Analyze the Evolution of Programming Languages

Yiqiao Yin
Ph.D. Student

Abstract

This report introduce the evolution of programming languages from switches and machine learning to high-level programming languages. Large corporations often have software solutions teams to develop the production code and these teams are usually separated based on development projects using languages such as Python, Java, and .NET. This report investigates several of these programming languages and discover their origins in the past.

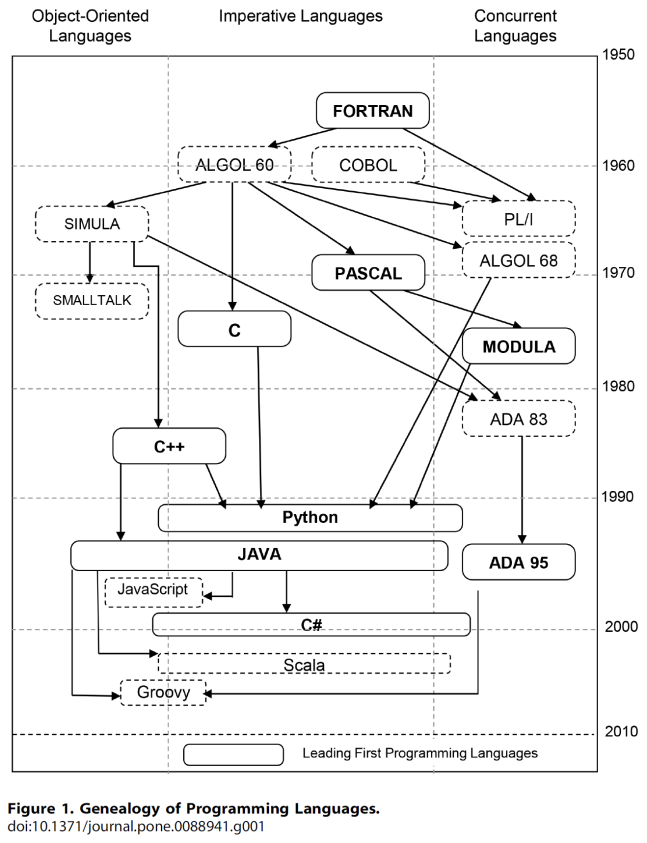
1 Introduction

The development of modern day Artificial Intelligence (AI) is largely based on the advancement of computer programming languages. In any computer science major courses from any universities, computer programming languages would be the first course to teach and sometimes this concept is spanned to multiple different classes with their own concentrations. This is where the First Programming Language (FPL) is usually introduced. When introducing FPL, it is also accompanied with the evolution of programming languages in different stages. This is because the pool of each field of programming languages have been developing based on the needs of different teams' across different functions. In the literature, there are many investigations propose different requirements to survey and evaluate computer programming languages Gupta (2004); Parker et al. (2006); McIver (2002) which is not concluded into any protocol or authoritative reference. Thus far, there is no definitive knowledge about the survey methods for such evaluation. This report starts with the work by Farooq et al. (2014) which proposes a granular framework to evaluate the present day object oriented languages in terms of the appropriateness each language is evaluated as an FPL and the report proposes a summary table to demonstrate a relationship diagram for the discussed FPLs.

2 Timeline

The introduction of the first programming language for any computer science directory is crucial and it is going to guide how the computer scientist think from a fundamental perspective. Hence, it is important to survey the First Programming Language (FPL) when it comes to study this subject and eventually move further to machine learning and

Figure 1: **Genealogy of Programming Languages**. The figure presents the genealogy of computer programming languages.



deep learning. Many studies have surveyed the First Programming Language (FPL) in the literature Siegfried et al. (2012); Pears et al. (2007); Davies et al. (2011).

