Towards a Structured Mechanism for Comparing Programming Languages, Regarding Execution Complexity and Syntactic Similarities

A dissertation submitted in partial fulfilment of the requirements for the degree of Bachelor of Science (Honours) in Computing

By: Vidhuran Sivabalan

April 2017
DECLARATION

I hereby declare that this dissertation entitled ‘Towards a Structured Mechanism for Comparing Programming Languages, Regarding Execution Complexity and Syntactic Similarities’ is entirely my own work, and it has never been submitted nor is it currently being submitted for any other degree.

Candidate: Vidhuran Sivabalan
Signature:
Date: 24/04/2017

Supervisor: Dr Ana Calderon
Signature:
Date:
ABSTRACT

Java is currently one of the most popular languages used for programming; it has been developed over decades and is used widely within the computing industry. However, other programming languages such as Python and C# are developing and advancing at a rapid rate and are currently on route to surpass the popularity of Java in the near future. Making a comparison between some of the three most popular programming languages C#, Java and Python will help to analyse the reasons for their popularity and increased use. Python is currently taught to higher education students and is being adopted more prevalently for use in certain job sectors. For the past decade, the most used programming language has been Java. In addition, C# was also relatively more popular than Python in the past, however, Python has surpassed C# in recent years (TIOBE, 2017). It is essential to realise all programming languages have their unique properties. This dissertation intends to clarify the reasons for, and justify the shift in popularity and job trends. In general, varying job sectors are increasingly requiring the use of more programming languages and therefore it is imperative to focus on the performance and usability of the languages in order to garner a greater understanding. By optimising the functionality of programming languages through decreasing code density, the efficiency and computation power of the languages can be enhanced, creating the perfect programming language. Community driven languages have a greater rate of development, thus increasing the proficiency of applications (Oracle, 2017). The outcomes in this study will hopefully distinguish the differences between all three language by testing their execution complexities. The results from this study will either contradict or correspond with the current job trends which represent the programming languages that are in demand. Finally, it is important for higher education to teach computing students with the appropriate programming language that is most sort after to give them the best opportunities.

ACKNOWLEDGEMENTS

I would like to thank my project supervisor, Dr Ana Calderon, for her continuous support throughout the year. I would also like to thank my peers and lecturers in the Cardiff School of Management for their support and for enhancing my personal and professional development.
# Contents

1. INTRODUCTION
   1.1 Introduction 1
   1.2 Rationale for the study
      1.2.1 Problem it will address 2
   1.3 Objectives 2
   1.4 Hypothesis 2

2. LITERATURE REVIEW
   2.1 Introduction 3
   2.2 History of Programming
      2.2.1 Early Programming Languages
         2.2.1.1 Introduction 3
         2.2.1.2 Assembly Language 4
         2.2.1.3 FORTRAN 4
         2.2.1.4 COBOL 60 4
         2.2.1.5 ASCII 4
         2.2.1.6 Conclusion 4
      2.2.2 General Purpose languages
         2.2.2.1 Pascal 5
         2.2.2.2 C 5
         2.2.2.3 C++ 5
         2.2.2.4 MATLAB 6
         2.2.2.5 Perl 6
         2.2.2.6 Conclusion 6
      2.3 Pioneering Programming Languages
         2.3.1 Java 7
         2.3.2 C# 8
         2.3.3 Python 9
      2.4 Use of Programming Languages within Professional Environments
         2.4.1 Bioinformatics 10
         2.4.2 Economics 11
         2.4.3 Higher Education 12
      2.5 Conclusion 13

3. METHODOLOGY
   3.1 Introduction 14
      3.1.1 Integrated Development Environment (IDE) 14
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.3 Usability</td>
<td>53</td>
</tr>
<tr>
<td>4.2.3.1 Uses</td>
<td>53</td>
</tr>
<tr>
<td>4.2.3.2 Syntax</td>
<td>53</td>
</tr>
<tr>
<td>4.2.3.3 Runtime</td>
<td>54</td>
</tr>
<tr>
<td>4.3 Conclusion</td>
<td>54</td>
</tr>
<tr>
<td>5. Conclusion and Limitations</td>
<td>55</td>
</tr>
<tr>
<td>5.1 Conclusion</td>
<td>55</td>
</tr>
<tr>
<td>5.2 Limitations</td>
<td>56</td>
</tr>
<tr>
<td>6. REFERENCES</td>
<td>57</td>
</tr>
<tr>
<td>8. APPENDIX</td>
<td>60</td>
</tr>
<tr>
<td>Appendix A</td>
<td>60</td>
</tr>
<tr>
<td>Appendix B</td>
<td>62</td>
</tr>
<tr>
<td>Appendix C</td>
<td>74</td>
</tr>
<tr>
<td>Appendix D</td>
<td>86</td>
</tr>
</tbody>
</table>
List of Figures
Graphical Abstract TIOBE Programming Community Index .................................. iii
Figure 1.1. Indeed Job Trends Graph ........................................................................ 1
Figure 2.1. TIOBE Index Popularity of Programming Languages 1987 - 2017 .......... 6
Figure 2.2. TIOBE Index for Java ........................................................................... 7
Figure 2.3. TIOBE Index for C# ............................................................................. 8
Figure 2.4. TIOBE Index for Python ....................................................................... 9
Figure 2.5. Blast Parsing Programming ................................................................ 10
Figure 2.6. TIOBE Index for C++ .......................................................................... 11
Figure 2.7. Entry Level Programming Languages .................................................. 12
Figure 2.8. Programming Languages for Top US Science Departments ..................... 12
Figure 2.9. TIOBE Programming Community Index .............................................. 13
Figure 3.1. to Figure 3.82. Screenshots of Test Implementation ............................... 15 - 46
Figure 4.1. Execution Time for Hello World [1] & [2] Tests ....................................... 50
Figure 4.2. Execution Time for For Loop [3] & [4] Tests ......................................... 50
Figure 4.3. Execution Time for Sum of Integers [5] & [6] Tests .................................. 51
Figure 4.4. Execution Time for Sorting Sequence [7] & [8] Tests ............................... 51
Figure 4.5. Lines of Code Needed to Complete all Tests ......................................... 53
Figure 5.1. Most Popular Desktop IDEs and Code Editors ....................................... 56

List of Tables
Table 3.1. .................................................................................................................. 21
Table 4.1. ................................................................................................................... 49
Table 4.2. ................................................................................................................... 49
Table 4.3. ................................................................................................................... 49

List of Acronyms and Abbreviations

ANSI: American National Standards Institute
ASCII: American Standard Code for Information Interchange
BCPL: Basic Combined Programming Language
COBOL: Common Business Oriented Language
EDSAC: Electronic delay storage automatic calculator
GUI: Graphical User Interface
IDE: Integrated Development Environment
LOC: Lines of Code
MVS: Microsoft Visual Studio
PTVS: Python Tool for Visual Studio
1. INTRODUCTION

1.1 Introduction
In this dissertation, comparisons of different high level programming languages will be made. C#, Python and Java are some of the most popular programming languages used to create applications, websites and software. Cyber security, Web design and Data analyst jobs require programming in different languages to communicate with one another. This study will consider the similarities and differences between the main languages taught to entry level students; in addition, it will also consider the reasons for the use of a particular programming language for specific sectors of work (Alkadi et al., 2011). However, the main focus of the study is to discover why in recent years there has been a shift in trends regarding job roles related to C#, Java and Python programming.

1.2 Rationale for the study
There are several reasons for conducting this study. Firstly, it will determine which programming language of the three will offer the greatest practicality? Secondly, it will enable us to discover the purpose of the languages even though the outputs in most cases are the same? Finally, the study will also help discover which of the languages will have the most demand in the future? The interest for this particular study was first triggered due to a graph displayed on Indeed. Indeed is a recruitment based website that advertises job opportunities to all career sectors on behalf of business organisations. It also provides statistical data analysis and comparison of roles within particular sectors of work. In the figure below it is clear to see the shift in job trends within recent years regarding Java, C# and Python programming roles (Indeed, 2017).

![Job Trends Graph](image)

Figure 1.1. (Indeed, 2017). The Indeed job trends graph representing the percentage of job postings from 2012 to 2016 corresponding to the programming languages Java, Python and C#.
1.2.1 Problem it will address
The graph also clearly indicates that trends in computer science related jobs are highly linked and filtered by the chosen programming language. Discovering the reasons why there has been a shift in job trends will help forecast and envision the direction of growth for programming languages and their development (Indeed, 2017).

This dissertation will hopefully contribute towards innovation within the computer science community, especially for higher education students, by highlighting the most beneficial and usable languages. Acquiring knowledge regarding the differences between the languages may aid students when determining which language to study or deciding which language will increase future career prospects.

1.3 Objectives
The purpose of this dissertation is summarised into five objectives.

1. To develop a series of tests to examine the differences in C#, Java and Python programming languages.
2. To conduct a literature review to explore prior research findings regarding the development and progression of programming languages.
3. To conduct a thorough analysis of the results obtained from this study.
4. To evaluate the performance and usability aspect of the programming languages.
5. To summarise findings and compare the results from this study to the Indeed Job Trends graph and TIOBE Index.

