HW1: Programming Language Comparison Essay

Organization of Programming
Due: End of Week 2
Points: 100 points
Learning Outcomes: 1, 8

Assignment Overview

Write a comparative analysis essay examining three programming languages to develop your skills in language assessment and critical analysis. This assignment will help you understand how to systematically evaluate programming languages and make informed decisions about language selection.

Choose 3 languages from the following list:

- Python
- C++
- JavaScript
- Java
- C
- Go
- Rust

Requirement: Your three chosen languages must represent different programming paradigms or have significantly different design philosophies.

Essay Structure and Requirements

Write a **4-5 page essay** (double-spaced, 12pt font) with the following sections:

Section 1: Introduction (15 points)

Length: ~1 page

Content:

- Introduce the three languages you selected and briefly explain why you chose them
- State your thesis: What is the main argument or insight about these languages that your essay will demonstrate?
- Preview the comparison criteria you'll use
- Explain why comparing these languages is valuable for programmers

Example thesis: "While Python prioritizes developer productivity, C++ emphasizes performance control, and Haskell promotes mathematical correctness, each language's design reflects different priorities that make them suitable for distinct problem domains."

Section 2: Language Profiles (45 points - 15 points each)

Length: ~1 page per language

For each language, address the following:

Design Philosophy and History

- What problem was this language designed to solve?
- What are the core design principles?
- Brief historical context (when/why was it created?)

Key Characteristics

- **Type System:** Static/dynamic, strong/weak, type inference capabilities
- **Memory Management:** How does the language handle memory allocation/deallocation?
- **Primary Paradigm:** Imperative, object-oriented, functional, etc.
- **Execution Model:** Compiled, interpreted, or hybrid

Strengths and Ideal Use Cases

- What does this language excel at?
- What types of projects is it best suited for?
- What makes it attractive to developers?

Limitations and Challenges

- What are the language's main weaknesses?
- When would you NOT choose this language?
- What are common criticisms?

Include: One simple code example (3-5 lines) that illustrates something distinctive about the language.

Section 3: Comparative Analysis (30 points)

Length: ~2 pages

Choose 2-3 comparison themes from the following and analyze how your languages differ:

Theme Options:

- **Performance vs. Productivity:** How do the languages balance execution speed with development speed?
- **Safety vs. Control:** How do the languages handle memory safety, type safety, and programmer control?
- **Simplicity vs. Power:** How do the languages balance ease of learning with expressive capability?
- **Abstraction Levels:** How do the languages support different levels of abstraction?
- **Error Handling:** How do the languages approach error detection and recovery?

For each theme:

1. Explain why this comparison is important
2. Analyze how each language approaches this aspect
3. Provide specific examples or evidence
4. Discuss the trade-offs involved

Section 4: Practical Decision-Making (10 points)

Length: ~1/2 page

Scenario-Based Analysis: Present a specific programming scenario (e.g., "building a web application for a startup," "developing a real-time game engine," "creating a data analysis tool for scientists") and explain:

- Which of your three languages you would choose
- Your reasoning based on the analysis in previous sections
- What factors were most important in your decision
- What trade-offs you're accepting with this choice

Technical Requirements

Research and Sources

- **Minimum 4 credible sources** (official documentation, books, academic papers, reputable industry articles)
- **Hands-on exploration:** Write and run at least one small program in each language
- Use sources from the last 10 years when possible (language design evolves)

Code Examples

- Include 1 working code example per language (3-5 lines each)
- Examples should compile/run correctly
- Add brief comments explaining what the code demonstrates
- Focus on illustrating key language characteristics

Writing and Formatting

- **Length:** 4-5 pages (not including code examples or references)
- **Format:** Double-spaced, 12pt Times New Roman, 1-inch margins
- **Citations:** Use a consistent citation style (APA, MLA, or IEEE)
- **File:** Submit as PDF

Deliverables

Submit the following:

- [ ] `HW1_Essay_[YourLastName].pdf` - Your complete essay
- [ ] `code_examples.zip` - Folder containing your code examples organized by language
- [ ] Include a brief bibliography/references section at the end of your essay

---

Grading Rubric (100 points total)

Introduction (15 points)

Excellent (14-15)	**Good (12-13)**	**Satisfactory (9-11)**	**Needs Improvement (0-8)**
Clear thesis, engaging introduction, languages well-motivated	Good thesis, adequate introduction	Weak thesis or unclear introduction	Missing thesis or poor introduction

Language Profiles (45 points - 15 each)

Excellent (14-15)	Good (12-13)	Satisfactory (9-11)	Needs Improvement (0-8)
Comprehensive, accurate, insightful analysis of each language	Good coverage with minor gaps	Basic information provided	Significant gaps or inaccuracies

Comparative Analysis (30 points)

Excellent (27-30)	Good (24-26)	Satisfactory (18-23)	Needs Improvement (0-17)
Thoughtful comparison with clear themes, strong evidence, good examples	Good comparison with some insights	Basic comparison, obvious points	Weak or superficial comparison

Practical Application (10 points)

Excellent (9-10)	Good (7-8)	Satisfactory (5-6)	Needs Improvement (0-4)
Realistic scenario, well-reasoned choice, clear trade-off awareness	Good scenario and reasoning	Basic scenario analysis	Weak or unrealistic analysis

---

Getting Started Tips

Week 1 Tasks

1. **Choose your three languages** - pick languages you're curious about or want to understand better
2. **Set up basic environments** for each language (use online compilers if needed)
3. **Write simple "Hello World" programs** to get familiar with basic syntax
4. **Begin research** using official documentation and course readings

Week 2 Tasks

1. **Write your language profiles** - one per day works well
2. **Develop your comparative analysis** focusing on 2-3 clear themes
3. **Create your practical scenario** and decision analysis
4. **Revise and proofread** before submission

Recommended Research Strategy

1. Start with official language documentation for authoritative information
2. Look for "language comparison" articles that include your chosen languages
3. Check recent developer surveys (Stack Overflow, GitHub) for usage trends
4. Read blog posts by experienced developers who've used multiple languages

---

Example Language Combinations

Good combinations (representing different approaches):

- **Python, C++, Haskell:** High-level vs. low-level vs. functional
- **JavaScript, Java, Go:** Dynamic vs. enterprise vs. systems programming
- **Python, Rust, C:** Productivity vs. safety vs. performance

Avoid combinations like:

- Java, C#, Scala (too similar in design philosophy)
- Only scripting languages (Python, JavaScript, Ruby)
- Only systems languages (C, C++, Rust)

Academic Integrity

This is an individual assignment. While you may discuss general concepts with classmates, your analysis and writing must be entirely your own.

Acceptable:

- Discussing which languages to compare
- Sharing resource recommendations
- Getting help with environment setup

Not acceptable:

- Sharing essay drafts or outlines
- Using someone else's code examples without attribution
- Copying analysis from online sources

Cite all sources and ensure any code you reference is properly attributed.

FAQ

Q: Can I compare languages I already know well? A: Yes, but challenge yourself to learn something new about them. The goal is developing analytical skills, not just describing familiar languages.

Q: What if I can't get a language running locally? A: Online compilers/interpreters (like repl.it, CodePen, or language-specific online tools) are acceptable for this assignment.

Q: How technical should my analysis be? A: Focus on concepts rather than syntax details. You're analyzing language design decisions and their implications, not teaching someone how to program.

Q: Can I go over the page limit? A: Stay within 4-5 pages for the main essay. Code examples and references don't count toward the limit.

This assignment will help you develop the critical thinking skills needed to make informed language choices throughout your programming career. Focus on understanding the "why" behind language design decisions!