Introduction to Programming Languages

Programming Language

It is used for communicating instructions to a computer. It is designed to facilitate certain operations such as numerical computation, text manipulation, or input/output.

Reasons for Studying Concepts of Programming Languages

- Increased capacity to express ideas
- Improved background for choosing appropriate language
- Increased ability to learn new languages
- Better understanding of the significance of implementation
- Better use of languages that are already known
- Overall advancement of computing

Programming Domains

- Scientific Applications
- Business Applications
- Artificial Intelligence
- Systems Programming
- Web Software

Language Evaluation Criteria

- Readability – the ease with which programs can be read or understood
- Writability – how easily a language can be used to create programs
- Reliability – if a program performs to its specifications under all conditions
- Cost

Characteristics that Affect Readability

- Overall Simplicity: A language with a large number of basic constructs is more difficult to learn than one with a smaller number.
- Orthogonality – means that a relatively small set of primitive constructs can be combined in a relatively small number of ways to build the control and data structures of the language
- Data Types
- Syntax Design
    - Special Words
    - Form and Meaning

Characteristics that Affect Writability

- Simplicity and Orthogonality
- Support for Abstraction: Abstraction is the ability to define and then use complicated structures or operations in ways that allow many of the details to be ignored.
    - Process (Ex. The use of a subprogram to implement a sort algorithm that is required several times in a program
    - Data (Ex. A binary tree that stores integer data in its nodes can be implemented using an abstraction of a tree node in the form of a simple class with two pointers and an integer
- Expressivity – means that a language has relatively convenient ways of specifying computation.

Characteristics that Affect Reliability

- Type Checking – testing for type errors in a given program, either by the compiler or during program execution
- Exception Handling – the ability of a program to intercept run-time errors (as well as other unusual
- conditions detectable by the program), take corrective measures, and then continue
- Aliasing – having two or more distinct names that can be used to access the same memory cell
- Readability and Writability

Characteristic	Readability	Writability	Reliability
Simplicity	✓	✓	✓
Orthogonality	✓	✓	✓
Data Types	✓	✓	✓
Syntax Design	✓	✓	✓
Support for Abstraction		✓	✓
Expressivity		✓	✓
Type Checking			✓
Exception Handling			✓
Aliasing			✓

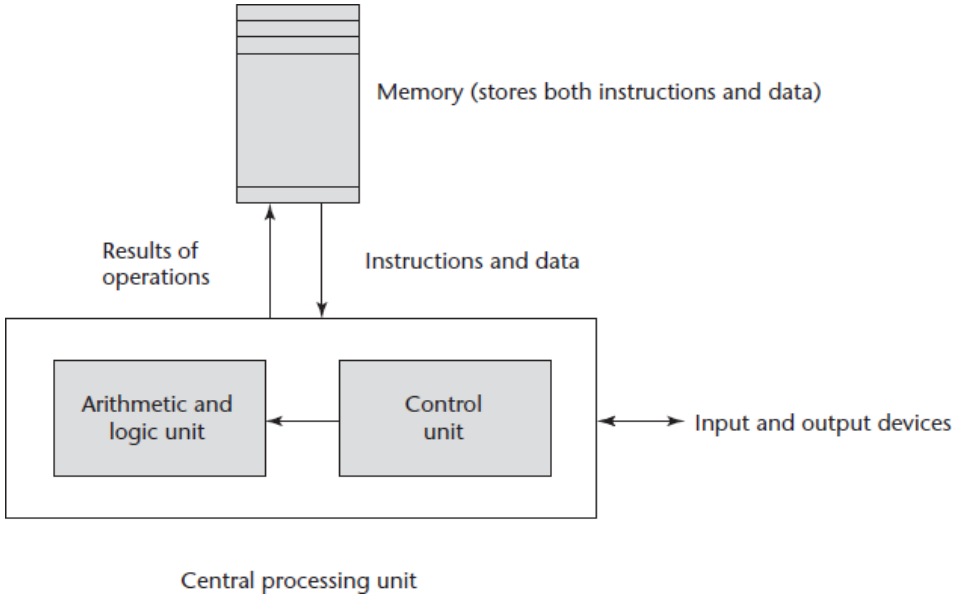
Language Evaluation Criteria

Source: Sebesta, 2012

Influences on Language Design

- **Computer Architecture**

  Most of the popular languages of the past 50 years have been designed around the prevalent computer architecture, called the von Neumann architecture, after one of its originators, John von Neumann (pronounced "von Noyman"). These languages are called imperative languages.



