A Pragmatic Comparison of Four Different Programming Languages

Professor: Mamona Mumtaz
Departament of Computer Science

Authors: Saqib Ali
Sammar Qayyum
A Pragmatic Comparison of Four Different Programming Languages

Abstract:

Programming language debates are extremely common among programmers, computer science students and of course, software engineers. However, all of the above-mentioned professionals can concur that different languages excel in different scenarios. Software Engineers and programmers working on different projects can easily use different languages for different tasks during their work. Every year different programming languages are designed and created. In this Research, we will keep in focus the Four Horsemen of programming languages: C, C++, Python and Java; with respect to the criteria of time, speed and simplicity. The same optimized piece of pseudo-code is used to write the code of the different programming languages mentioned above by following their respective syntax and rules. The results of the comparison will be displayed with the help of a table in order to simplify the final results for the reader.

1 Introduction

“A Programming Language is like a natural, Human Language in that it favors certain metaphors, images and way of thinking” — Seymour Papert

In 1950s, the first of the high-level programming languages were created and designed, with the purpose of shielding the programmer from the amount of effort it took to design even the simplest of programs in low level programming languages [2].

There are thousands of different programming languages that exist and different sources say different numbers. Wikipedia states that are around 700 different Programming Languages but HOPL states that there are 8,495, programming languages to date [3]. But, don’t let this distract you from the fact that there are many new languages created each year all having different functionalities, capabilities and paradigms (Functional, object-oriented, reflective, aspect-oriented and imperative being the common ones).

There are some programming languages (e.g. Python, C, Java and C++) that are popular world wide and are used in almost every field because they are highly generalized and offer extreme control. However, language developers aim to create new languages whose purpose is to combine simplicity, power and efficiency and having support for multiple paradigms [2]. The main purpose of multi-paradigm languages is to facilitate the programmer so he can use different paradigms for different projects whichever is more efficient in any given scenario and realizing that any particular paradigm is not the best one.
There are so many languages out there but some of the languages are extremely popular like C, C++, Java and Python. So, Which language is better suited towards which task? Which language of these bunch is faster? Which offers more control and is easier to program in? In our research we will be performing two experiments and after that we will be able to draw a conclusion to answer these questions.

1.1 Similar Work:

In the Journal, “A Comparative Study of Language Support for Generic Programming” [4] detailed comparison of generic programming is carried on six languages: Standard ML, Haskell, Java, C++, C#, Eiffel but Python is absent in this research journal, which is now one of the most widely used language in the field of Artificial Intelligence and Data Science.

In the Journal, “Programming Languages: A Comparative Study” [2] authors have put plenty of effort and time to compare and contrast four languages: Lisp, C++, Perl and Java to assess their efficiency, complexity, re-usability familiarity, their IDE’s. The authors also studied the number of lines, resources required and the memory consumption to implement the same program among these languages but once again Python is absent from this list.

In the Journal, “A Comparative Study between Computer Programming Languages for Developing Distributed Systems in Web Environment” [1] the authors studied and evaluated the performance of C++, Java and C# in the field of distributed systems in the web environment. In relation to the field, they also measured their simplicity, reliability, concurrency and maintainability of the languages. This work, however, also excludes Python from its list with its rich and easy to use web frameworks like Django and Flask.

1.2 C:

C is a successor to the programming language B. It was developed by the brilliant Dennis Ritchie at Bell Labs between 1972-1973. C gained popularity gradually and became popular enough to become the most widely used programming language. It provides an Imperative programming approach. It is a compiled programming language with low-level memory accessibility and functions like pointers. It was used to develop many low-level applications including operating systems such as Windows. It is a general-purpose programming language and it possesses great speed and efficiency when used by an expert programmer. It however, lacks Object Oriented programming.
1.3 C++:

C++ was designed and developed by Bjarne Stroustrup as an expansion to C. It was released in 1985, but in 1989 its newer version was released which added more advanced features while also adding error handling. It is a general-purpose programming language and is mostly similar to C with some key differences. It is a compiled programming language with a blend of both low and high-level features making it highly versatile. It supports full object-oriented programming, structures, unions, operator overloading and templates as well. It has full pointer support.

1.4 Python:

Python was developed and designed in 1991 by Guido Van Rossum. It is an interpreted programming language. It supports object-oriented programming. Programs written in Python are short and easier to understand and read because it is a high-level programming language with minimal low-level functions. One major positive point of Python is that it has a huge library support making it highly efficient in almost all fields like Artificial Intelligence, Web Development and Data Science just to name a few. Since it’s an interpreted programming language, it is slower as compared to C and C++, but it supports features like Garbage collection and Dynamic typing. Python is also extremely popular due to being beginner friendly.

