms towards lung diseases such as asthma and cystic fibrosis. By using these apps there are benefits, such as saving money and time for hospitals and patients [88].

4.2.2.2 Java full stack development

Full stack development involves several programming fields like for example frontend and backend development. As mentioned in the beginning of this chapter Java was used a lot in early applet programs for the web during the 90’s [12]. However, the web development environment has changed a lot since then and other languages like Javascript is more popular for web development today but Java is still not irrelevant since there are still many people that uses java based frameworks such as Spring [79, 89]. Oracle has developed a platform, they call JEE (Java Enterprise Edition) which is the platform to use when developing web applications in Java. It has support for many web frameworks and tools such as Spring and Hibernate which is useful for testing, database access, etc [11, 89, 90]. It is very common to use these frameworks and tools in full stack applications, Spring seems to be the
most popular framework since it is the only Java framework that was mentioned among the most popular web frameworks in stack overflows survey from 2019 [79]. JEE was developed to give programmers the ability to create multi-tiered applications where the tiers have different responsibilities for example to access database and business logic [91]. It should be noted that JEE is just a platform developed by Oracle to give programmers a large set of tools for developing applications that need many different components, it is not another language [91]. The JEE platform is modular which means that the programmer can combine different components and write them independent of each other and then connect them in the same environment, this makes it possible to write both the front-end and back-end in Java (together with HTML) [91]. Two main components in JEE applications are JSP (Java Server Pages) and servlets which both serve their own purpose. JSPs are used to define how the each page of the web application shall look, it provides the feature to write Java code inside HTML, this makes it possible to create dynamic web pages. Servlets are one of the other main components which is often used together with JSPs to handle request that are made from the user interface [91].

4.2.3 Javascript usage areas
Javascript was first developed for the web to work as a scripting language that could easily be written inside an HTML page. The language has grown a lot over the years and the usage areas has changed a lot since the beginning. The language can now be used for both front-end, back-end and even desktop applications which means that the language can be considered a general purpose language [14, 92]. Two major strengths of Javascript and the reason Javascript is used a lot today is due to the fact that it is flexible and handles error in a special way, a Javascript program has the ability to not crash when a runtime error is encountered and instead continues executing future tasks [92]. These two factors allows developers to implement and expand on existing applications quickly which can be highly desired in both commercial applications and small scripts [92].

4.2.3.1 Javascript full stack development
When it comes to web development, Javascript is considered to be “The programming language of the web” [14]. Most modern, if not all, browsers have support for Javascript, this includes desktop, game consoles, tablets and smartphones [14]. With the amount of frameworks and libraries which allows for additional web development features, such as Angularjs and Reactjs, there is no question about whether or not Javascript is still popular in web development. According to the stackoverflow survey which spanned 63585 responses, the three most popular web frameworks and libraries were all related to Javascript [79]. The development of Javascript follows the overall development of the web where it
was first developed to integrate code logic inside HTML pages but has since then developed new
techniques for syncing the web page directly with for example a database. This can be done with
React which is developed by Facebook to keep applications UI directly synchronized with the
backend changes. One example of when this can be useful is when having a friend list on a web page
that needs to be updated if anyone logs in or out without having to refresh the page in the browser
[93]. Another main feature of React is that it let’s the developer write different components that are
independent from each other to combine them on the same page [94]. A typical use case for this can
be on the feed page on Facebook where you can see the latest updates in one component, the friends
list in another component etc. Angular is a framework that has similar features as React with features
like writing module based applications and having direct synchronization between the UI and backend
[95]. Another popular framework is Vue which is a little different from React and Angular, it focuses
more on the view (in the Model-view-controller workflow) part of the application and can be used in
combination with other tools to create single page applications. However, it is not focused on single
page applications but it is similar to the other frameworks by also having the feature to build pages
with different components [96]. Although Javascript is known for its front end development, new
runtime are being introduced to the language. One of these is Nodejs, which allows developers to
write code for the server/backend using Javascript. Some of the core features of Nodejs is that it is
event driven and has a non-blocking I/O [97]. Blocking is the concept of a process having to idle and
wait for something to complete and respond before said operation can continue. Non-blocking
programs instead notify the process when the request is done, and therefore the process does not have
to wait for the response and waste time [98]. The Javascript code that is running in Nodejs is executed
on one single thread which makes it very lightweight and highly scalable [97, 98]. Nodejs allows
frameworks such as express to be added on top of it in order to simplify the development of server
applications. Some of the major things these simplifications include is reducing the complexity when
for example, sending files which is more difficult if one were to try to accomplish the same thing in
only Nodejs. Express is also good for splitting up request handlers in smaller and simpler controllers
that only handle specific requests. The reason why this is so helpful is because of the fact that code
becomes more modular and bite sized, which makes it easier to change, scale up and maintain in the
future [99]. According to the “State of Javascript 2018” survey Express is the most popular backend
frameworks for Javascript and is well liked since many people would like to use it again after having
tried it [100].

