Which Programming Languages Should One Learn?*

Kshitij Sudan

Abstract

During my undergraduate years, especially the initial ones, I used to wonder what programing language would be the best one to master. I tried my hand at various languages when during my final few months at college, the reality dawned on me. Thanks, in part, to my faith in Open Source and, in part, to a good friend, I realized that there is no such thing as the perfect language. If you are facing a similar dilemma then I hope this short article will be a guiding post.

1 Introduction

Every once in a while and that too, too often, somebody confounds me by asking a seemingly innocuous question:

> What programming language should I learn?

I am not only clueless about the answer to this simple question, but lately I have begun to be irritated by the question itself. The reason is for being clueless is simple: *There is no, one perfect language which one **"should"** master.* The reason for being irritated is more involved and is described in later sections.

But not all hope is lost. There are three aspects critical to this debate in my view and these should be your guiding oracles while you decide which language to learn and more importantly ***when!***

2 Programming is a means, not an end.

It is very strange to observe the fact that people always forget this very critical fact. Programming is a method which is used to *" tell"* computer the problem it's supposed to solve, and thus, by extension, programming in itself is not a "problem". So my advice is to first worry about the **problem** and then choose the perfect *tool* to encode that problem into a computing system. While choosing this tool for a particular problem the guiding post is **efficiency**.

*This document was typeset in LaTeX 2ε using GNU-Emacs by the author.

Generally, people tend to ignore this facet and believe that learning a particular language "really well" (whatever that means!) is the holy grail of their careers in computer science. They obviously are misguided souls! I am sure people like Paul Graham [1] were successful not only because they were great programmers, but also because they use perfect tools for the problem at hand.

3 What language is the "best-est" language to learn.

You need to be a fool to be asking this question ever in your life, even if you are on your deathbed. There is no one "perfect" language as there is no one "perfect, do-it-all" tool for any trade (carpentry, masonry etc. for e.g.). The tool depends on the task (a.k.a the problem) at hand. Choose the best language fit for the task and never force a tool do work which it was not designed for. Imagine using a jack hammer when all you need is a mallet.

There is an interesting corollary to this point. All those who think that they can get away by learning only one programming language during their careers, can expect a shock somewhere down the line. The reasons are obvious. However, don't construe this as a ratification to learn every possible language that ever existed.

A good, healthy and most importantly, correct perspective on this issue is brought forth by Steve Yegge in the following article :
http://steve.yegge.googlepages.com/tour-de-babel
*This article is a must read. If you skip this web link, the current article loses it's relevance as this is actually an extension of ideas built on by Yegge.**So you must read it!***

4 What platform is most suitable to learn programming languages?

If you are still in undergraduate college and exploring various languages, then the following is what the doctor recommends in terms of computing platform for all your programming needs:

1. *NIX O.S.
2. Emacs Editor. **(Emacs is the 100-year editor.)**
3. gcc compiler suite.

[1] Graham wrote the web program to build on line stores in LISP and C which later was acquired by Yahoo! Read more about it here: http://www.paulgraham.com/avg.html

Of course you would need compilers/interpreters for the languages you are learning if they are not available in the gcc suite!

5 Pitfalls

After Reading this article, please do not rush off to the nearest bookstore to pick up books on LISP, C, Perl, Ruby, Python etc.

Remember that first you need a problem to solve. I have always found it useful to know a tad bit about most of the major programming languages (and their strengths and weaknesses) in most application domains. That enables me to later make an informed decision about what language to use for a particular project. Thus my suggestion would be to just know enough (initially) to make this choice and then learn the language as you go when you "code" that project.

6 Conclusion

Always remember (and this holds true even for natural languages), the language is not important, the concept is. If you are talking to a Frenchman, you can communicate your thoughts even without knowing French (although very, very inefficiently). It's the same with computers, you can use Java to write an OS but C will help you write an efficient OS. But in the end what's more important is that you are writing an OS (i.e. you have a problem to solve) not reversing a linked list (i.e. you are not learning the language, such exercises are what they are, exercises to teach you a language, not the problem in itself!).