1. Introduction

Why Learn This Stuff?

- **Little Languages are everywhere :** Successful general-purpose programming languages often coexist with numerous niche languages, previously referred to as “little languages” but now known as “domain-specific languages.” These languages are specifically designed for particular tasks, such as application scripting, template engines, markup formats, and configuration files. In large software projects, it is beneficial to reuse existing languages rather than creating new ones due to the complexities involved in documentation, debugging, and support. However, developers may still need to create custom parsers or tools when suitable libraries are unavailable, necessitating ongoing debugging and maintenance of these implementations.
- **Languages are great exercise:** Long distance runners often enhance their training by using weights or training at high altitudes, which improves their performance when they return to normal conditions. In programming, implementing a language serves as a rigorous test of one’s skills, requiring a deep understanding of complex concepts such as recursion, dynamic arrays, trees, graphs, and hash tables. While hash tables may be familiar to many programmers, mastering their implementation from scratch can deepen one's comprehension. The process of creating an interpreter is challenging but ultimately rewarding, leading to improved programming abilities and a better grasp of data structures and algorithms.
- **One more reason:** The author reflects on their lifelong fascination with programming languages, describing how this interest was initially accompanied by feelings of awe and intimidation. They felt excluded from the "wizard-like" community of language developers, believing they lacked an innate quality to join them. However, upon finally attempting to create their own interpreters, they discovered that programming languages are simply code, and that the barriers to learning them are surmountable. The author hopes that their insights will empower others who feel intimidated by programming languages to embrace the challenge and perhaps even create their own.

Importent Notes:

A compiler processes files written in one language, translates them, and produces files in a different language. You can create a compiler in any programming language, including the same language it compiles, a method referred to as “self-hosting.” While you cannot compile your own compiler with itself initially, if you have another compiler for your language that is developed in a different language, you can use that to compile your compiler one time. After that, you can utilize the compiled version of your own compiler to create future iterations of itself, allowing you to eliminate the original version that was compiled by the other compiler. This process is known as “bootstrapping,” akin to the idea of lifting yourself up by your own bootstraps.