1.5 Java:

James Gosling begin the work on Java in 1991 and after a steady development and design procedure released it in 1996. Java is an object-oriented simple and portable programming language developed at the Sun Microsystems lab. Its syntax is equitable to C++ with some major differences in the language itself. It is both a compiled and interpreted programming language with support for garbage collection. Some of the features missing are the absence of structures and unions both of which are present in C++, support for pointers is also limited in Java. One more major difference is that it is a “Write Once Run Anywhere” language because of JVM support.

2 Methodology:

Since Programming is all about speed and efficiency, our focus in the paper will be on the quantitative side of things with a little sprinkle of qualitative evaluations and conclusions as well.
2.1 Language Versions:

| Compiler Version | C       | C++     | Java    | Python  |
|------------------|---------|---------|---------|---------|
|                  | TDM GCC 4.9.2 | TDM GCC 4.9.2 | Java SE 16 | Python 3.8 |

2.2 Hello World in Four Languages:

In Python:

```python
print("Hello World")
```

In Java:

```java
public static void main(String[] args) {
    System.out.println("Hello, World!");
}
```
A Pragmatic Comparison of Four Different Programming Languages

In C:

```c
#include <stdio.h>

int main()
{
    printf("Hello, World!");
    return 0;
}
```

In C++:

```cpp
#include <iostream>

int main()
{
    std::cout << "Hello World!";
    return 0;
}
```

As it can be witnessed from the above programs: C, C++ and Java’s hello world program has exactly 5 lines of code whereas Python’s hello world program has a mere single line. This simple example can point us in one direction that C, C++ and Java’s programs will become much longer the bigger the project is, whereas, Python’s programs will be significantly shorter than the rest of the languages discussed here.
2.3 Experiment 1:

This experimental code will dynamically assign memory and initialize the array and perform lots of int and float calculations, its only purpose is to perform mass calculation and the algorithm’s results are not useful for anything except for the purpose of measuring speed and efficiency. The experiment will be carried out on Windows 10 21H2 (Installed on SSD), with 8 GB of RAM running on Core i5 3rd gen 3570 with only essential background processes, so the results would not be interfered by anything else.

This experiment will compare the efficiency and speed based on the criteria of:

- Dynamic Memory Access (heap).
- Array item iteration.
- Int and float calculations.

### Running Time (In Milliseconds) Iteration Table: Experiment 1

| Language  | C     | C++   | Python | Java  |
|-----------|-------|-------|--------|-------|
| Iteration 1 | 324.6 | 365.1 | 2900   | 397   |
| Iteration 2 | 327.7 | 336.5 | 2900   | 390   |
| Iteration 3 | 325.7 | 332.3 | 2900   | 391   |
| Iteration 4 | 322.4 | 335.3 | 2900   | 394   |
| Iteration 5 | 320.4 | 341.7 | 2900   | 397   |
| Average    | 324.16 | 342.18 | 2900   | 393.8 |
## CPU Usage Iteration Table: Experiment 1

| Language | C  | C++ | Python | Java |
|----------|----|-----|--------|------|
| Iteration 1 | 19 | 26 | 41 | 40 |
| Iteration 2 | 18 | 24 | 43 | 33 |
| Iteration 3 | 19 | 25 | 38 | 40 |
| Iteration 4 | 19 | 27 | 38 | 37 |
| Iteration 5 | 18 | 28 | 37 | 35 |
| **Average** | **18.6** | **26** | **39.4** | **37** |

## Experiment 1 Observations

| Language | CPU Usage (AVG.) | No. of Lines | Execution Time (AVG.) (MS) |
|----------|------------------|--------------|---------------------------|
| C        | 18.6             | 24           | 324.16                    |
| C++      | 26               | 24           | 342.18                    |
| Python   | 39.4             | 23           | 2900                      |
| Java     | 37               | 21           | 393.8                     |
We ran the experiment exactly 5 times for each language and its program and then the average results are calculated and displayed in the above table. The experiment presents us with some interesting information, like Python had two more lines of code then Java, which made it the 2ND in our experiment on the number of lines and surprisingly Java had the least number of lines with C and C++ having an equal number of lines as expected.

C had the least CPU usage with C++ only falling behind it by about 8%. Python had the most CPU usage as expected because it’s interpreted whenever the code is running. Java on the other hand had about 2 percent less CPU usage then python.

The execution time presented us with a big disappointment from Python as it took a whopping 2900 MS (or 2.9 seconds) to perform the calculations, showing us that if speed is a factor (which is a huge factor) then you are better of to picking something else like C which had the least execution time with C++ on the 2ND place with only a negligible difference and once again Java sitting comfortably in the middle with about 69 MS more execution time then C which is still quite less as compared to Python.