4.2.3.2 Javascript mobile development

When it comes to mobile development, Javascript has the possibility with tools such as React-native
to develop full mobile applications [101]. As the name suggests, the React-native framework is based
on the previously mentioned React library and works in a very similar way and it makes it possible to develop another type of applications with the same knowledge which makes it easy for normal React web developers to develop mobile applications [101]. It is worth noting that the application that is built in the framework is not just an application that is running in a browser on a mobile device, it is actually calling the platforms own API to render the UI and access the devices own sensors etc. The framework is cross platform for both Android and IOS where most of the code can be reused on both platforms which makes it fast to write applications for both operating systems [101]. However, React-native is not the only tool that can be used to develop mobile applications using javascript. Some lesser known frameworks that supports javascript mobile development includes Cordova, Ionic and Nativescript [102].

4.2.3.3 Javascript Desktop application development

In the early days of Javascript, the language was not created for developing either mobile applications or desktop applications. Although many applications has moved in to the web, the language has evolved and some frameworks and tools has been created to develop desktop applications with Javascript [92]. One of the more well known frameworks for these type of applications are Electron [102] which brings in CSS, HTML and Javascript to the desktop environment. The framework consists of two other tools that are often used in Javascript development, Nodejs and Chromium [103]. Nodejs was previously discussed and Chromium comes packaged with Electron, it is used for rendering and showing web pages [103]. Unlike React-native (which was used for mobile application development) the applications that are developed in Electron is actually running in a browser process and this is what Chromium is used for [103]. Normal web applications are also characterized by only being able to run in a web browser which limits the program to access certain parts of the operating system resources like the file system etc. If someone wants to develop a desktop application then it might be desirable to have access to these resources and that is one reason to why Nodejs comes packaged together with Electron [103]. Nodejs gives programmers the ability to call its API to interact with low level system resources inside the browser windows [103]. The framework has gained much attraction and is now used in large popular application like: Discord, Skype and Visual Studio Code [104].
4.2.4 Comparison on different usage areas between the languages

Based on the previously discussed languages and their usage areas, the venn diagram (See figure 3) below describes each languages main usage areas and how they compare to each other.

![Venn diagram of what each language is prefered to be used for according to the findings](image)

As the venn diagram shows and the paragraphs above mention, full stack development is something that is possible to do in all three languages but the way the development is done differs. Java web applications are often developed in JEE (Java Enterprise Edition) which has both the front-end and back-backend part of the application on the same platform using a multi tier architecture [91]. The same applies to Python since it is possible to develop both front-end and back-end in Django [80]. However, Python has a collection of different frameworks to choose from compared to Java that has the possibility to run frameworks inside the JEE platform [81, 82, 89]. Javascript approach is different as the front end side can be developed in a completely different framework while using Nodejs to run the backend code. It is also possible to add snippets of Javascript code inside the html which does not require any library or framework, as long as the browser that runs the application has support for Javascript.
Desktop application development is something which all three languages can accomplish, however the surveys would suggest that it is not as popular to develop these types of applications in Python or Java [75, 105]. Although the languages are not popular for this area, it is possible to do with frameworks such as Javafx for Java and Tkinter for Python [106, 107]. Modern frameworks like electron is being used to develop big and widely used Javascript applications such as discord and visual studio code [103, 104].

When it comes to developing machine learning applications or algorithms, none of the languages is stopping the programmer from doing this because it is the process of giving the program some information and then develop algorithms to analyze and learn from the information for a certain reason. Compared to the other usage areas it is a bit different because the application must not run on a specific platform like web applications that often requires a browser. This is why the type of application can be developed in any of the languages. In terms of popularity among languages in machine learning, Python is the clear winner and it is also the most popular usage area for Python [75, 108]. Java does have some usage within this area, but it is important to realize that its popularity is decreasing. Javascript although being supported by machine learning libraries such as tensorflow [109], is still not a big part of the discussion [108].