The diagram of the evolutionary procedure for computer programming languages is summarized in Figure 1. This figure is originally presented in the work by Farooq et al. (2014) and it presents a time-series relationship diagram of a variety of different computer programming languages. For modern day education of computer programming, C#, C, Java, and Scala remain the most recent FPLs. In the old days 1990s, python was introduced and it has risen over the frequency of usage than Java and C-based languages in the past 20 years due to the advancement of machine learning. In the old days such as 1980s or even before, C has been popular FPL and, moreover, there were also FORTRAN, COBOL, PASCAL and so on.

It is easy to list and mention the most important first programming languages. However, it is challenging tasks to survey and evaluate these FPLs. The first reason is that the time has been changing since the first ever computer languages: FORTRAN. The tasks and the agendas for software engineers have also been changing too. Later on, deep neural networks were born and a whole wave of new jobs were developed. New responsibilities were assigned because there are not only software engineers but also machine learning engineers. The work by Farooq et al. (2014) developed a scoring function that is based on the flexibility of customization and ways of developing tuning parameters. Though the second reason sounds more intertwined with machine learning, it is definitely possible to use it as an overview for other field of programming needs as well.

2.1 Discussion of Each Generation of FPL

Though the evaluation of different FPL is difficult, it is possible to come up with a universal measure benchmark for all FPLs. The study by Farooq et al. (2014) proposed an evaluation framework. There are two aspects worth our notice: technical features, and environmental features.

The technical features refer to orthogonality, enforceability, security, effort level, and typing requirements. The environmental features refer to the demand in the industry which commonly consists of contemporary features, readability, coding quality, difficulty of transition, and level of easiness for integration.

2.1.1 FORTRAN

The first ever designed computer programming language is FORTRAN. It is designed for some basic machinery and mathematical operation. In early design of software development procedure, FORTRAN is heavily used in matrix operation specifically in the field of physics and some basic engineering pipelines.

The downside, although some may argue otherwise, is that FORTRAN has been changing ever since its original birth. This is because FORTRAN remains its popularity across the decades since its early development in the 1950s. The early stage of FORTRAN was out of date, yet is still being developed as the community moves forward.

A sample FORTRAN code to count from the integers 3 to 5 is listed below. As an example, the code initialize at an integer 3 and it is required to define the parameter. The operation is iterated and there is a "DO" command. This is not as elegant as some other programming languages developed later, but it does get the job done.

```
INTEGER, PARAMETER :: Init = 3, Final = 5
INTEGER :: Iteration

DO Iteration = Init, Final
   WRITE(*,*) 'Iteration ', Iteration
END DO
```

2.1.2 ALGOL60

ALGOL60 was developed in the 1960s and it was a big milestone in the early days of computer science. The programming language is more elegant and it was the successor of ALGOL58 which was initially developed in 1958. Today it is easy to handle data frames using C++ or Python, however, for ALGOL60 the earliest data types that can be operated was arrays.

A sample ALGOL60 code can be found below that is doing the exact same task as the one provided above for FORTRAN. Comparing with that of FORTRAN, it is quite elegant and much more easier to write.

```
for index := 3 step 1 until 5, index + 1 until 5 do
```

2.1.3 PASCAL

Before this report, I have not been very familiar with PASCAL. This report taught me, on some high levels, the basics of PASCAL. On top of ALGOL60, PASCAL is designed to be efficient and easy to write. The syntax is much more elegant in terms of design and it is a procedural programming language. PASCAL was born in early 1970s and it marks the end of an era and the beginning of another generation of computer programming languages. It is worth to mention that PASCAL is very strict. Though cumbersome, the strict design actually prevented some simple mistakes for programmers. In other words, the strict protocol in its syntax serves as a secondary pair of eyes watching over the validity of the code to ensure quality control. PASCAL has been existing for 50 years now and it has its unique role in the entire field of computer science, because, along with C, it is one of the earliest object-oriented programming languages.