1.4 Hypothesis

- **Hypothesis**

  “In this study, the most popular language Java will have the best performance and usability compared to Python and C#, in addition, Python will outperform C#.”

- **Null Hypothesis**

  “In this study, there will be no difference between the programming language used and the performance and usability of the language.”
2. LITERATURE REVIEW

2.1 Introduction
Recent studies can be utilised to understand the underlying complexities between programming languages. There is much discussion on the differences between programming languages, with various benefits and weaknesses. Using relevant research and results from the tests conducted during this study, a viable comparison of the languages will be achievable. Furthermore, debating the current trends in programming languages will help establish why some programming languages are advantageous over others. To understand the history of modern programming languages, a thorough look at the history of programming alone will enable a greater understanding about how the modern languages came to be. Programming languages are a set of instructions used in computer systems to create computer programs, often referred to as a high-level language. The purpose of a high-level language is to communicate and interpret instructions to the hardware. It is predicted that there are currently over 256 languages (DZone, 2013). Modern languages have inherited legacy features from existing languages, as well as, its ancestors. This has helped develop the performance and usability of computing languages. In programming, it is often beneficial to program in different languages, depending on the purpose of the program (McCarthy et al., 1963); this explains the need for multiple languages as each one has its unique benefits.

2.2 History of Programming

2.2.1 Early Programming Languages

2.2.1.1 Introduction
In 1941, one of the earliest digital computers, the Z3, operated with the Plan Calculus language; this was the creation of German civil engineer Konrad Zuse (German-way, 2017). In 1948, at the University of Manchester the Small-Scale Experiment Machine, better known as the “baby” was developed. One of the developers, Tom Kilburn, wrote a program which consisted of seventeen instructions to run on the electronic random access memory computer. In the 1950’s, the mathematician Grace Hopper created a program like that of a present-day compiler. The program called A-0, had the purpose of minimising the use of numbers in instructions and replacing them with English words. This program made it easier to develop instruction sets, and was widely used in commercial computer systems. Another important creation in the early 50’s was the OXO game created for the EDSAC (Electronic delay storage automatic calculator) computer at the University of Cambridge. The EDSAC computer was designed in 1947 and hosted a game, which today resembles the popular tic-tac-toe game (Tnmoc.org, 2017). This brief history about programming, summarises the development and exploration surrounding the first basic computing languages that helped shape today’s use of the many programming languages needed for modern technology (Computerhistory.org, 2017).
2.2.1.2 Assembly Language
Assembly language is a type low-level programming language, a form of mnemonic code. The language is utilised to correspond directly to the processor. During the 1950’s, it provided the solution to many problems, although, programmers found its methods inconvenient; as the language was time-consuming and the usability was poor. At the time, it provided the advantage of being portable across multiple platforms which made it efficient compared to other methods of low-level programming (Wang, 2012).

2.2.1.3 FORTRAN
The year 1957 hosted a big step forward in computing when IBM developed FORTRAN, John Backus, who was the man responsible for Speedcode, led the team in IBM. The project was founded upon improving the usability of assembly language. As stated previously, assembly language was problematic but was useful because of its efficiency. IBM aimed to manufacture a high-level language that was straightforward and performance-driven, in addition to maintaining the efficiency of assembly language. The success of FORTRAN showed that programmers can write programs faster with greater understanding. Since then, IBM has developed FORTRAN 66, 77, 90, 95, 2003, 2008 and 2015 (Metcalf et al., 2011).

2.2.1.4 COBOL 60
At the start of the 1960’s, COBOL (Common Business Oriented Language) was created by computer manufactures and the Department of Defence. COBOL 60 was developed to contain a common language for business use, as well as, the intention to stop the rising costs of programming. FORTRAN being a very popular programming language at the time, did not have the capabilities needed for business use. Businesses adopted COBOL effortlessly, with the main feature being its English-like syntax. COBOL is still a popular language widely used in various business sectors and is run on multiple systems, including Windows and Unix (Paulson, 2001).

2.2.1.5 ASCII
In 1963, ASCII (American Standard Code for Information Interchange) was created and published by the ASA (American Standard Association). The ASCII language permits the creation of sequences that can be interpreted into letters of the alphabet. This formed a uniform standard, allowing the transfer of data between different manufactured machines. However, ASCII was restricted and could only represent up to 256 symbols, a similar uniform standard – Unicode, permitted over 100,000 symbols (Metcalf et al., 2011).

2.2.1.6 Conclusion
The creation of these languages increased innovation and enabled the development of the languages to progress at a much faster pace. Where one language lacked to cater to a specific market, another language utilised, to develop and improve depending on existing requirements. For example, FORTRAN was not suitable as a business standard language, but could cater to the needs of other users; whereas COBOL 60 was intentionally created to serve the needs of business users.
2.2.2 General Purpose languages

2.2.2.1 Pascal
Pascal is a programming language that was created in 1970 by Professor Niklaus Wirth. In the past Wirth also aided in the development of ALGOL 60. Pascal became a mainstream programming language and was widely used in the commercial industry. The program was named after Blaise Pascal, a Physicist and mathematician; he aimed to develop a language that could be used in different commercial sectors and for scientific purposes. The language became a standard that was then used in educational settings to instruct upcoming computer scientists, who have the potential to then proceed and apply the language in different sectors (Wirth, 1975).

2.2.2.2 C
Dennis Ritchie lead a team to create the programming language ‘C’ between 1969 and 1973. C is used as a structure oriented language that is recognised for its wide-ranging capability; C was heavily based on a programming language called BCPL (Basic Combined Programming Language). It was initially invented for the purposes of reinventing the Unix system but it is now also a major part of the Windows and Linux operating systems. Unix became compatible with other systems making it a popular language that was adopted rapidly (MacDonald, T., 1992). Like Pascal, C, is a structure oriented language with the focus on the processes and procedures that operate upon the data. C is often referred to as a middle language because it does not offer all the functions found in high level languages but delivers the foundation needed to produce the outcomes wanted. C can be used for operating system development, compilers, database systems and network drives. Currently, C is available in two standardised editions, C89/C90 and C99; both developed by the American National Standards Institute (ANSI) (Singh, 2015).

2.2.2.3 C++
C++ is an extension of the C language with the addition of classes, the language is a strong general purpose language giving the portability of programming in both C and object-oriented styles. C++ consists of both high and low level features; therefore, it is regarded as an intermediate language. The creator desired that the language would produce more complex programs, as well as, giving a gratifying experience to programmers. Currently, C++ is used within system design and development software and is recognised as one of the more popular programming languages. The features of the language include its ability to initiate data types several times, its consists of logical, comparison and arithmetic operators; with a mechanism to overload them within programs. The main feature of C++ is class, class can be described as a user-defined type. The programmer can separate an application into workable sections by creating new types that correspond with the theory of the application. This enables the user-defined types to be applied to a good degree of methodology making it easier to sustain programs. C++ was an extension of C and its libraries; it also retained its efficiency when dealing with the fundamental objects of the hardware (Stroustrup, 2000).
2.2.2.4 MATLAB
MATLAB which stands for Matrix Laboratory, is a scientific and mathematic programming language. This high-level programming language was created by MathWorks and chief scientist Professor Cleve Moler; it aimed to assist students who use mathematical software libraries without prior knowledge regarding FORTRAN. MATLAB scripturally resembles C++. It was intended specifically for research purposes due to its ability to handle data analysis. MATLAB offers solutions for easy creation of scientific and engineering graphics, as well as, for application development. MATLAB further advanced establishing use within the aerospace and automobile industries, as well as, in computational science. It has the necessary tools to provide solutions to complex systems of equations which aided industries with modelling, simulation and prototyping (Noble, 2012).

2.2.2.5 Perl
Perl is a cross platform general purpose programming language, it was released in 1987 with the purpose of extracting information and text manipulation within systems. These general purpose programming languages were used to enhance and develop different industries; it became apparent that certain languages were well suited to specific sectors and were therefore developed purposefully for those sectors. These languages formed the foundation needed for pioneering languages such as Java, C# and Python. Perl as a language has certain disadvantages, for example, usability is difficult with regards to written code. However, Perl provides many resources such as CPAN and Bioperl which increase the efficiency of the language. It also has rapid text processing which is extremely useful for those who require a programming language for simple business use (Schwartz et al., 2017).