From the experiment we can conclude that, if speed is of importance then C, C++ or Java should be chosen, however, if speed is not of importance and rather the handling and less complexity of code is the priority, then Python can be favored instead.
2.4 Experiment 2:

For experiment 2, we choose to do large matrix calculations. The size of the matrices was kept 1024 x 1024 and random values were assigned to them from 0.0 – 1.0 (the effect of random values has no effect on the results). The experiment was conducted 5 times for each language and the average was calculated. When the results for the languages were out, we were confused at first because Java’s program ran faster than C and C++, but then we realized that Java and Python’s compiler by default performs optimizations, keeping this in mind, we turned towards our C/C++ compiler and used the “-o3” optimization flag, and the results were quite shocking, the optimization flag not only gave the lead to C and C++, but it also widened the result’s gap by a large margin as seen from the Table:

| Language | C   | C++  | Python | Java  | C (-O3 optimizations) | C++ (-O3 optimizations) |
|----------|-----|------|--------|-------|-----------------------|-------------------------|
| Iteration 1 | 17.03 | 19.33 | 436.06 | 11.98 | 3.519 | 3.511 |
| Iteration 2 | 17.08 | 19.33 | 438.93 | 11.93 | 3.507 | 3.493 |
| Iteration 3 | 17.13 | 19.36 | 435.33 | 12.11 | 3.503 | 3.490 |
| Iteration 4 | 17.06 | 19.22 | 436.15 | 12.06 | 3.508 | 3.496 |
| Iteration 5 | 17.03 | 19.43 | 438.01 | 11.90 | 3.514 | 3.488 |
| Average | 17.06 | 19.33 | 436.89 | 11.99 | 3.510 | 3.495 |
## CPU Usage % Iteration Table: Experiment 1

| Language | C/C(-O3) | C++/C++(-O3) | Python | Java |
|----------|----------|--------------|--------|------|
| Iteration 1 | 29 | 29 | 29 | 30 |
| Iteration 2 | 29 | 29 | 29 | 30 |
| Iteration 3 | 29 | 29 | 29 | 30 |
| Iteration 4 | 29 | 29 | 29 | 30 |
| Iteration 5 | 29 | 29 | 29 | 30 |
| Average | 29 | 29 | 29 | 30 |
The result presented us with some interesting information, Python took most time and we had to sit and wonder what was happening around us, but the rest of the languages were quite competitive with each other, but however, C++ with its optimizations flags, took the victory by a huge margin from Java but C as it’s predecessor reminded close with negligible difference only.

The amount of difference -O3 made was quite surprising and was rather unexpected, we hoped it would shove off 5 or 6 seconds, but to our surprise it rather rounded of 15 seconds which is huge in the world of computing where even milliseconds of efficiency counts.
3 Conclusion:

From the above experiments, we can conclude that, if the goal is speed efficiency and reliability then C++ is the king of all the languages we compared, it takes the lead from C because it offers wider support of approaches because of supporting OOP which is absent in C. When developing system software like Drivers and Operating Systems, C still holds strong as compared to C++.

Java is the middle ground of all languages, if you can’t agree on the complexity of C but also want the speed that Python lacks then Java is your best bet because not only it offers “Write Once Run Anywhere” support it also automatically applies optimizations, something that a beginner programmer might not be familiar with.

Python is best used if you want to write software where speed is not of major concern. It is also a great choice for beginners and for fields like Data Science and Artificial Intelligence because of the huge number of libraries it offers.

4 Future Work:

In the future, we will perform more experiments on these languages as well as including newer languages. Since the field of computer science is continuously evolving, we have to evolve with it as well because it is the future of everything from medical to engineering, there is absolutely no field that cannot benefit from computing but some fields require higher accuracy and precision from the rest because many lives depend on them, like offering faster and less radiation induced X-Rays. There is nothing that won’t benefit from efficiency but for that we have to continuously design and write better and efficient code for our machines.

5 Acknowledgement:

We would like to offer thanks and respect to these persons or entities, without whom this research might not have been possible or would have been difficult.

- Our Parents for their continued support.
- our dearest teacher, Ma’am Mamona.
- Xara, for giving us tools to create the beautiful tables we used in this research.
- Gregory, for the Source codes used in Experiment No. 1.
- Gunavaran, for the Source codes used in Experiment No. 2.
References

[1] K. Aldrawiesh, A. Al-Ajlan, Y. Al-Saawy, and Abdullah Bajahzar. A comparative study between computer programming languages for developing distributed systems in web environment, 11 2009.

[2] Zakaria Alomari, Oualid Halimi, Kaushik Sivaprasad, and Chitrang Pandit. Comparative studies of six programming languages, 04 2015.

[3] Codelani. How many programming languages are there in the world?, Nov 2017.

[4] Ronald Garcia, Jaakko Järvi, Andrew Lumsdaine, Jeremy Siek, and Jeremiah Willcock. An extended comparative study of language support for generic programming, 03 2007.