The same task for FORTRAN and ALGOL60 to list the integers from 3 to 5 is written in PASCAL below. It is apparent that PASCAL takes a bit more work and effort to code. However, the "var" defines the variable and the "for" starts the for loop which are similar functionalities comparing with C, Java, and Python that we are familiar today.

```
program forLoop;
var
   a: integer;

begin
   for a := 3 to 5 do

   begin
      writeln('value of a: ', a);
   end;
end.
```

2.1.4 C

As one of the oldest and the most important programming language, C programming language carries certain importance amongst its peers. The most fundamental role C programming language is playing, which I also appreciate, is that C is the building block for many other computer languages. The backend of R is coded in C/C++ and the backend of Python is also supported by C-based languages. C is extremely powerful and efficient, much more efficient than many of its predecessors. In addition, C is flexible and C can be loaded on different machines without breaking the code or malfunction.

There are also downside for C as well. The C programming language is compiled and ran. The errors will not present themselves until the program finishes running. This invites many questions and makes the lives of many software programmers extremely difficult especially when they are dealing with large-scale software programs.

A sample C code is presented below solving the exact same task: listing integers from digit 3 to digit 5. The code looks more familiar to Java, Python, and R than it did before with FORTRAN, ALGOL60, and PASCAL. The code looks more efficient and the syntax is more intuitive towards the logical expression.

```
// Print numbers from 3 to 5
#include <stdio.h>

int main() {
  int i;

  for (i = 3; i <= 5; ++i)
  {
    printf("%d ", i);
  }
  return 0;
}
```

2.1.5 C++

C++, without any doubt, is the most famous and fundamental programming languages every born. It is extremely efficient. Some may say C++ is their go-to computer language and it is easily ranked on the top of the list of FPLs. The portability and low-level manipulation are two of its strongest suits for C++. In addition, it is object oriented and it is extremely easy to manipulate. The concept of C++ (or even C) is scalable and can be developed into many other computer programming languages. It is the root language for machine learning and statistical languages such as Python and R.

Though there is very minimal downside to C++, one big drawback is lack of security. This is because C++ is extremely programming friendly and very comfortable to read which is a big trade-off in computer programming languages.

A sample code for C++ doing the same task is presented below.

```
#include <iostream>
using namespace std;

int main() {
  for (int i = 3; i <= 5; i++) {
    cout << i << "\n";
  }
  return 0;
}
```

2.1.6 Python

Python is the most famous machine learning and software engineer language today. It holds the highest esteem and allows software engineers and machine learning scientist to do anything where at anytime. In the old days, Python has solely been used for web-based application development. However, due to the development of machine learning, it has quickly risen up and earn its place. It is the most flexible computer programming language comparing with all the ones listed before and it is probably the most commonly known computer programming language for data science. It also has web-based application django to communicate with when the task is to build web-based applications. Python has becoming a programming language that you “MUST” learn today.

The same task above is coded using Python and the code is presented below:

```
for i in [3, 4, 5]:
    print(i)
```

Another way to do the same task is the following

```
[i for i in [3, 4, 5]]
```

2.1.7 Java

If C and C++ sets the milestone in the two decades from 1970 to 1980, then Java (along with Python) sets the building block for 1990. Java is extreme famous, and almost as famous as Python. From machine learning and data science perspective, Python would probably be more popular than Java. However, for other software engineering work Java can outperform Python in some scenarios.

Java is simple to use and it is an object-oriented language, just like Python. Unlike C or C++, the structure of Java is secure since Java uses object-oriented programming (OOP) concepts. Java is less costly and easy to manage. In addition, it is very simple to tailor down a large-scale production into small and manageable level for software engineers using Java, a trait that is not easily come by for other programming languages. There are plenty of downside for Java. It can be very slow if the time component is loosely written in a software program. The graphical user interface (GUI) is less ideal and can be less tactic to read off the syntax. It can uses significant amount of memory just like C or C++ though sometimes scholars may argue Java takes up more space comparing with other programming languages.

The code of the same task is presented below using Java

```
public class Main {
  public static void main(String[] args) {
    for (int i = 3; i <= 5; i++) {
      System.out.println(i);
    }
  }
}
```

2.1.8 JavaScript

JavaScript serves less amount of importance across the entire development of computer programming languages. However, for each generation, the survey requires to review three languages and this is probably one big reason JavaScript (or JS) made to the list.