2.2.2.6 Conclusion
General purpose programming languages have a pivotal role in our lives; supporting our infrastructure, as well as, developing the tools to increase the accessibility of our communications around the world. Due to numerous general purpose languages, there has been a vast rate of development in the past two decades, leading to an abundance of programming languages. Although, the most sophisticated languages, Java, C, C++ and C# have prospered from previously popular languages expanding upon the features that they lack.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>C++</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>C#</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Python</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>10</td>
<td>27</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PHP</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>JavaScript</td>
<td>7</td>
<td>9</td>
<td>8</td>
<td>3</td>
<td>20</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Visual Basic .NET</td>
<td>8</td>
<td>22</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Perl</td>
<td>9</td>
<td>8</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>15</td>
<td>-</td>
</tr>
<tr>
<td>Assembly language</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>COBOL</td>
<td>24</td>
<td>33</td>
<td>15</td>
<td>5</td>
<td>3</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>Fortran</td>
<td>27</td>
<td>24</td>
<td>20</td>
<td>22</td>
<td>15</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 2.1. (TIOBE, 2017). TIOBE index – ‘Very Long Term History’ represents the past rankings of the current most popular programming languages.
2.3 Pioneering Programming Languages

2.3.1 Java
Sun Microsystems released Java as a programming language in 1995. Many applications and websites currently still rely heavily on Java as a language. Java is known for its safe and secure use; it is a very reliable language that is still to-date the most popular programming language according to Tiobe Index. Java was developed from its predecessor Oak programming, first created by James Gosling. It is widely used in many computing sectors, including for gaming, datacentres, mobile phones and the internet. Java is a dynamic and manageable language, the ease of maintenance when using Java is an attractive feature. This is possible due to code being organised into classes, allowing a Just In Time compiler to convert source code into assembly code to decide at any time when to run the class. This is useful as the whole program does not need to be compiled, optimising the programs functionality and performance (Taboada et al., 2013).

Java is an object-oriented and platform independent language; this enables Java to be compiled into bytecode for the internet instead of into platform specific machines; distinguishing it from languages like C and C++. Java is a relatively simple language to learn as the design is easy to understand, especially if an individual has prior knowledge of object-oriented programming. An advantage of using Java is that virus free systems can be developed as it has the tools required to produce secure encrypted frameworks. Due to Java’s runtime system, it is also recognised as an architecture-neutral language, as the compiled code can be executed in various processors. Existing as an architecture-neutral language also enables Java to be portable and dynamic. Therefore, Java is designed to adapt to changing environments unlike languages like C and C++. Another advantage of using Java is its robustness, with multithreading abilities and checks for error during compile and runtime. Java is constantly improved and supported by the Java community, which combines a vast array of programming skills to keep Java relevant (Gosling and Gilton, 1995). According to figure 2.8 Java is the second most popular beginner language taught in US universities. The reason being is that Java is extremely versatile, the language operates on a write it once, run it anywhere idea. Due to Java’s popularity, its platform is integrated into many systems across the world. The average salary of a Java developer is around $96,000 a year, with Java having the most job listings compared to any other programming language (Gooroo, 2017).

![Figure 2.2. (TIOBE, 2017). TIOBE Index for Java, this graph shows the shifting popularity ratings of Java over the last 16 years.](image)
2.3.2 C#

Another programming language, C#, was released during the new millennium by Microsoft’s .NET enterprise. The lead designer Anders Hejlsberg wanted to develop an all-round, object orientated language that could work with the Common Language Infrastructure. Within six years of its initial release, C# emerged as a popular programming language with investors keen to expand and develop the language, making it a more stable and usable language (Microsoft, 2017). C# was a success; this consequently affected Java, another popular programming language. The use of C# spread substantially; this enabled the language to be used in dynamic areas of technology, such as computer gaming and within operating systems. C# is a good introduction language for early programmers as it will appear relatively familiar to those with prior knowledge regarding programming languages (Alkadi et al., 2011).

A negative aspect of C#, is its reliance upon the .NET framework to improve and run applications. However, the framework has vast libraries that enables C# access to APIs, cross network communications and database accessibility. As well as these features, multi-platform support, development tools and generics enable C# to be utilised in innovative software/ rapid development projects. An advantage of C# is that the strict nature of the language, it is a strong type language which avoids common programming errors that weak type languages are prone to. Another positive of C# is its efficient garbage collection feature which eliminates the inconveniences of memory leaks and automatically manages obscure object memory (Terry, 2004).

C# is a form C and C++, with the emphasis on being an object-oriented language; it also virtually resembles the syntactical properties of the two languages. This offers transferable skills when coding in similar languages like Java. In the United States, C# is taught in a limited number of Universities due to its specific proprietary platforms, although used widely in industry, it does not have the same level of cross platform outreach as Java. Nevertheless, the Tiobe Index lists C# as the 4th most popular language just under C++ and above Python. The current average salary for a C# programmer is around $85,000, which is around $11,000 less than Java (Gooroo, 2017). Indeed, also shows that C#’s job postings has been on a negative trend from as far back as 2012 (Indeed.com, 2017).

![Figure 2.3. (TIOBE, 2017). TIOBE Index for C#, this graph shows the shifting popularity ratings of C# over the last 16 years.](image-url)
2.3.3 Python

Python was first released in 1991; the creator Guido van Rossum, who is an employee of Dropbox, had strived to create a simple to understand programming language. The language has an open license, allowing programmers to experiment and experience coding efficiency; this brought out quicker turnarounds of programs. In its infancy stage, it attracted programmers to explore the use of simple syntax with the satisfaction of an unrestricted programming language. At one stage Java was the most popular language taught to entry level computer scientists at educational institutions, this changed when Python was widely adopted for web development, desktop applications, games, data analysis and scientific computing. The rapid progression of Python is shown in figure 2.8 which indicates Python has surpassed Java, becoming the more popular language to be taught to entry level computer science students (Mészárosová, 2015).

Programming has helped to expand knowledge within scientific communities; it highlights the need for a programming language that is easy to pick up and understand, whilst having all the necessary tools to cope with big data workloads. Python is a general purpose high level language, it is user friendly especially to beginners as the unrestricted syntax allows programmers to concentrate on the important programming concepts rather than language restrictions (Perkel, 2015). Unfortunately, the flexibility of the language makes the language hard to maintain especially if the program or application is on a larger scale. Although, errors are easier to avoid, for example, if there is a difficult error that has culminated in a very specific situation it is harder to resolve unless years of Python programming is behind. The freedom to manipulate code is advantageous as Python is a cross platform language capable of running on multiple operating systems, such as, Mac IOS, Linux and Microsoft. The issues with Python include lack of definition. systems would need to do the referencing which influences the performance of the language (Bird et al, 2009). On the TIOBE index Python is currently the 5th most popular programming language out of 100, just behind C# (TIOBE, 2017). There has been a gradual increase in the number of job postings on Indeed from 2012 till the end of 2016, overtaking C#'s job postings (Indeed, 2017). The average salary across the globe for jobs requiring Python programming knowledge, is around $99,000 (Gooroo, 2017). Python was used to create popular sites such as Google, Dropbox and Instagram, the demand for Python programmers are on the increase which explains why the top US Universities have decided to teach Python to entry level students. Every year Python releases updates and new versions; this proactively keeps the language relevant (Guo, 2014).

![TIOBE Index for Python](image)

*Figure 2.4. (TIOBE, 2017). TIOBE Index for Python, this graph shows the shifting popularity ratings of Python over the last 16 years.*
2.4 Use of Programming Languages within Professional Environments

2.4.1 Bioinformatics

Computer programming languages have aided the scientific community by equipping scientists with the right tools to progress extensively with regards to research. One study called “A comparison of common programming languages used in bioinformatics”, compares languages C, C++, C#, Java, Perl and Python in implementing bioinformatic methods. The study suggests that the appropriate selection of a programming language is vital to optimise speed and efficiency. It further states that C# and Java have the performance aspects of C++ with the practicality elements and usability of Python. In terms of bioinformatics, programming languages must handle large data sets, this means computation speed is essential. Currently, there are several dominant languages used for bioinformatics; these include R, Python and Perl. However, the use of each language varies in dominance; Perl which was once a key language used for bioinformatic roles is extensively being replaced by Python. Depending on the purpose of use, any programming language can be functional with regards to bioinformatics. R, is another programming language that is deemed extremely useful for quantitative data and statistical analysis. In comparison, languages like C and C++ are important for applications that are more reliant on memory and speed. Bioinformatic applications are usually created in order to store heavy pieces of information, such as DNA sequencing. It is also required to perform various functions regarding the sequence and subsequently produce a results compilation. Therefore, programming in languages such as Perl and Python is more popular with bioinformatic scientists as they both have the usability and capability to complete such extensive tasks. However, with regards to bioinformatic programming there are still a number of areas that require improvement these include storage space, increased computation speed and analytical presentation (Fourment and Gillings, 2008).

