2025

CS 4402

Comparative Programming Languages

LEARNING JOURNAL 5

SANA UR REHMAN

INSTRUCTOR: JIM CASALE

Introduction

This week's focus on scripting languages allowed me to explore the distinctions between traditional programming and scripting paradigms, while also deepening my understanding of languages like Perl, Python, and JavaScript. Through readings, self-quizzes, and a discussion comparing PHP and JavaScript, I strengthened my grasp of how scripting languages function across various environments. The week emphasized practical coding exercises, regular expressions, and language evaluation all of which built upon my analytical and problem-solving skills.

What I Did and How I Did It

I began by reviewing the module readings on the history, types, and purposes of scripting languages. I practiced simple Perl and Python scripts that performed text processing and file handling tasks to experience their syntax differences. The quizzes tested key ideas such as interpreted execution, portability, and the role of regular expressions in data manipulation. My discussion post compared PHP and JavaScript, focusing on runtime behavior, performance, and adoption trends. This helped me synthesize theoretical understanding with real-world applications.

When exploring regular expressions, I used Python's `re` module to extract patterns from log files. Initially, I found the syntax complex, but after experimenting with quantifiers and character classes, I began to see how powerful and compact pattern matching can be. I also explored JavaScript's regular expression evaluator in the browser console, which reinforced the concept of scripting for dynamic client-side tasks.

Reactions and Feedback

At first, I underestimated scripting languages as “lightweight” tools compared to compiled programming languages. However, I was surprised by their versatility and integration in modern development environments (Zandstra, 2020). Feedback from my discussion post noted that my comparison of concurrency models between PHP and JavaScript effectively clarified their different strengths. This encouraged me to dig deeper into asynchronous programming patterns and how event loops handle concurrent I/O in JavaScript.

Feelings and Attitudes

Initially, I felt slightly overwhelmed by Perl’s syntax and the dense nature of regular expressions. Over time, I developed confidence as I recognized recurring logic patterns across languages. I also grew more appreciative of scripting languages’ flexibility. Unlike compiled languages that demand rigid structures, scripting allows rapid testing and immediate feedback, which matches my learning style. I now see scripting as a creative playground for automating repetitive tasks and connecting diverse systems.

What I Learned

This week clarified that scripting languages are not merely simpler programming tools but essential for efficiency in automation, data processing, and web development. I learned that scripting emphasizes interpretation and runtime flexibility rather than compilation, which makes it ideal for quick prototyping and integration (Sebesta, 2016). I also recognized that Python’s

readability and vast library ecosystem make it the most adaptable general-purpose scripting language, while JavaScript remains unmatched for browser-based interactivity.

Regular expressions, though challenging, proved to be powerful mechanisms for parsing, validating, and transforming text. Their integration in scripting languages reinforces how developers can handle data dynamically without verbose code.

Reflection and Application

What surprised me most was realizing how scripting languages blur the boundary between client and server logic especially through Node.js, which enables full-stack JavaScript applications. The most challenging part was internalizing the subtle differences in how scripting engines interpret code at runtime. This required me to think less about compilation steps and more about runtime evaluation.

Through this unit, I recognize my growing ability to analyze languages critically rather than memorizing syntax. As a learner, I am discovering that my curiosity deepens when I experiment and test concepts interactively. Moving forward, I plan to apply scripting to automate repetitive file-processing tasks and build small prototypes quickly before implementing them in larger software projects.

One key takeaway is that scripting languages embody the principle of *doing more with less* achieving functionality through concise, readable, and adaptable code. This realization will shape how I approach software development, focusing on flexibility and clarity rather than complexity.

REFERENCES

Sebesta, R. W. (2016). *Concepts of programming languages* (11th ed.). Pearson Education. https://www.cs.csubak.edu/~jyang/Robert%20W.%20Sebesta%20-%20Concepts%20of%20Programming%20Languages-Pearson%20(2015).pdf

Zandstra, M. (2020). *PHP objects, patterns, and practice* (6th ed.). Apress. https://ftp.zhirov.kz/books/IT/PHP/PHP%208%20Objects%2C%20Patterns%2C%20and%20Practice%20%28Matt%20Zandstra%29.pdf

Word count: *630*