The “Automation script” term is quite wide and can involve many things from gathering some information and writing it to a file automatically to machine learning algorithms. However, in this part of the thesis, the term is used to describe scripts that does small tasks on the computer to help users with certain things. Typically an automation script can be used to rename a set of files in a folder or copy a large set of files from one folder to another. Just like with machine learning, there is none of the chosen languages that does not allow automation scripting but there are certain features of the languages that may be desired when developing these type of scripts, for example it is important that one can implement the code faster than doing the task manually. This might be the reason to why automation is among the most popular usage areas of python [75] because it often requires one to write less code. It also has a big collection of libraries like “beautiful soup” and “shutil” which can be useful when developing automation scripts [78]. However, it should be noted that automation programs is possible to develop in Java and Javascript but it may not have the same support of libraries and features as Python. The state of Javascript 2018 survey did not cover this aspect which indicates that it is not very popular in this field [102]. Java has reading and writing to file features [68, 69] which is useful when automating tasks. When choosing what language to use for automation it is up to each developers personal preference because all the languages has the capabilities but there are certain things that can not be done with Javascript in the browser.
Mobile development is a field which is more platform dependent compared to for example machine learning and automation which was previously mentioned. It is possible to develop apps for Android using Java and it has been the main programming language for the platform since the beginning [85]. Although it is now seeing competition with Kotlin which is now the primary language for Android [87]. Javascript is another language that has gained some attraction when developing applications for mobile devices, the major advantage to using Javascripts React-native is the fact that applications can be written for both IOS and Android [101]. Python has the capability to develop cross platform applications [110] (which includes both IOS and Android), but it is not popular for this use case at all according to the Python developers survey 2018 results [75].
4.3 Which language is solving tasks the fastest? (RQ3)

This part of the thesis is looking at the different languages in terms of performance and how fast they run a particular task compared to each other. The chosen tasks to compare in each language is quicksort, file reading operation and the Fizzbuzz game sequence. The tasks were chosen to compare three very different tasks to see if the result would vary.

4.3.1 Quicksort

The first task presented in figure 4 shows the average completion time for the three languages with 10000 100000 and 1000000 entries to sort in a quicksort algorithm. These algorithms have been run 150 times (in the same execution) each in every size entry. When comparing the languages speed for 10000 and 100000 entries, Java and Javascript have similar result with Java being slightly faster. The differences between Java and Javascript when it came to sorting one million entries showed a bigger speed difference, however Python was significantly slower in all three cases compared to the other two languages. A conclusion that can be drawn is that for smaller lists it is completely fine to use any language to order the data entries, but when talking about lists that contains millions of numbers, Java is the clear choice out of the three languages. The algorithms that were used during the tests can be seen in figure B1, B2, B3, and B4 in appendix B.

Figure 4. Graph displaying the average execution time of quicksort over 150 runs
4.3.2 Fizzbuzz game

The result for the fizzbuzz game was fairly close for Java and Javascript with Javascript being slightly faster for the two smaller numbers and Java being the fastest for the largest number. Python was significantly slower in all three cases, even the smallest number in python was much slower than the highest number for the other two languages. The results from the testing can be seen in figure 5. The algorithms that were used during the tests can be seen in figure B5, B6, and B7 in appendix B.

Figure 5. Average execution time for fizzbuzz over five runs.
4.3.3 Reading from file

Reading a 30MB file line by line produced some interesting results where Javascript was the most efficient by quite a margin out of the three languages. Followed by Java which was somewhere in between. Python was slower by a large margin for both the languages. It is important to note that this testing was done with Nodejs features to read from file and not the standard Javascript. The results from reading from a file can be seen in figure 6. The algorithms that were used during the tests can be seen in figure B8, B9, and B10 in appendix B.

![Reading from file](attachment:reading_from_file.png)

Figure 60. Average time to read a 30MB file
5. Discussion

5.1 Discussion of research question 1

To answer the question of which language has the easiest syntax to understand, the authors considered the document review and their own testing when making the decision. Although we have a strong familiarity with Java and were not as comfortable with the other two languages, we have found Python to be the easiest language in terms of syntax. Some of the features are simpler and more intuitive compared to the other languages, which for example can be how Python implements its file management. There was no need to create several objects out of built-in classes. The major thing that was somewhat difficult to grasp as an experience Java programmer, was how python implemented its encapsulation of code due to the fact that it used indentations. However, in some aspects Python was more similar to another language such as Javascript when it came to the flexibility of the list feature. The list could be used as a queue or stack in both of these languages without having to import built-in objects. It is of importance to realize that although the conclusion is that Python has the easiest syntax, there are benefits to learning Java because of the fact that there are languages which are similar to it. Javascript felt very familiar due to the fact that it has some syntax that mirror Java.