Figure 2.5. (Fourment and Gillings, 2008). This graph represents the time taken to execute the BLAST parsing program in C, C++, C#, Java, Perl and Python.
2.4.2 Economics
Roles relating to Economics are increasingly requiring the use of computer programming as a central tool. Complex mathematical models and simulations require sophisticated algorithms and therefore the most appropriate programming language. Economics, like bioinformatics deals with big data, this can impact global economies which handles sensitive data that needs to be immensely secure. A recent study “A Comparison of Programming Languages in Economics” finds that amongst the top three compiled languages C++ has the faster performance rate. C++ with a GCC compiler is the best performing language to use in Economics, although it is not the most user friendly. The results indicate that Python is considerably slower, which is interesting when relating back to the Indeed graph which clearly shows the progressing trend in Python programming. The study highlights the need to use a variety of languages in different areas of economics. The study also found that by using CPython interpreter the python implementation ran slower than running it using Pypy; this gives a good indication that interpreters and compilers can affect the performance of the programming language (Kendrick and Amman, 1999). The study also showed that the scripting language Julia had a phenomenal performance outcome in comparison to other scripting languages MATLAB and R. FORTRAN is another language that is widely used by economists, however, the performance speed was slower than that of Java. FORTRAN is recognised as one of, if not, the oldest language created; therefore, due to its consistent development it still retains significance within the scientific computing industry and economic fields (Da Cruz Vieira and Santana de Lelis, 2011). In economics, speed of language is not as important a factor as for computing related applications, rather, the reliability of algorithms and secure storage of data is key. The study also recognises the commercial use of C#, although, it suggests that the language is not best suited for economics, due to its computational properties. The information gathered clearly demonstrates that the performance difference between the languages, although optimising program solutions, can enhance the computational performance of any language. Therefore, depending on the economically related program, each language can provide a given solution to a particular situation, this may be dependent on performance and usability (Aruoba and Fernandez-Villaverde, 2014).

![TIOBE Index for C++](TIOBE, 2017) TIOBE Index for C++, this graph shows the shifting popularity ratings of C++ over the last 16 years.
2.4.3 Higher Education

Programming languages are taught in higher educational institutions; however, it is a topic that is heavily debated. It is still undecided as to what may be the best approach to take with regards to entry level programmers. It is an important factor, as the chosen language will bare significance upon a student’s understanding regarding the fundamental concepts. It will also help equip them with transferable skills that will help them comprehend other computing languages. In the early 90’s, Perl was the most popular language taught to entry level students, until Java, C++ and Python gained momentum; they were widely used throughout the late 90’s and are still currently very popular. Java and C# are object-oriented languages, whereas, Python is a scripting language. Some believe that starting with objects, classes and inheritance is premature, and instead concentrating on the procedural aspects of computing languages will better serve students (Farooq et al., 2014). The popular use of Java in higher educational settings is due to global use of the language, it has now become the standard language; it is highly developed by the ever-expanding Java community. Another popular language is Python; it is a simple and concise language, making it easier to understand as a beginner. However, it does not have the capability and constructs of general purpose languages such as Java and C# (ALEKSIĆ, V. and IVANOVIĆ, M., 2016).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada</td>
<td></td>
<td>15</td>
<td>19</td>
<td>18</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>8</td>
<td>11</td>
<td>12</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>C++</td>
<td></td>
<td>4</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>C#</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Fortran</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Java</td>
<td></td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>60</td>
<td>56</td>
</tr>
<tr>
<td>Modula-2</td>
<td></td>
<td>13</td>
<td>11</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pascal</td>
<td></td>
<td>40</td>
<td>33</td>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Python</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td>18</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 2.7. (Farooq et al., 2014). The table shows the percentage of the first programming languages taught to entry level students.

Figure 2.8. (Guo, 2014). The graph represents the programming languages taught by the top 39 U.S. computer science departments.
2.5 Conclusion
In general, programming languages can be utilised in various professions and industries. It is becoming more essential to have a certain degree of programming skill-sets. From developers within the banking industry, to biological scientists conducting crucial DNA analysis; everyone requires the use of modern, up-to-date programming technology. Therefore, it needless to say that finding the right language is important as it can help those within their specific fields to accomplish such specific tasks. For example, those within the financial field, like economists, require programming languages like R and MATLAB to process, analyse and store vast amounts of quantitative data. However, for individuals such as software developers, languages such as C# and Java provide the necessary facilities to accomplish tasks much more efficiently.

Figure 2.9. (TIOBE, 2017) TIOBE Programming Community Index representing the popularity of programming languages based on skilled engineers, worldwide courses and third party vendors.
3. METHODOLOGY

3.1 Introduction
The methodology of this study primarily focuses on the testing of the three key programming languages Java, C# and Python. To test the three languages, a test plan was introduced which includes eight separate tests, the test was created to determine the difference in usability and performance between the languages. The tests were coded in each language considering the most efficient methods of programming the small programs. Two key results were taken from each test, first being the number of lines of code and second being the time taken to execute the test program. The tests were conducted using an unbiased method, by placing a strict protocol for each set of tests for each of the languages, this is so to ensure valid and accurate results, and to avoid anomalies. There were two programs that were predominately used to conduct the test, C# and Python were used in Microsoft Visual Studio and Java was used in Eclipse. These two programs provided the environment necessary to run the tests in order to obtain accurate results; the results were then formatted onto a graph for comparison.

3.1.1 Integrated Development Environment (IDE)
Integrated Development Environments, are applications that provide the necessary tools for programming languages to write and run applications or programs. IDE’s provide graphical user interface (GUI), increasing the efficiency of application creation. Many IDE’s offer compatibility between languages such as NetBeans, Microsoft Visual Studio and Eclipse. The software offers in-built compilers, interpreters and debuggers; it makes it easier for programmers to view the effects of each line of code. External extensions can also be added onto most IDE’s increasing its features, such as, analysing tools and error diagnostics, which can manage and identify memory leaks and examine CPU usage. An emerging market is cloud IDE’s, this offers global code editors accessibility around the world by multiple users. The creation of community driven applications and software’s are hence increased; this enables the best methods to be implemented and increases the rate of software development.
3.1.2 Microsoft Visual Studio
Microsoft Visual Studio is an IDE, integrated development environment that was first created and released by Microsoft in 1997. The original purpose of the IDE was to produce and develop computer programs for Microsoft Windows. It has since progressed into a platform that can now develop websites, applications and web services. Microsoft Visual Studio is a proficient code editor as it can produce both native code and managed code; it also has capability to debug and refactor code. The IDE has great diagnostic tools with features such as a code profiler; it also accepts plug-ins to increase the functionality of the program (Crutchfield, 2017).

Figure 3.1. Downloading and installing Microsoft Visual Studio Community edition was the initial step taken to begin testing; this program is free and offers all the functionality needed to execute the test plan.

Figure 3.2. Microsoft Visual Studio comes with C# and its relevant libraries; although it does not have Python pre-installed there is a given option to install the language.
Figure 3.3. In Microsoft Visual Studio, below new project, a C# Console application is used to carry out each test, the reason for this is that it is the most basic application and the test will only need to display an alphabetical and numerical output.

Figure 3.4. When the application is created it already contains 15 lines of code; this includes the curly brackets, C# libraries, its namespace and lines of code to execute the program when the application is built.
3.1.2.1 Python Tools for Visual Studio (PTVS)

Figure 3.5. As Microsoft Visual Studio already contains C# as a preinstalled programming language and its relevant libraries, the next step was to install Python Tools for Visual Studio (PTVS). Python Interpreters can be installed on both Eclipse and Microsoft Visual Studio, although, having used Visual Studio prior to this study the workspace within the program is preferable (Microsoft, 2015).

Figure 3.6. Anaconda 4.3.1 is an open source Python interpreter package with rich libraries and an abundance of features. Python 2.7 64-Bit was installed because Microsoft Visual Studio Community 2015 is only compatible with Python 2.7 (Continuum, 2017).
Figure 3.7. After PTVS and Anaconda 4.3.1 was downloaded and installed, Microsoft Visual Studio automatically identifies the Python interpreter within its system. As seen in the figure above, Visual Studio also includes a Python tab below the new project tab.

Figure 3.8. On start-up, the Python application is empty, which bears a stark contrast to C#’s version. This already decreases the number of lines of code required to execute a program. As mentioned before, IDE’s include a great amount of functionality. Microsoft Visual Studio also includes Solution Explorer, as shown towards the top right of this figure; this organises each application’s file directory. Towards the bottom right of the image there is a properties toolbar which enables the manipulation of applications without the necessity of hard-coding.
3.1.3 Eclipse Neon
Eclipse is also a popular IDE that is commonly used for computer programming; it is most frequently used for Java development. It has the capability to also develop applications in other programming languages such as C++, FORTRAN and Perl (Eclipse, 2017).

Figure 3.9. From the Eclipse website a version of Eclipse Neon can be obtained and was installed, this IDE software allowed Java tests to be compiled and run. Eclipse is one of the most popular open source IDE’s available to download (Eclipse, 2017).

Figure 3.10. Once Eclipse was installed the next step was to install the Java SE Development Kit which provided a valid license of Java to be downloaded and installed onto the system (Oracle.com, 2017).

Figure 3.11. Eclipse automatically detects Java and makes it its default programming language. Eclipse has the functionality to also host Python programming language once the right extensions are installed, it is an area that can be explored in the future.
Figure 3.12. To create a new Java Project, the project name must be entered and the default Java must be checked. There are also two option regarding project file organisation.