The von Neumann Architecture

Source: Sebesta, 2012

- **Programming Design Methodologies**: The evolutionary steps in software development methodologies led to new language constructs to support them. The most familiar design methodologies are procedural and object-oriented programming.

Language Categories

- Imperative Language
- Functional Language
- Logic language
- Object-Oriented Language

Imperative Language

- Based on commands that update variables in storage
- Provides statements such as assignment statements, which explicitly change the state of the memory of the computer

- Examples: Algol, Cobol, PL1, Ada, C, Modula-3

Functional Language

- Expresses computations as the evaluation of mathematical functions
- Treats values as single entities
- Values are never modified.
- Their computations are performed largely by applying functions to values.
- Examples: Lisp, Haskell, ML, Miranda, APL

Logic Language

- Expresses computations in exclusively in terms of mathematical logic
- Focuses on predicate logic, on which the basic concept is a relation
- Useful for expressing problems where it is not obvious what the functions should be
- Example: Prolog

Object-Oriented Language

- The fundamental of OOP was characterized by Alan Kay as follows:
    - Everything is modeled as an object.
    - Computation is performed by message passing.
    - Objects communicate with one another via message passing.
    - Every object is an instance of a class where the class represents a group of similar objects.
    - Inheritance defines the relationships between classes.

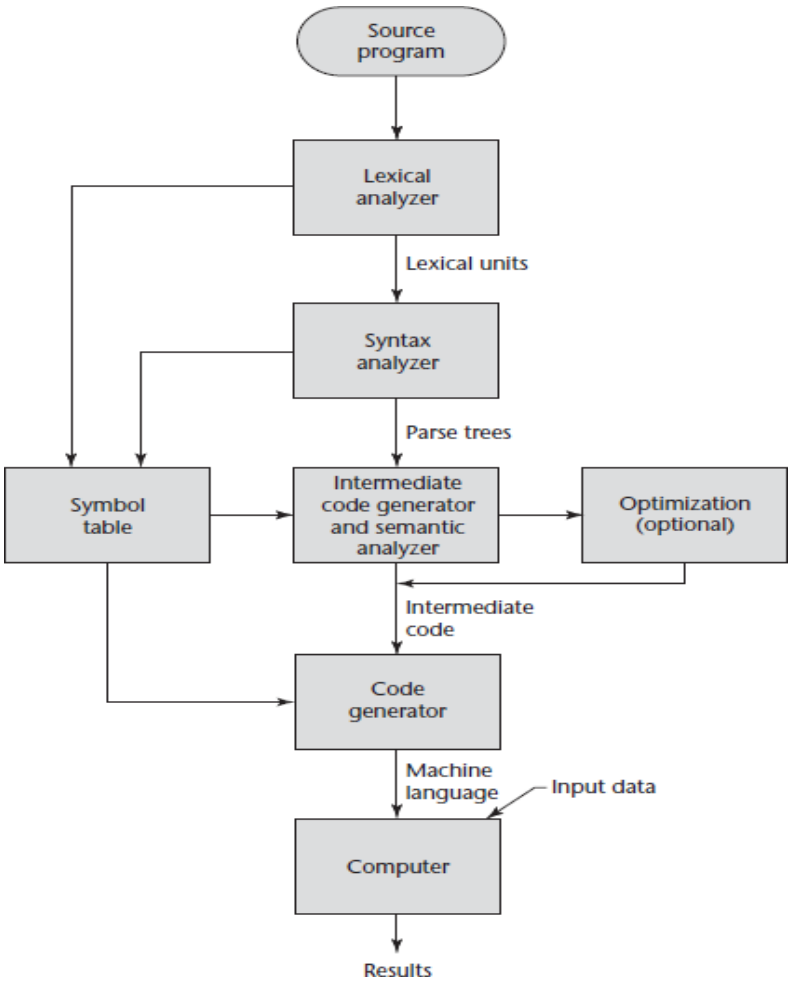
Imperative	Functional	Logic	Object-Oriented
Algol	Lisp	Prolog	Smalltalk
Cobol	Haskell		Simula
PL/1	ML		C++
Ada	Miranda		Java
C	APL		
Modula-3			