One example of how Python differed from Javascript and Java was in the way they structure the code. Both Java and Javascript used semicolons and curly brackets as parts of the language and since Python differed in this aspect, it was the language that had the biggest difference compared to the other two. Starting with Python does not mean that all the information is useless when transitioning into other languages such as Java. It is worth considering that code in Python, while it is simpler to write in initially could cause problems because of the fact that it has less boundaries to follow compared to a more structured language such as Java. Keeping track of variables and what value they hold is significantly more difficult due to the fact that the variables require no specific data type and can also change later on in the program. This means that the code can be harder to follow while reading and can cause a lack of structure, which leads to the code being harder to maintain in a large system. Javascript suffers from the same lack of a clearly defined structure, for example semicolons being optional for the most part. Along the fact that no exception is thrown if a method with a parameter is called without having a value for that parameter. However, Java requires the programmer to adopt a more strict and structured syntax where if someone would read the code of a large system, it is more likely that the reader would have more comprehensive code to read. Such in the case of methods where the return and input parameter values has to be defined at the start. Having this structure built in can give a bigger sense of control since the programmer knows what is being sent and what is being returned. It should be noted that what is easy to understand is highly subjective and
learning the more strict syntax of Java can be beneficial for some while the more lose syntax of Python and Javascript can be easier to get in to.

5.2 Discussion of research question 2

When we (the authors) initially started research about in which field a programming language should be used, the first impression was that each language would have a specific usage area where they dominate. The authors thought that Python would be focused around the machine learning area, Java the mobile and backend area and Javascript for the most part revolving around web and frontend. These ideas were partly true, but the authors underestimated the flexibility of the languages, especially Javascript which was known to the authors as being used mainly for the web. A lot of the reason why Javascript is so flexible is because of the continued support it gets from its community, where people develop frameworks, libraries and even runtime environments in order to get the language to work in different applications. This is true for the other languages as well but maybe not to the same extent.

The impression the authors got by researching on the different libraries and frameworks was that Javascript had more to choose from while Java gave the impression that it was much based around Oracles own tools and Python was somewhere in between. The authors found that it was important to realize the difference between if a language could or should be used for a specific task. Although Javascript has the possibility to work as a language for machine learning, it may not be the best choice since the developers has to consider many aspects when choosing, for example check how it performs and which libraries and frameworks it supports and some of the aspects to look for are subjective. This made it hard to decide if there are any language that can be considered the perfect for any certain usage area, but in some cases it was pretty evident that there were one language that was superior to the rest, for example Python and machine learning. In other cases the result was a lot more undefined, which was the case when discussing mobile development, Java is an obvious choice because of the fact that android used it as its main language for several years. However, the fact that Python and Javascript both support mobile applications which can be launched cross platform makes it a very attractive choice when it comes to saving time. In other cases it was not very clear to why the language was so dominant in that field, for example when it came to automation scripting and Python.

All the languages have the ability to automate things to some extent but for some reason Python seems to really exceed at this field. The authors own speculation around this is the fact that it is easy to get in to Python and the code generally requires less time to write.
5.3 Discussion of research question 3

The speed comparison in this report was a lot more unpredictable than initially expected. The authors expected Java to be ahead in most if not all cases. However, this was not true and Nodejs really performed beyond the initial expectations. With its fast file reading, mathematical operations and solid speed for sorting collections, it showed that it is a good choice for large scale applications. The reason for running these algorithms in Nodejs instead of in a browser is that the authors believe that most of these tasks are typical backend activities, if someone is using Javascript for building servers then it is very likely that he/she is using Nodejs. It is important to note that this may affect the performance of the code the javascript performance may also change depending on which browser it is running in.

Java performed very well in the case of sorting collections and outperformed the other two languages early on and as the data entries increased. When it came to the mathematical operations of fizzbuzz it was initially slower than the Javascript application. However, once more iterations were introduced to the Java program it started to have faster execution time, which could point to the fact that in terms of scalability Java pulls ahead. Javas file writing speed was not as fast as Javascript and not as slow as Python, although, Java have several classes that reads files in different ways so it is inconclusive weather the approach the authors took in this report was the absolute best way.

Python was behind in every test and the speed difference increased as bigger data collections and more mathematical operations were introduced. Pythons file reading was also significantly slower than the other two languages with it being around eight times slower compared to Javascript. It is important to note that speed is not necessarily the focus for Python and that it trades efficiency for simplicity in order to create and implement applications faster in the cases where speed does not matter.