Figure 3.13. Java includes default lines of code that are useful and provide the functionality to run the application.
3.2 Implementation of Test Plan

3.2.1 Test Plan
The performance and usability of programming languages are essential in professional environments. By using a variation of programs to simulate different workloads, the data gathered from the test plan will help determine the differences between the programming languages. There are four distinctive tests in the test plan, each of the four tests are repeated for a second time with the number of iterations increased from the previous version. Therefore, altogether there are eight tests. Each of the distinctive tests will measure different properties of the programming languages, with the increased number of iteration measuring the difference in workload. The test plan is shown in figure…

<table>
<thead>
<tr>
<th>Test Programs:</th>
<th>Definition:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello World [1]</td>
<td>Print “Hello World” on the console 1000 times.</td>
</tr>
<tr>
<td>Hello World [2]</td>
<td>Print “Hello World” on the console 10,000 times.</td>
</tr>
<tr>
<td>For Loop [3]</td>
<td>Count through and print on the console elements 1 to 1000.</td>
</tr>
<tr>
<td>For Loop [4]</td>
<td>Count through and print on the console elements 1 to 10,000.</td>
</tr>
<tr>
<td>Sum of Integers [5]</td>
<td>Using the sum algorithm calculate the sum of 50,000 elements, printing the answer on the console.</td>
</tr>
<tr>
<td>Sum of Integers [6]</td>
<td>Using the sum algorithm calculate the sum of 100,000 elements, printing the answer on the console.</td>
</tr>
<tr>
<td>Sorting Sequence [7]</td>
<td>Sort through 10,000 elements, only printing the square integers on console.</td>
</tr>
<tr>
<td>Sorting Sequence [8]</td>
<td>Sort through 50,000 elements, only printing the square integers on console.</td>
</tr>
</tbody>
</table>

Table 3.1. Displaying the Test Plan comprised of 8 test programs with corresponding definitions.

The test programs have been coded whilst taking into consideration the most appropriate methods for each programming language. The methods were programmed to be equal and unbiased. For each language, the test programs were executed and displayed on a console, the time taken for each execution were used to measure the performance; the lines of code (LOC) and its features will help to measure the usability of each language.

The Hello World [1] & [2] programs print “Hello World” multiple times on the console to test the performance of loading word constructs, For Loop [3] & [4], the for-loop function was used to test the performance of producing a set stream of numerical output onto the console. Test program Sum of Integers [5] & [6], tests the efficiency of performing arithmetic operations, this will give a good indication about which language copes best with numerical data. Finally Sorting Sequence [7] & [8], tests the performance of using multiple functions in a program with test [8] going through 50,000 elements, this will establish which languages work best with decision performance and big data.
3.2.2 Hello World Program

3.2.2.1 C# Implementation

Figure 3.14. A new project was opened, and a C# Console Application named “HelloWorldCSharp” was created.

Figure 3.15. A timer to time the execution of the program was also used. The times will be used to determine the speed of the execution. By importing “System.Diagnostics”, the stopwatch function can be used to time.

Figure 3.16. At beginning of the program the Stopwatch function is initialised “My Timer”, anything below is now timed. To print 1000 “Hello Worlds” onto the console, a for loop is used. “MyTimer.Stop();” will stop the timer once the Hello World cycle is completed.
Figure 3.17. To complete the program a “Console.WriteLine” is included to print out the time taken to execute the program. The timer is timed in milliseconds, it is one of the most accurate timers included in C# libraries.

Figure 3.18. A test demo shows that the program is working as theorised. The for loop checks if “i” is less than or equal to 1000, if it is not, then print “Hello World”. Once it is, stop and print “Time Taken”.

Figure 3.19. Hello World [2] consists of 10,000 iterations. This screenshot shows the program now prints “Hello World” 10,000 times.
3.2.2.2 Java Implementation

Figure 3.20. A new project was opened and a Java project named “HelloWorldJava” was created.

Figure 3.21. After much consideration, the timer “System.nanoTime();” was included as it represents a more accurate execution time for small programs. This is compared to the timer function “System.currentTimeMillis();”.

Figure 3.22. The “startTime” initialises the timer, below the for loop, like for C#, it will print 1000 “Hello World” executions onto the console. There are only a few minor differences between the codes.
Figure 3.23. Finally, the “startTime” is taken away from the variable duration to calculate the total time taken to execute the program. This is printed onto the console. A demo test is also shown to represent the program working as theorised.

Figure 3.24. This screen shot represents test Hello Word [2], with 10,000 iterations. As the test will be conducted in milliseconds, the nanoseconds from the Java tests will be converted as appropriate.
3.2.2.3 Python Implementation

Figure 3.25. A new project was opened and a Python Application named “HelloWorldPython” was created.

Figure 3.26. After much research ‘timeit’ seemed to be the best option to time code execution. To use ‘timeit’, it was first imported by calling the function “import timeit”.

Figure 3.27. The first attempt to code “Hello World” proved successful. Although, the method used to print the statement was different to C# and Java.
Figure 3.28. Instead a “for i in range()” was used to resemble the for loop cycle. This method means that for every number up to 1000 print “Hello World”. This will be printed onto the console using the timeit function and repeated once.

Figure 3.29. A demo test proves the edited version works perfectly with the time taken to execute shown at the bottom in seconds.

Figure 3.30. This screenshot represents the Hello World [2] test with a range of 10,000. As the Python program measures in seconds, a conversion to milliseconds was completed after the tests.
3.2.3 For Loop Program

3.2.3.1 C# Implementation

Figure 3.31. The next application is For Loop [1] & [2]. A new project was opened and a Console Application named “ForLoopCSharp” was created.

Figure 3.32. Now there is a format which can be coded to measure the execution time. The same follows insert “System.Diagnostics;”, followed by a new Stopwatch called “MyTimer” and start “MyTimer”. In between start and stop “MyTimer” the test program is written.
Figure 3.33. A for loop was used once again to cycle through 1000 elements with each of them printed onto the console. This time instead of the same words printed numerous times, each numerical character was printed from 1 to 1000.

Figure 3.34. The screen shot shows a demo test, representing the intended purpose of the test.

Figure 3.35. This screen shot represents the For Loop [4] test 10,000 iteration.
3.2.3.2 Java Implementation

Figure 3.36. A new project opened, and a Java Project named “ForLoopJava” was created.

Figure 3.37. The “System.nanoTime” function was first inserted. It is a “long” variable type to save memory allocation which enhanced the speed of the program.

Figure 3.38. Similar to C#, the for loop cycled through each numerical character and printed it on the screen.
3.2.3.3 Python Implementation

Figure 3.41. To create For Loop [1] & [2] tests in Python. A new project was opened and a Python Application named “ForLoopPython” was created.

Figure 3.39. This is a demo test of For Loop [3], which worked as predicted, it is also measured in nanoseconds.

Figure 3.40. The screen shot shows the demo of the For Loop [4] test.
Figure 3.42. First by importing “timeit”, execution time can be measured. The method “for count in range” essentially counts through from 1 to 1000. The “print(count)” printed the number onto the

Figure 3.43. Once the code for the For Loop [3] was established, the only thing left was to time the execution.
Figure 3.44. The screen shot shows the demo of For Loop [3] test using Python Application.

Figure 3.45. This screen shot shows the For Loop [4] test with count set to 10,000; with the execution time measured in seconds.
3.2.4 Sum of Integers

3.2.4.1 C# Implementation

Figure 3.46. To create Sum of Integers [5] & [6] tests in Python. A new project was opened and a Console Application named “SumCSharp” was created.

Figure 3.47. The code needed to execute the Sum of Integers was entered in between “MyTimer”.

Figure 3.48. IEnumerable function is a simple iteration over a non-generic collection, in this case it has a range of 1 to 50,000 and it named “num”.

[34]
Figure 3.49. A “foreach” is used to run through the range using the “Sum” to create a running total which is stored into the “total” variable. The “total” variable is printed onto the console along with the time taken to execute the

Figure 3.50. The screen shot shows the demo test of Sum of Integer [5], with the execution time measured in milliseconds.

Figure 3.51. The screen shot shows the Sum of Integer [6] test with range set to 50,000, with the execution time measured in
3.2.4.2 Java Implementation

Figure 3.52. To create the Sum of Integer test, a Java Project called “SumJava” in Eclipse was created.

Figure 3.53. By putting in place the “nanoTime()” function, it was used to time the execution time.

Figure 3.54. An integer list was created, this can hold 10,000 elements. Then “i” was inputted into the list labelled “ia”.

[36]
Figure 3.55. The “sum” variable was set with the “int” type to value zero. Then a “for” function was used to calculate the sum of the “ia” list. After which, this was printed onto console and a demo test

Figure 3.56. Sum of Integer [6] with 50,000 elements proves to be working by the value on the console.
3.2.4.3 Python Implementation

Figure 3.57. A Python Application called “SumPython” in Microsoft Visual Studios was created.

Figure 3.58. A timeit function was imported to use the timer to time the execution of the program. Then a variable called “a” with the range 10,000 was created. As the test is sum of 10,000, this means the answer should be 50005000.

