Chapter 1

Reasons to study concepts of PLs

1. Increased capacity to express programming concepts

2. Improved background for choosing appropriate languages

3. Increased ability to learn new languages

4. Understanding the significance of implementation

5. Increased ability to design new languages

6. Overall advancement of computing

Programming Domains

1. Scientific applications
2. Business applications
3. Artificial intelligence
4. Systems programming
5. Scripting languages
6. Special purpose languages

Chapter 1

Language Evaluation Criteria

1. *Readability*
    - The most important criterium
    - *Factors:*
        - Overall simplicity
            - Too many features is bad
            - Multiplicity of features is bad
        - Orthogonality
            - Makes the language easy to learn and read
            - Meaning is context independent
        - Control statements
        - Data type and structures
        - Syntax considerations

2. *Writability*
    - *Factors:*
        - Simplicity and orthogonality
        - Support for abstraction
        - Expressivity

3. *Reliability*
    - *Factors:*
        - Type checking
        - Exception handling
        - Aliasing
        - Readability and writability

Chapter 1

Evaluation criteria (continued)

4. *Cost*
   - *Categories*
     - Programmer training
     - Software creation
     - Compilation
     - Execution
     - Compiler cost
     - Poor reliability
     - Maintenance
5. *Others:* portability, generality, well-definedness

Primary influences on language design

1. *Computer architecture*
   - We use imperative languages, at least in part, because we use von Neumann machines
2. *Programming methodologies*
   - *1950s and early 1960s:* Simple applications; worry about machine efficiency
   - *Late 1960s:* People efficiency became important; readability, better control structures
   - *Late 1970s:* Data abstraction
   - *Middle 1980s:* Object-oriented programming

Chapter 1

Language Categories

1. Imperative
2. Functional
3. Logic
4. Object-oriented (closely related to imperative)

Language Design Trade-offs

1. Reliability versus cost of execution

2. Writability versus readability

3. Flexibility versus safety

Implementation Methods

1. Compilation
   - Translate high-level program to machine code
   - Slow translation
   - Fast execution

Chapter 1

2. Pure interpretation
   - No translation
   - Slow execution
   - Becoming rare

3. Hybrid implementation systems
   - Small translation cost
   - Medium execution speed

Programming Environments

-The collection of tools used in software development

1. UNIX
   - An old operating system and tool collection

2. Borland C++
   - A PC environment for C and C++

3. Smalltalk
   - A language processor/environment

4. Microsoft Visual C++
   - A large, complex visual environment