Language Categories

Source: Sebesta, 2012

Implementation Methods

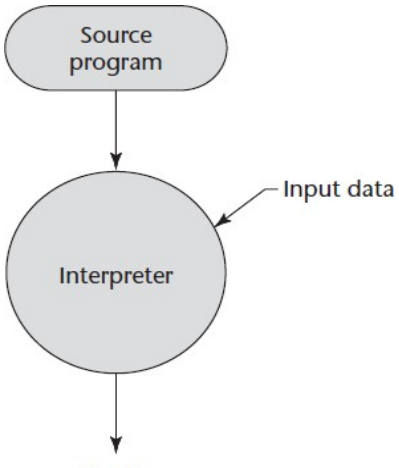
- Compilation – programs are translated into machine language



Compilation

Source: Sebesta, 2012

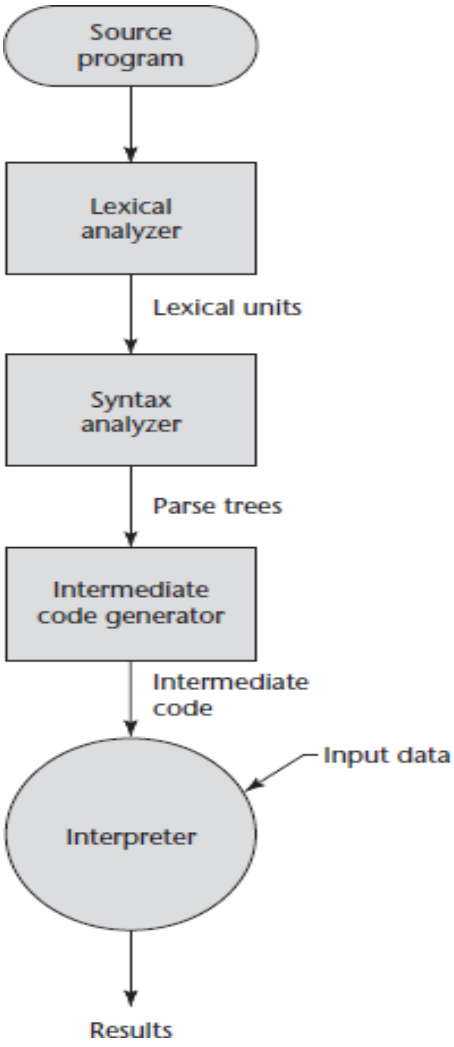
- Pure Interpretation – programs are interpreted by another program called interpreter, without translation



Pure Interpretation

Source: Sebesta, 2012

- Hybrid Implementation Systems – the combination of compilation and interpretation



Hybrid Implementation Systems

Source: Sebesta, 2012

Programming Environment

- A collection of tools used in software development
- May consist of a file system, a text editor, a linker, and a compiler
- May also include a large collection of integrated tools
- Examples: UNIX, JBuilder, Microsoft Visual Studio .NET, NetBeans

References:

Sebesta, Robert W. (2012). *Concepts of Programming Languages.* 10th ed. USA: Pearson Education, Inc.

Ben-Ari, Mordechai (2006). *Understanding Programming Languages*. Chichester: John Wiley & Sons, Inc.

Tucker, Allan B. and Noonan, Robert E. (2002). *Programming Languages: Principles and Paradigms.* 2nd ed. New York: Mc-Graw Hill