Figure 3.59. Variable “b” includes the total sum of “a” by using the “sum()” function which calculates the sum. This only needs to run once, this is then printed onto the console.

[38]
3.2.5 Sorting Sequence

Figure 3.60. The demo test ran as planned with the correct result for the Sum of Integer [5] test.

Figure 3.61. To do Sum of Integers [6] test, the range must be changed from 10,001 to 50,001 to get the sum of 50,000.

Figure 3.62. The screen shot shows a successful Sum of Integer [6] test with the correct result.
3.2.5.1 C# Implementation

**Figure 3.63.** The last test of the Test Plan is Sorting Sequence. To create C#’s implementation, a new Console Application was created called “SortingCSharp”.

**Figure 3.64.** “MyTimer” was setup to time the execution of the test. A “Console.WriteLine” was used to print the time onto the console which was measured in milliseconds.

**Figure 3.65.** By inputting this code, I can initialise an “IEnumerable” with the “Int64” data type.
Figure 3.66. A public class was created to call upon “Range.Int64” in my ‘Main()’ class. This will insure that the answer achieved will have the correct memory allocation.

Figure 3.67. This is the complete IEnumerable interface which will produce the square numbers in a given range. There may be more efficient ways of producing the test, although, this was the best method that produced the correct results.

Figure 3.68. In the ‘Main()’ class Range.Int64 was called upon, this has the range between 1 to 50000. Then the method to square the numbers in the range, is followed. This is printed onto the console with the time taken to execute.
Figure 3.69. This screenshot shows the functioning demo displaying the Sorting Sequence [7].

Figure 3.70. The alteration of the range to 100,000 elements making the Sorting Sequence [8] test.

Figure 3.71. This screenshot shows the functioning demo displaying the Sorting Sequence [8].
3.2.5.2 Java Implementation

Figure 3.72. To implement a Sorting Sequence test in Java a New project was created called SquareJava in Eclipse.

```java
public class SquareJava {
    public static void main(String[] args) {
        final long startTime = System.nanoTime(); // Initialiser, start timer in nanoseconds.
        System.out.println(square); // Print variable "square" onto the console.
    } figure

    final long duration = System.nanoTime() - startTime; // Stop timer and calculate total time taken. System.out.println("Time Taken: "+ duration); // Print onto console time token to execute program.
}
```

Figure 3.73. A timer was initialised to measure the Sorting Sequence execution time.

```java
public class SquareJava {
    public static void main(String[] args) {
        final long startTime = System.nanoTime(); // Initialiser, start timer in nanoseconds.
        final[] array = new long[50001]; // Create a list with 50,001 elements.
        for (int i = 0; i < 50001; i++) {
            // Run through 0 to 50,000.
            array[i] = i + 1; // Store 'i' in 'array' list.
            System.out.println(square); // Print variable "square" onto the console.
        }
    }
```

Figure 3.74. An ‘array’ list was then created, this contained 50,001 elements. A for loop ran through 50000 elements; storing ‘i’ in the ‘array’ list.
Figure 3.75. A variable called ‘square’ was created this squared and stored the ‘i’ variable. Then, the ‘square’ variable was printed on the console and the for loop would continue until 50,000 elements are reached. This console shows that the demo test worked perfectly.

Figure 3.76. This screenshot shows the functioning demo displaying the Sorting Sequence [8].
3.2.5.3 Python Implementation

Figure 3.77. To implement a Sorting Sequence test in Python, a Python Application was created called SortingPython in Microsoft Visual Studio.

Figure 3.78. ‘timeit’ was imported in order to obtain the execution time of the program. Variable a contains the range of 50,000 elements, variable ‘c’ squares the numbers in the range and print them onto the console.

Figure 3.79. This is the final code for the Sorting Sequence [7] test with the ‘timeit’ function.
Figure 3.80. This screenshot shows the functioning demo displaying the Sorting Sequence [7].

Figure 3.81. The alteration of the range to 100,000 elements making the Sorting Sequence [8] test.

Figure 3.82. This screenshot shows the functioning demo displaying the Sorting Sequence [8].
3.3 Testing

3.3.1 The System Properties
The system properties are important to recognise as the Test Plan may be repeated using a different system and may produce different results.

System Properties:

- Manufacture: Hewlett-Packard
- Model: ENVY
- Processor: Intel® Core ™ i7-6560u CPU @ 2.20 GHz
- Installed memory (RAM): 8.00 GB (7.89GB usable)
- System type: 64-bit Operating System, x64-based processor
- Operating system: Windows 10 Home.
- Storage: 1 TB HDD & 128 GB SSD

3.3.2 Fair Test Procedure
It is crucial to have a fair environment for each programming language to help obtain the most accurate results. With a set procedure, unreliable results and anomalies can be avoided. To ensure that the substance of the Test Plan was precise, protocols were put in place before the tests began and strictly followed throughout. Warm up tests were put in place to ensure CPU performance was optimised.

3.3.3 Procedures for Test Plan

- A full reboot of the system before a new programming language is tested.
- Once the system is booted wait two and a half minutes then open the appropriate IDE software.
- Open the first Hello World test in the appropriate programming language.
- Wait a further one and a half minutes to open the Test Plan document to store screen shots of the tests after completion.
- Wait a further one minute, then executed Hello World [1]. (Warm up test).
- After 30 seconds, execute the test three more times with 30 second intervals.
- Change to Hello World [2], in the last 30 second interval.
- At the end of the 30 seconds execute Hello World [2]. (Warm up test).
- After 30 seconds, execute the test three more times with 30 second intervals.
- Change to For Loop [3], in the last 30 second interval.
- At the end of the 30 seconds execute For Loop [3]. (Warm up test).
- After 30 seconds, execute the test three more times with 30 second intervals.
- Change to For Loop [4], in the last 30 second interval.
- At the end of the 30 seconds execute For Loop [4]. (Warm up test).
- After 30 seconds, execute the test three more times with 30 second intervals.
- Change to Sum of Integers [5], in the last 30 second interval.
- At the end of the 30 seconds execute Sum of Integers [5]. (Warm up test).
- After 30 seconds, execute the test three more times with 30 second intervals.
• Change to Sum of Integers [6], in the last 30 second interval.
• At the end of the 30 seconds execute Sum of Integers [6]. (Warm up test).
• After 30 seconds, execute the test three more times with 30 second intervals.
• Change to Sorting Sequence [7], in the last 30 second interval.
• At the end of the 30 seconds execute Sorting Sequence [7]. (Warm up test).
• After 30 seconds, execute the test three more times with 30 second intervals.
• Change to Sorting Sequence [8], in the last 30 second interval.
• At the end of the 30 seconds execute Sorting Sequence [8]. (Warm up test).
• After 30 seconds, execute the test three more times with 30 second intervals.
• End of test.
• Repeat the procedure for the next programming language.

3.3.4 Testing Screenshots
The testing screenshots can be found in:

- Appendix B – Java Test Plan
- Appendix C – C# Test Plan
- Appendix D – Python Test Plan

3.3.5 Methodology Conclusion
To conclude it has been useful to understand the variety of methods used to code the small programs within the Test Plan. This has helped to develop a more proficient study that will provide valid data for further analysis. During the methodology stage, there has been critical assessment of the most popular methods of implementation which has helped to produce an appropriate test plan that will establish a greater understanding for this study.
4. FINDINGS & ANALYSIS

4.1 Findings

4.4.1 Java Test Results

<table>
<thead>
<tr>
<th>Test Programs</th>
<th>LOC</th>
<th>Test 1:</th>
<th>Test 2:</th>
<th>Test 3:</th>
<th>Mean:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello World [2]</td>
<td>14</td>
<td>152.606</td>
<td>152.817</td>
<td>122.375</td>
<td>142.600</td>
</tr>
<tr>
<td>For Loop [4]</td>
<td>14</td>
<td>80.054</td>
<td>125.185</td>
<td>62.964</td>
<td>89.401</td>
</tr>
<tr>
<td>Sum of Integers [5]</td>
<td>18</td>
<td>0.362</td>
<td>0.365</td>
<td>0.339</td>
<td>0.355</td>
</tr>
<tr>
<td>Sum of Integers [6]</td>
<td>18</td>
<td>1.327</td>
<td>1.273</td>
<td>1.159</td>
<td>1.253</td>
</tr>
<tr>
<td>Sorting Sequence [7]</td>
<td>22</td>
<td>46.874</td>
<td>76.536</td>
<td>45.193</td>
<td>56.201</td>
</tr>
<tr>
<td>Sorting Sequence [8]</td>
<td>22</td>
<td>1021.200</td>
<td>949.632</td>
<td>1384.444</td>
<td>1118.426</td>
</tr>
</tbody>
</table>

Table 4.1. Time Taken & Lines of Code Produced when Executing the Test Plan using Java in Eclipse (milliseconds)(3d.p.).