There are many factors which could have affected the result of these tests. Firstly the code in the three languages may not be structured exactly the same and might not be completely optimized. Many of these tasks can be implemented in many ways, for example Java has more than one class that can be used for reading from files, but the authors have done as much as possible to try and make them similar. There are also factors which may affect the performance of the computer that the tests were done on, such as processes running in the background and hardware resources being unequally assigned. However, the authors turned off all programs that was running and started the tests from the command prompt but there is always things happening in the background which can not be controlled.
6. Future work

The area covered in this report is very broad and have many factors that can be addressed. The authors had to set several limitations in order to structure the work, make it focused and readable. An interesting finding was that Python was dominant in machine learning, while being the slowest of the languages. It would be compelling to find out why Python was chosen as the machine learning language of choice. The presented result shows that it may be because of its wide variety of libraries and tools for machine learning. However, if these tools were available in languages with better performance would Python still be the preferred choice?

More in depth testing can also be performed, it would be good to test more algorithms but also to look at more parameters than just speed, like for example memory and cpu usage. Javascripts testing would also be appropriate to test in different environments, like for example to test it in different browsers and see how much of a difference there is between server and client side Javascript.

7. Conclusion

Comparing languages can be done in many ways, the way we chose to compare languages was from syntax, use case and speed perspective. We found that trying to objectively say which language has the most readable and simple syntax is extremely difficult and instead we gathered as much information as possible to give the reader a comparative analysis, upon which one can base their own perception of which language has the easiest syntax. However there are some things that can be extracted from the first research question. If simplicity and ease of use is something that is favourable to one person, then Python or Javascript is probably the languages they will favour. Someone who likes more structure and abiding by rules and standards will probably feel that Java is more suited for them. The second research question focused on which field one can apply the knowledge that has been gathered in the first research question. We found that it is important to distinguish between what a language can be used for, and shall be used for. The languages was very flexible and could be used for many different applications. We based our results on what people prefer to use the languages for and which libraries, tools and frameworks it supports. The result was then presented in a venn diagram (see figure 3). The third research question was about the speed of the languages where we tried different tasks for all three languages, namely quicksort, fizzbuzz and read from file. One of the key findings that was made was that Python was the slowest in all three cases and the other two languages varied in performance. After having read about these three areas within the languages one has gathered knowledge that is valuable for any developer. This thesis presents a way to quickly get knowledge and understanding of three popular programming languages.
8. References


[9] NE.se [Internet]. Malmö: Nationalencyklopedin; unknown date. Python. [Cited date: 2019-03-31]. Available from:


[12] NE.se [Internet]. Malmö: Nationalencyklopedin; unknown date. Java. [Cited date: 2019-03-31]. Available from: https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/java-(programspr%C3%A5k)


[33] Oracle. The while and do-while Statements[Internet] Redwood City: Oracle; [Updated not available; Cited 2019-03-31]. Available from: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html


[70] Oracle. Class BufferedReader[Internet] Redwood City: Oracle; [Updated not available; Cited 2019-03-31]. Available from:


[110] Kivy. Kivy - Open source Python library for rapid development of applications that make use of innovative user interfaces, such as multi-touch apps [Internet]. publisher not available; [Update not available; Cited 2019-05-06]. Available from: https://kivy.org/#home

9. Appendix

9.1 A Syntax

Figure A1. different variables in Python

```python
string = "string"
number = 20
b = 10
number2 = 1.23
character = 'j'
number2 = "5.21"
```

Figure A2. Variables in Javascript

```javascript
var string = "string";
var number = 20;
var b = 10;
var number2 = 1.23;
var character = 'j';
```

Figure A3. Code showing some different datatypes in java.

```java
int number = 20;
String numberString = String.valueOf(number);
System.out.println(numberString);
```

Figure A4. Code to make a number into a string in java

```
#This line will be read as a comment
```

Figure A5. Python comments

```
"""This is a way to simulate a block comment"
```

Figure A6. Code showing how to simulate a block comment

```
//this line will be read as a comment
/*these
rows too */
```

Figure A7. Java comments

```
//this line will be read as a comment
/*these
rows too */
```

Figure A8. Javascript comments
Figure A9. Code showing how to use the line joining function [111].

```
if 1900 < year < 2100 and 1 <= month <= 12 
    and 1 <= day <= 31 and 0 <= hour < 24 
    and 0 <= minute < 60 and 0 <= second < 60:  # Looks like a valid date
    return 1
```

Figure A10. Java semicolon separator

```
System.out.println("Hello world");
```

Figure A11. Javascript separators

```
console.log("Hello world");
```

Figure A12. Example of indentation in Python

```
if (i < 2):
    print("in if statement")
print("outside if statement")
```

Figure A13. Java showcasing when curly brackets are not needed

```
if (x == 0):
    System.out.println("in if statement");
    System.out.println("not in if statement");
```

Figure A14. Using curly brackets in Javascript

```
if (x == 0):
    console.log("Is if statement");
```