JavaScript is most famous in web design. The language is easy to use and it does not invite a lot of questions. The language is interpreted so one big benefit is the speed. The interpretation process is less costly than its peers. Its simplicity is what gains its popularity.

One big drawback is the security problem for JavaScript. For example, any user can right click a web page and inspect the page source. The browser will output the JavaScript component along with HTML code. The source code is directly viewable from the client-side and the information is transparent. This may serve well for learning purpose, but it renders its content vulnerable. It is also quite difficult to debug in JavaScript as well. It is very rare to see JavaScript exists by itself. It is almost always embedded inside HTML.

The task of listing integers between 3 and 5 is coded below using JavaScript.

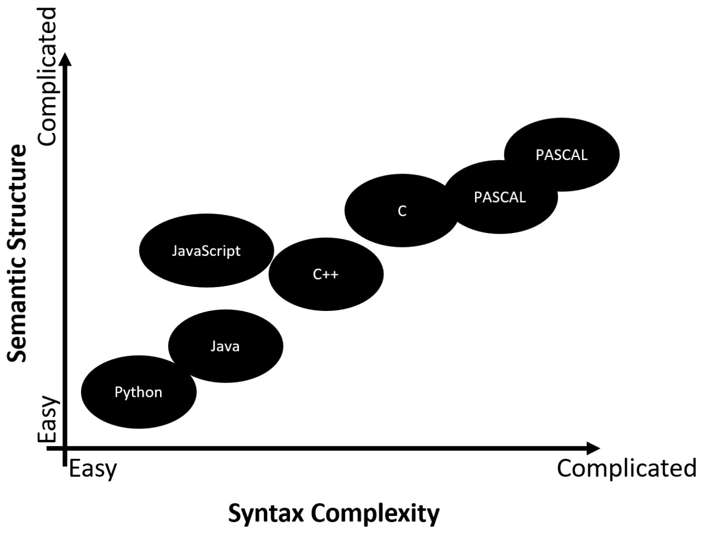
```
<html>
<body>
```

```

<h2>JavaScript For Loop</h2>

<p id="demo"></p>

<script>

let text = "";
for (let i = 3; i <= 5; i++) {
  text += i + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

2.2 Comparison Amongst Different FPLs

This subsection summarizes the different programming languages according to semantic structure and syntax complexity. These two metrics are discussed and used to help distinguish the major difference among different FPLs. The relationships are summarized in Figure 2. The syntax complexity and semantic structure can be ranged from "easy" to "difficult".

Figure 2: **General Matrix**. This is the relationship matrix of different FPLs compared in this report.



3 Conclusion

This report surveys the evolution of the past 50 years of development of computer programming languages. The report summarizes each generation of FPL and covered eight important computer programming languages. The report further investigates the benefits and drawbacks of each programming language and a summary table is provided in the report to present the relationship between semantic structure and syntax complexity for each language.

References

Davies, S., Polack-Wahl, J. A., and Anewalt, K. (2011). A snapshot of current practices in teaching the introductory programming sequence. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 625–630.

Farooq, M. S., Khan, S. A., Ahmad, F., Islam, S., and Abid, A. (2014). An evaluation framework and comparative analysis of the widely used first programming languages. *PloS one*, 9(2):e88941.

Gupta, D. (2004). What is a good first programming language? *Crossroads*, 10(4):7–7.

McIver, L. (2002). Evaluating languages and environments for novice programmers. In *PPIG*, page 10. Citeseer.

Parker, K. R., Chao, J. T., Ottaway, T. A., and Chang, J. (2006). A formal language selection process for introductory programming courses. *Journal of Information Technology Education: Research*, 5(1):133–151.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., and Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *Working group reports on ITiCSE on Innovation and technology in computer science education*, pages 204–223.

Siegfried, R. M., Greco, D., Miceli, N., and Siegfried, J. (2012). Whatever happened to richard reid's list of first programming languages? *Information Systems Education Journal*, 10(4):24.