4.4.2 C# Test Results

<table>
<thead>
<tr>
<th>Test Programs</th>
<th>LOC</th>
<th>Test 1:</th>
<th>Test 2:</th>
<th>Test 3:</th>
<th>Mean:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello World [2]</td>
<td>28</td>
<td>3691.547</td>
<td>3074.167</td>
<td>3781.651</td>
<td>3515.788</td>
</tr>
<tr>
<td>For Loop [3]</td>
<td>29</td>
<td>503.460</td>
<td>590.519</td>
<td>652.976</td>
<td>582.319</td>
</tr>
<tr>
<td>For Loop [4]</td>
<td>29</td>
<td>1397.794</td>
<td>4184.912</td>
<td>4356.368</td>
<td>3313.015</td>
</tr>
<tr>
<td>Sum of Integers [5]</td>
<td>30</td>
<td>0.657</td>
<td>1.408</td>
<td>0.625</td>
<td>0.900</td>
</tr>
<tr>
<td>Sum of Integers [6]</td>
<td>30</td>
<td>1.043</td>
<td>1.0777</td>
<td>1.531</td>
<td>1.217</td>
</tr>
<tr>
<td>Sorting Sequence [7]</td>
<td>59</td>
<td>3007.737</td>
<td>2638.399</td>
<td>3125.498</td>
<td>2923.878</td>
</tr>
<tr>
<td>Sorting Sequence [8]</td>
<td>59</td>
<td>5831.650</td>
<td>6129.235</td>
<td>5828.600</td>
<td>5923.695</td>
</tr>
</tbody>
</table>

Table 4.2. Time Taken & Lines of Code Produced when Executing the Test Plan using C# in Microsoft Visual Studio (milliseconds)(3d.p.).

4.4.3 Python Test Results

<table>
<thead>
<tr>
<th>Test Programs</th>
<th>LOC</th>
<th>Test 1:</th>
<th>Test 2:</th>
<th>Test 3:</th>
<th>Mean:</th>
</tr>
</thead>
<tbody>
<tr>
<td>For Loop [3]</td>
<td>3</td>
<td>879.338</td>
<td>891.300</td>
<td>878.841</td>
<td>883.600</td>
</tr>
<tr>
<td>For Loop [4]</td>
<td>3</td>
<td>8053.725</td>
<td>8032.853</td>
<td>8145.520</td>
<td>8077.366</td>
</tr>
<tr>
<td>Sum of Integers [5]</td>
<td>5</td>
<td>0.768</td>
<td>0.872</td>
<td>0.952</td>
<td>0.864</td>
</tr>
<tr>
<td>Sum of Integers [6]</td>
<td>5</td>
<td>1.684</td>
<td>1.639</td>
<td>2.617</td>
<td>1.980</td>
</tr>
<tr>
<td>Sorting Sequence [7]</td>
<td>5</td>
<td>145.122</td>
<td>144.795</td>
<td>150.066</td>
<td>146.661</td>
</tr>
<tr>
<td>Sorting Sequence [8]</td>
<td>5</td>
<td>371.523</td>
<td>386.667</td>
<td>372.735</td>
<td>376.975</td>
</tr>
</tbody>
</table>

Table 4.3. Time Taken & Lines of Code Produced when Executing the Test Plan using Python in Microsoft Visual Studio (milliseconds)(3d.p.).
### 4.4.4 Test Results Charts

**Figure 4.1.** The execution time for the Hello World [1] & [2] tests using Java, C# and Python.

**Figure 4.2.** The execution time for the For Loop [3] & [4] tests using Java, C# and Python.
The Execution Times of the 'Sum of Integers' Tests in Java, C# and Python Programming Languages

Test Programs
- Java
- C#
- Python

**Figure 4.3.** The execution time for the Sum of Integers [5] & [6] tests using Java, C# and Python.

The Execution Times of the 'Sorting Sequence' Tests in Java, C# and Python Programming Languages

Test Programs
- Java
- C#
- Python

**Figure 4.4.** The execution time for the Sorting Sequence [7] & [8] tests using Java, C# and Python.
4.2 Analysis

4.2.1 Introduction
In this section, there will be a discussion regarding the findings for this study, including the performance, usability and functionality of the three programming languages. The aim of this section is to analyse the purpose of the languages and examine the results in order to outline the findings of this study.

4.2.2 Performance
The test plan gives a good indication regarding the performance and capability of the programming languages. From the results, it is clear to see that Java, for the most part, is faster than C# and Python. The first two tests, Hello World [1] & [2], tested the programming languages by printing out numerous copies of “Hello World” using a for loop. In the Hello World [1] test, Java was 32 times faster than C#, in addition, C# performed slightly better than as it was 1.3 times faster than Python. In the Hello World [2] test, Java surpassed the other two languages again demonstrating that it can cope well with increased iterations, however, the interesting fact is that Java was only 25 times faster than C#; therefore, this shows that C# has improved performance whilst dealing with a bigger workload. Java was 57 times faster than Python, this gives an indication that Python slowed down further whilst tackling a bigger workload in relation to this test.

The tests For Loop [3] & [4], measured the languages by using a for loop to cycle through numerical data. In the For Loop [3] test, as predicted Java was the best performer, in addition, the variance between the languages increased. In this test, the Python language using Microsoft Visual Studios, was 64 times slower than Java, and a further 90 times slower in the For Loop [4] test. Once Again, C# was faster than Python; however, it was far behind the performance of Java, although it is still better at dealing with bigger workloads as it was quicker in the For Loop [4] test.

The tests Sum of Integers [5] & [6] tests, produced some interesting results. The tests challenged the languages by increasing the workload by calculating the sum of a particular range of numbers. For Sum of Integers [5], Java was only 2 times faster than both C# and Python, although for Sum of Integers [6], C# was 1.03 times faster than Java. This means that C# works more proficiently when calculating large mathematical equations; Python also had increased capability as the time taken was not too far behind that of C#; it was only 1.6 times slower.

In the final tests, Sorting Sequence [7] & [8], the competence of Python was established. The tests required the programming languages to square all the numbers in a particular range. In the test, Sorting Sequence [7], it was interesting to see that Java was 52 times faster than C#, the difference is large, which indicates that C# suffers with regards to logical performance. On the other hand, Python coped well and was only 2.6 times slower than Java. Although, in the Sorting Sequence [8] test, Python, with the range of 100,000 came out on top as it was almost 3 times faster than Java. This definitely indicates that the logical performance Python is far superior. In this test, Java was only 5 times faster than C#, again proving C# performs better with bigger workloads.
4.2.3 Usability

4.2.3.1 Uses
From the research and tests, Python fits perfectly into the simple script category; this means it is best for quick and simple code snippets as it does not function with too much overhead like the other programming languages. Java and C# are high performance languages, which have the capability to program cross platform graphical applications, games and servers.

4.2.3.2 Syntax
Regarding syntaxial differences, it is obvious that Java and C# are forced to use semicolons for functions, whereas for Python it is optional. Curly braces are often used in Java and C# to include blocks or sections of code which may help to organise and maintain the code; this feature is not used in Python, therefore maintaining code is a difficult task. Java and C# are also strong type programming languages; from the tests screen shots, you can see that the type of data that is being manipulated must be explicitly stated, whereas in Python that is not the case. In Java, there are packages which can be imported to access classes to be used in a program, these packages can be accessed through JAR files. C# uses the vast .Net libraries to access external function and can be brought in by declaring “using”; this is then followed by the extension needed.

As you can see from the figure above Python programming language required less code density when executing the tests in the test plan, Java followed second and C# was last.

Figure 4.5. Line of code needed to execute the tests Hello World, For Loop, Sum of Integers and Sorting Sequence using Java, C# and Python.
4.2.3.3 Runtime
Python is an interpreted programming language, this means Python Tools for Visual Studio converts the programs into machine code to run on the system, this would be stored as a source code. The source code is useful because it can be used on any operating system, whereas compiled programming languages are hardware dependent. Java and C# are Just In Time compiled programming languages, this signifies that the programs are converted into machine code whilst running; again this has the same advantages as interpreted languages however it performs better. All three languages make use of a garbage collector in terms of memory management, this is an efficient method of controlling the use of memory in the system.

4.3 Conclusion
The results gained from this study indicate that Java is a far superior programming language when comparing to Python and C#; this is not a surprising outcome as many current programmers believe Java to be greater and prior studies also confirm this fact. However, this point is not a weakness, but instead a strength for this study, as it indicates that the study was conducted under fair conditions and in an unbiased environment. However, recent trends indicate a shift in correlation with regards to programming language in relation to job roles. Therefore, when analysing the results gained from this study it provides substantial information as to why there is this shift. Firstly, from the results, it is clear to see that Python is the slowest performing programming language in comparison to both Java and C# on most occasions, however, with regards to the usability it far out performs Java and C# as it requires less lines of code and has less limitations. This may specify why Python has had a recent increase in popularity, it is easier to learn and more enjoyable to use as less code is required in order to program; therefore, there is a faster turnaround of programs.
5. Conclusion and Limitations

5.1 Conclusion
This study aimed to identify the link between the popularity of programming languages and its relation to the increasing shift in job trends with regards to the language used. From the onset of this study it was clear to see that the job trends graph provided by the Indeed site; and popularity graphs such as the TIOBE Index have a strong and valid link. Using a test plan, the performance and usability aspects of the programming languages Java, C# and Python were tested. The results achieved from the tests, provided a greater understanding about the performance and usability of the programming languages and its general wider effect on job roles within relating fields and industries. Earlier studies have previously established the impact programming languages have had on improving specific job sectors such as within Bioinformatics and Economics.