Figure A15. Code showing a sequence of if, elif and else statements.

```
number1 = 2
if number1 == 1:
    print("Its 1")  #This will be executed
elif number1 == 2:
    print("Its 2")
else:
    print("nothing is true")
```

Figure A16. Basic switch statement in Javascript

```
var animal = "dog";
switch(animal){
    case "cat":
        console.log("The animal is a cat");
        break;
    case "dog":
        console.log("The animal is a dog");
        break;
    default:
        console.log("Animal is not familiar");
}
```
Figure A17. Basic switch statement in Java

```java
public class Main {
    public static void main(String[] args) {
        String animal = "dog";
        switch(animal){
            case "cat":
                System.out.println("The animal is a cat");
                break;
            case "dog":
                System.out.println("The animal is a dog");
                break;
            default:
                System.out.println("Animal is not familiar");
        }
    }
}
```

Figure A18. For loop in Java

```java
for (int i = 0; i < 10; i++){
    System.out.println(i);
}
```

Figure A19. For loop in Javascript

```javascript
for(var i = 0; i < 10; i++){
    console.log(i);
}
```

Figure A20. Code showing the use of the range function in a for loop in Python.

```python
string[] foodArray = ["Rice", "Tomatoes", "Pasta", "Pizza"];  
for (String food : foodArray){
    System.out.println(food);
}
```

Figure A21. Enhanced for loop in Java and also a basic array

```java
var numbers = [1, 2, 3, 4];
```

Figure A22. Array in Javascript

```javascript
var numbers = arr.array('i', [1, 2, 3, 4]);
```

Figure A23. Array in Python

```java
ArrayList<String> names = new ArrayList<>();
names.add("Tobias");
names.add("Rasmus");
names.remove(0); "Tobias";
```

Figure A24. Array list in Java
Figure A25. list in Java

```java
list list = new ArrayList();
list.add("String");
list.add("char");
list.add(123);
```

Figure A26. Set operations in Python

```python
Set<String> toySet = new HashSet<>();
toySet.add("Ball");
// this entry will be removed because
// there already exists a ball in the set
toySet.add("Ball");
toySet.add("Train");
```

Figure A27. A basic set in Java

```java
const toys = new Set(['ball', 'plane', 'ball', 'train']);
console.log(toys);
```

Figure A28. Implementation of a set in Javascript

```javascript
const t = 100, 123, 'apple' // create a tuple
print(t[1]) // print the element at index 1 in the tuple
print(t) // print the entire tuple
u = t, (3,2,1) // nest tuples
print(u[1]) // print the second tuple
```

Figure A29. Code showing tuplet operations in Python

```python
123
(100, 123, 'apple')
(3, 2, 1)
```

Figure A30. Code showing the output from the code in figure A29

```java
Stack<String> myStack = new Stack<>();
myStack.push (rem: "One");
myStack.push (rem: "Two");
myStack.pop ();
```

Figure A31. Create and use a stack in Java

```java
Queue<String> myQueue = new LinkedList<>();
myQueue.add("One");
myQueue.add("Two");
myQueue.remove ();
```

Figure A32. Create and use a queue in java
Figure A33. Code showing how to create and use a stack in Python

```python
stack = []
stack.append(1)
stack.append(2)
stack.pop()
```

Figure A34. Queues in Python

```python
import queue
queue = queue.Queue()
queue.put(1)
queue.put(2)
queue.get()
```

Figure A35. How to use a stack in Javascript

```javascript
var stack = [];
stack.push("One");
stack.push("Two");
stack.pop();
```

Figure A36. Shows how to use a queue in Javascript

```javascript
var queue = [];
queue.push("One");
queue.push("Two");
queue.shift();
```

Figure A37. Code showing how to create a dictionary and how to access an element in Python

```python
telephoneNumbers = {'Tobias': 12374742, 'Rasmus': 14352463}
print(telephoneNumbers['Tobias'])
print(personInformation['Age'])
```

Figure A38. Code showing how to create a hashmap in Java and how to use it

```java
Map<String, Integer> telephoneNumbers = new HashMap<>();
technicalNumber.put("Tobias", 12374742);
technicalNumber.put("Rasmus", 14352463);
System.out.println(telephoneNumbers.get("Tobias"));
```

Figure A39. Code showing how to create and use a dictionary in Javascript

```javascript
var telephoneNumber = {"Tobias": 12374742, "Rasmus": 14352463};
console.log(telephoneNumber["Tobias"])`n```

Figure A40. Javascript code showing the possibility to create a dog object without having a name as a parameter

