CS 4402-01 Comparative Programming Languages

Instructor: Professor Rajesh Soni

Name: Ryohei Hayashi

Learning Journal 5

**Overview of the Week**

This week, I studied scripting languages, focusing on their characteristics, use cases, and differences from traditional programming languages. The topics covered included:

* Overview and survey of scripting languages
* Introduction to scripting
  + Unix scripting language: Perl
  + Web-centric scripting language: Python
  + Browser scripting language: JavaScript
* Regular expressions

Additionally, the discussion assignment required me to compare PHP and JavaScript, analyzing their performance, functionality, adoption, and application areas. The readings, such as "Scripting: Higher Level Programming for the 21st Century" by John K. Ousterhout and Competitors to Java: Scripting Languages by Judith Bishop, further supported the learning process.

**Personal Reflections**

This week’s learning made me reflect deeply on the differences between scripting and programming languages. Previously, I saw both as “writing code,” but I now understand that scripting languages excel at specific tasks and simple automation. By writing small scripts in Perl and Python, I realized their efficiency and versatility in solving practical problems.

Python stood out as a powerful scripting tool due to its simple syntax and speed. For example, I successfully processed text data and extracted specific patterns in just a few lines of code. This experience helped me appreciate scripting languages' strengths: flexibility and immediacy.

On the other hand, learning regular expressions initially felt challenging. Writing complex patterns in Perl and Python required time and practice. However, through repeated exercises, I recognized how powerful regular expressions are for text processing. Reflecting on situations where I struggled with data organization in the past, I thought, “If only I had learned regular expressions earlier!”

The discussion assignment comparing PHP and JavaScript was another enlightening experience. PHP excels at generating dynamic server-side content, while JavaScript dominates the frontend for creating interactive features. I was surprised to learn that JavaScript, with the introduction of Node.js, is now also used for server-side development. This realization highlighted the importance of selecting the right tool for the right task.

Overall, this week helped me understand scripting languages as more than “simple code.” They are efficient tools for automation, problem-solving, and managing complex tasks. I feel motivated to continue improving my scripting skills and leveraging them to solve real-world challenges effectively.

**Topics Studied in Depth**

Two key areas stood out during my learning this week:

1. **Understanding and applying regular expressions**

Regular expressions are powerful for text processing and data filtering. By practicing with Perl’s `grep` and Python’s `re` library, I created scripts for pattern matching and data extraction. While initially difficult, I gradually became comfortable using expressions to simplify complex tasks.

1. **Comparing PHP and JavaScript**

Through research, I identified that PHP is primarily suited for backend tasks like dynamic content generation and database integration, while JavaScript is optimal for client-side interactivity. However, JavaScript’s adoption in server-side development via Node.js surprised me. This comparison helped me understand the importance of choosing languages based on their strengths and application areas.

**Future Challenges and Goals for Next Week**

This week’s exploration of scripting languages provided valuable insights into text processing and automation. However, I still need to practice writing more complex regular expressions and further explore practical applications of PHP and JavaScript.

Next week’s focus on Object-Oriented Programming (OOP) will provide a deeper understanding of structured programming. Key areas I aim to explore include:

* Encapsulation, inheritance, and dynamic polymorphism
* Structuring classes, including abstract classes, generics, and multiple inheritance
* Understanding class data, constructors, destructors, and garbage collection

I plan to approach OOP concepts with the same hands-on mindset, writing code and comparing procedural and object-oriented paradigms to enhance my programming skills further.

Word Count: 583

References

1. Ben-Ari, M. (2006). *Understanding programming languages.* Weizman Institute of Science.