Programming languages can be utilised in various professions and within multiple industries; it is extremely useful to possess a certain level of programming skills. From the research completed during the literature review it became apparent that programming technology was increasingly being employed, this increase has led to the consistent development and improvement of the programming languages. Each language has its advantages and disadvantages based on the intended use; as previously stated. Discovering the correct language appropriate for an intended purpose can vastly improve the efficiency of the applications. For example, languages such as R and MATLAB are important for tasks based around quantitative data and therefore have the capability to store, analyse and process this data; this is useful for those who require data analysis on a larger scale, such as economists (Aruoba and Fernandez-Villaverde, 2014). Whereas, languages like C#, Java and Python are object-orientated and provide the necessary programming facilities required to process tasks such as application development which is useful to individuals such as software developers.

During the process of creating this study it was also useful to comprehend the varying methods used to code each of the individual languages. Firstly, determining the chosen software for this study was accomplished by researching the relevant IDEs that were appropriate for the three object-orientated programming languages. Different methods were used during the coding process; for example, measuring the execution time for the tests required the most accurate timing function, however, each of the languages have multiple timing functions; therefore, discovering ‘timeit’, ‘system.nanoseconds’ and ‘current milliseconds’ enabled the tests to include fair and unbiased data (Docs.python.org, 2017). Test procedures were also put in place to gather valid data for the study.

In terms of syntax and its methods, Python is very different when comparing to C# and Java. From the research, it was clear to see Python was steadily growing in popularity, to the point where it over-took C#. The research showed that Python was taught in the top 8 universities in United States of America, corresponding to this the results from this study showed that the code density of the language was far less in-depth when comparing to its competitor C#. The code density is important as it establishes the level of usability to the programmer; this is very important to students entering higher education courses relating to programming languages. Although, Java and C# are strongly typed languages; this is important to entry level students as it provides a greater understanding regarding the fundamental concepts relating to programming, which enables them to become a competent programmer (Nanz et al., 2014).
From the Tiobe Index there is an obvious decline in the use of C#; C# developers also receive less salary in comparison to their counterpart Java programmers (Gooroo, 2017). This may be due to the close similarity between Java and C#, however from the results it is clear to see that C# does not have the same performance capability as Java. Therefore, programmers may find other programming languages such as Java or C++ a better option depending on the functionality of the application. Nevertheless, C# is still useful for particular programmers, as it is perfect for visual applications due to its .Net library. From the results, the difference between the performance of C# and Python was minimal; as previously stated Python has a greater usability than C# this may explain the reason why there is an increased interest in Python programming.

From the research, it was clear to see that Java is still the most popular programming language used in all sectors. The usability aspect is far better than that of C#, however, it is not as great as Python. In the future, if there was further development of Python programming that led to the increased performance of the language, then there is a viable chance that Python may surpass Java as the more preferred choice of programming language.

5.2 Limitations

The study can be expanded by enhancing the test plan; for example, by increasing the workloads of the tests; the tests may include the creation of graphical applications, websites and servers. These tests will help to provide a more in-depth understanding regarding performance and usability of the languages which will influence the global perception of programming languages in general. In this study, IDEs such as Microsoft Visual Studio and Eclipse were used to conduct the tests. Another viable option that may increase the validity and accuracy of the study, is to use Eclipse to run Python services. The reason for this is that Eclipse is more widely used than Microsoft Visual Studio has a better cross-platform environment. Using standard compilers, for example, Pypy is another option; this will deliver a better performance as it will not require the memory consumption used by IDEs.

![Most Popular Desktop IDEs & Code Editors in 2014](image)

**Figure 5.1.** (Codeanywhere, 2017) Most popular desktop IDEs & code editors in 2014.
6. REFERENCES


[58]


8. APPENDIX

Appendix A
Ethics Approval Number - 2016D0450

When undertaking a research or enterprise project, Cardiff Met staff and students are obliged to complete this form in order that the ethics implications of that project may be considered.

If the project requires ethics approval from an external agency (e.g., NHS), you will not need to seek additional ethics approval from Cardiff Met. You should however complete Part One of this form and attach a copy of your ethics letter(s) of approval in order that your School has a record of the project.

The document Ethics application guidance notes will help you complete this form. It is available from the Cardiff Met website. The School or Unit in which you are based may also have produced some guidance documents, please consult your supervisor or School Ethics Coordinator.

Once you have completed the form, sign the declaration and forward to the appropriate person(s) in your School or Unit.

PLEASE NOTE:
Participant recruitment or data collection MUST NOT commence until ethics approval has been obtained.

PART ONE

<table>
<thead>
<tr>
<th>Name of applicant:</th>
<th>Vidhuran Sivabalan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervisor (if student project):</td>
<td>Dr Ana Calderon</td>
</tr>
<tr>
<td>School / Unit:</td>
<td>Cardiff School of Management</td>
</tr>
<tr>
<td>Student number (if applicable):</td>
<td>St20111090</td>
</tr>
<tr>
<td>Programme enrolled on (if applicable):</td>
<td>BSc (Hons) Computing</td>
</tr>
<tr>
<td>Project Title:</td>
<td>To investigate the difference between top programming languages within the industry.</td>
</tr>
<tr>
<td>Expected start date of data collection:</td>
<td>10/01/2017</td>
</tr>
<tr>
<td>Approximate duration of data collection:</td>
<td>1 Month</td>
</tr>
<tr>
<td>Funding Body (if applicable):</td>
<td>N/A</td>
</tr>
<tr>
<td>Other researcher(s) working on the project:</td>
<td>N/A</td>
</tr>
<tr>
<td>Will the study involve NHS patients or staff?</td>
<td>No</td>
</tr>
<tr>
<td>Will the study involve human samples and/or human cell lines?</td>
<td>No</td>
</tr>
</tbody>
</table>
Does your project fall entirely within one of the following categories:

<table>
<thead>
<tr>
<th>Category</th>
<th>Yes/No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper based, involving only documents in the public domain</td>
<td>Yes</td>
</tr>
<tr>
<td>Laboratory based, not involving human participants or human samples</td>
<td>No</td>
</tr>
<tr>
<td>Practice based not involving human participants (eg curatorial, practice audit)</td>
<td>Yes</td>
</tr>
<tr>
<td>Compulsory projects in professional practice (eg Initial Teacher Education)</td>
<td>No</td>
</tr>
<tr>
<td>A project for which external approval has been obtained (e.g., NHS)</td>
<td>No</td>
</tr>
</tbody>
</table>

If you have answered YES to any of these questions, expand on your answer in the non-technical summary. No further information regarding your project is required.

If you have answered NO to all of these questions, you must complete Part 2 of this form.

In no more than 150 words, give a non-technical summary of the project:

Collect and compare data concerning programming languages, through peer reviewed journals. Then devise a set of tests that will provide primary data to make a conclusion based on the analysis of the results.

**DECLARATION:**

I confirm that this project conforms with the Cardiff Met Research Governance Framework.

I confirm that I will abide by the Cardiff Met requirements regarding confidentiality and anonymity when conducting this project.

**STUDENTS:** I confirm that I will not disclose any information about this project without the prior approval of my supervisor.

Signature of the applicant: ___________________________ Date: 04/01/2017

**FOR STUDENT PROJECTS ONLY**

Name of supervisor: ___________________________ Date: 08/02/2017

Signature of supervisor:

2016D0450
Appendix B

Java Test – Hello World [1]
Java Test – Hello World [2]
Java Test – For Loop [3]
Java Test – For Loop [4]
public class FortuneLane {

public static void main(String[] args) {

    final long startime = System.nanoTime();

    for (int i = 0; i < 1000000000; i++) {
        System.out.println("==>");
        System.out.println("<==");
    }

    final long duration = System.nanoTime() - startime;
    System.out.println("Time Taken: " + duration + " ns");
}
}
Java Test – Sum of Integers [5]
Java Test – Sum of Integers [6]
Java Test – Sorting Sequence [7]
Java Test – Sorting Sequence [8]
Appendix C

C# Test – Hello World [1]
C# Test – Hello World [2]
C# Test – For Loop [3]
C# Test – For Loop [4]
C# Test – Sum of Integers [5]
C# Test – Sum of Integers [6]
C# Test – Sorting Sequence [7]
C# Test – Sorting Sequence [8]
Appendix D

Python Test – Hello World [1]
Python Test – Hello World [2]
Python Test – For Loop [3]
Python Test – For Loop [4]
Python Test – Sum of Integers [5]
Python Test – Sum of Integers [6]
Python Test – Sorting Sequence [7]
Python Test – Sorting Sequence [8]