```javascript
const Dog = function(name){
  this.name = name;
}

Dog.prototype.bark = function(){
  console.log(this.name + " said woof")
}

// No syntax error even though the constructor wants a name when the object is made
const dog = new Dog();
```

Figure A41. Importing and using a class in javascript

```javascript
const User = require('./user.js');
let users = new User("rasmus");
var name = users.sayHi();
```

Figure A42. exporting the javascript class

```javascript
class User {
  constructor(name) {
    this.name = name;
  }
  sayHi() {
    console.log(this.name);
  }
}
module.exports = User;
```
Car car = new Car();
System.out.println("Car tires "+ car.getTires());

Figure A43. Using a class in Java

import Animal
animal = Animal.Dog("Rasmus")
animal.bark()

Figure A44. Importing and using a class in Python

public String hello(String name) {
    return "hello "+ name;
}

Figure A45. Calling a method in Java

def hello(name):
    return "Hello "+ name

print(hello("Rasmus"))

Figure A46. Calling a method in Python

custom.log(hello("Rasmus"));

function hello(name) {
    return "Hello "+ name;
}

Figure A47. Calling a method in Javascript

try:
    x = 10 + "123"
except TypeError:
    print("Sorry, there is an error in the code")

Figure A48. Code showing the syntax of the try - except feature in Python

except (TypeError, NotImplementedError):
    print("Sorry, there is an error in the code")
extype ZeroDivisionError:
    print("you can not divide by zero")

Figure A49. Code showing how to handle more than one exception in the same clause

try {
    System.out.println(1/0);
} catch (ArithmeticException ae) {
    System.out.println("You can not divide by zero");
}

Figure A50. Using try - catch in Java
Figure A51. Using try - catch in Javascript (trying to access an undefined variable x)

```javascript
try {
    console.log(x);
} catch (e) {
    console.log("Something went wrong");
}
```

Figure A52. writing and reading a file in Javascript

```javascript
var fs = require("fs");
var information = "This is a text";
fs.writeFile("filename.txt", information, function(err){
    if (err) {
    }
});

fs.readFile("filename.txt", function(err, buf) {
    console.log(buf.toString());
});
```

Figure A53. writing and reading a file in Java

```java
String fileName = "filename.txt";
try {
    FileWriter fileWriter = new FileWriter(fileName);
    fileWriter.write("This is a text");
    fileWriter.close();
} catch (IOException e) {
    System.out.println("Something went wrong while writing to file");
}

String fileText;
try {
    BufferedReader BufferedReader = new BufferedReader(new FileReader(fileName));
    while ((fileText = BufferedReader.readLine()) != null) {
        System.out.println(fileText);
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Figure A54. writing and reading a file in Python

```python
#Write to file
f = open("filename","w")
f.writelines("This is a text")

#Read from file
f = open(filename, "r")
print(f.read())
```
9.2 B Benchmarking algorithms

```python
from timeit import default_timer as time

def quickSort(arr):
    less = []
    pivotList = []
    more = []
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[0]
        for i in arr:
            if i < pivot:
                less.append(i)
            elif i > pivot:
                more.append(i)
            else:
                pivotList.append(i)
        less = quickSort(less)
        more = quickSort(more)
        return less + pivotList + more

for x in range(150):
    f = open("Dataset1000000.txt", "r")  # This file was changed when choosing another dataset
    numbers = []
    for line in f:
        numbers.append(line)
    start = time()
    quickSort(numbers)
    stop = time()
    print((stop - start) * 1000)
```

Figure B1. The Python quicksort algorithm used during the tests [7]
function sort(array, less) {
    function swap(i, j) {
        var t = array[i];
        array[i] = array[j];
        array[j] = t;
    }

    function quicksort(left, right) {
        if (left < right) {
            var pivot = array[Math.floor((right - left) / 2)],
                left_new = left,
                right_new = right;

            do {
                while (less(array[left_new], pivot)) {
                    left_new += 1;
                } while (less(pivot, array[right_new])) {
                    right_new -= 1;
                }

                if (left_new <= right_new) {
                    swap(left_new, right_new);
                    left_new += 1;
                    right_new -= 1;
                }

            } while (left_new <= right_new);

            quicksort(left, right_new);
            quicksort(left_new, right);
        }

        quicksort(0, array.length - 1);

        return array;
    }

    var fs = require('fs');
    for (i = 0; i < 100; i++) {
        var array = fs.readFileSync('Dataset1000000.txt').toString().split('n'); //This file was changed when choosing another dataset

        console.time('timer1');
        sort(array, function(a, b) { return a < b });
        console.timeEnd('timer1');
        array = [];
    }
}

public static void main(String[] args) throws IOException {
    String line = "";
    long start = 0;
    long stop = 0;

    for (int i = 0; i < 150; i++) {
        List<Integer> numbers = new ArrayList<>();
        FileReader FileReader = new FileReader(); //This file was changed when choosing another dataset
        BufferedReader bufferedReader = new BufferedReader(fileReader);

        while ((line = bufferedReader.readLine()) != null) {
            numbers.add(Integer.parseInt(line));
        }

        start = System.currentTimeMillis();
        quickSort(numbers);
        stop = System.currentTimeMillis();
        System.out.println(stop - start);
        numbers.clear();
    }
}

Figure B2. The Javascript quicksort algorithm used during the tests [7]

Figure B3. The code that was used when calling the quicksort method in Java
Figure B4. The actual quicksort algorithm in Java [7]

```java
public static <E extends Comparable<? super E>> List<E> quickSort(List<E> arr)
{
    if (arr.isEmpty())
        return arr;
    else
    {
        E pivot = arr.get(0);
        List<E> less = new LinkedList<E>();
        List<E> pivotList = new LinkedList<E>();
        List<E> more = new LinkedList<E>();

        // Partition
        for (E i : arr) {
            if (i.compareTo(pivot) < 0)
                less.add(i);
            else if (i.compareTo(pivot) > 0)
                more.add(i);
            else
                pivotList.add(i);
        }

        // Recursively sort publics
        less = quickSort(less);
        more = quickSort(more);

        // Concatenate results
        less.addAll(pivotList);
        less.addAll(more);
        return less;
    }
}
```

Figure B5. The fizzbuzz algorithm in Python that was used during the tests [8]

```python
from timeit import default_timer as time
start = 0
stop = 0

def fizzbuzz():
    for i in range(0, 1000001): #The amount of times the loop was running was changed between the different tests
        if i % 15 == 0:
            pass
        elif i % 3 == 0:
            pass
        elif i % 5 == 0:
            pass
        else:
            pass

start = time()
fizzbuzz()
stop = time()
print((stop - start) * 1000)
```
Figure B6. The fizzbuzz algorithm in Javascript that was used during the tests [8]

```javascript
function fizzbuzz() {
  var i;
  for (i = 0; i <= 1000000; i++) {
    // the amount of times the loop was running was changed between the different tests
    if (!(i % 3)) {
      //
    }
    if (!(i % 5)) {
      //
    }
  }
}

console.time('timer');
fizzbuzz();
console.timeEnd('timer');
```

Figure B7. The fizzbuzz algorithm in Java that was used during the tests [8]

```java
public class Fizzbuzz100 {

    public static void main(String[] args) {
        Fizzbuzz100 fizzbuzz100 = new Fizzbuzz100();
        long start = 0;
        long stop = 0;
        start = System.currentTimeMillis();
        fizzbuzz100.fizzbuzz();
        stop = System.currentTimeMillis();
        System.out.println(stop - start);
    }

    private void fizzbuzz() {
        for (int i = 0; i <= 1000000; i++) {
            // the amount of times the loop was running was changed between the different tests
            if (i % 3 == 0 && i % 5 == 0) {
                //
            } else if (i % 3 == 0) {
                //
            } else if (i % 5 == 0) {
                //
            } else {
                //
            }
        }
    }
}
```

Figure B8. The reading from file code in Python that was used during the tests

```python
from timeit import default_timer as time
start = 0
stop = 0

def readFile(path):
    with open("text.txt") as f:
        content = f.read()

for j in range(0, 101):
    start = time()
    readFile("text.txt")
    stop = time()
    print(round((stop - start) * 1000)) # rounds the result to the nearest millisecond
```
Figure B9. The reading from file code in Javascript that was used during the tests

```javascript
var fs = require('fs');

function readFile(path) {
    content = fs.readFileSync(path);
}

for (i = 0; i <= 100; i++) {
    console.time('timert');
    readFile('text.txt');
    console.timeEnd('timert');
}
```

Figure B10. The reading from file code in Java that was used during the tests

```java
public class ReadWrite {

    public static void main(String[] args) throws IOException {
        ReadWrite readWrite = new ReadWrite();
        long start = 0;
        long stop = 0;
        for (int i = 0; i <= 100; i++) {
            start = System.currentTimeMillis();
            readWrite.readFile(path = "text.txt");
            stop = System.currentTimeMillis();
            System.out.println(stop - start);
        }
    }

    private void readFile(String path) throws IOException {
        String content = "";
        FileReader fr = new FileReader(path);
        BufferedReader br = new BufferedReader(fr);
        String line = "";
        while ((line = br.readLine()) != null) {
            content = line;
        }
    }

}