Introduction to Programming Languages

CSIT 313

Fall 2015

J.W. Benham

Why Study Programming Languages?

- You'll be better able to express ideas
- You'll have a better background for select appropriate language for given problem
- You'll be more able to learn new languages
- You'll have a better understanding of importance of implementation
- You'll be able to use features of languages you already know more effectively
- Advancement of computing

Programming Domains

- Different types of tasks lead to languages with different goals
- Scientific computing
- Business computing
- Artificial intelligence
- Systems programming

Language Evaluation Criteria: Readability

- Simplicity
- Orthogonality
- Data types
- Syntax design

Language Evaluation Criteria: Writability

- Simplicity and orthogonality
- Support for abstraction
- Expressivity

Language Evaluation Criteria: Reliability

- Type checking
- Exception handling
- Aliasing
- Readability and writability

Language Evaluation Criteria: Cost

- Cost of training
- Cost of developing software
- Cost of compiling
- Cost of executing programs
- Cost of language implementation system
- Cost of poor reliability
- Cost of maintenance

Influences on Language Design: Computer Architecture

- Von Neumann architecture
  - Fetch-execute cycle
  - Imperative languages mirror this type of process
  - Functional languages are less “natural” for this architecture – so more expensive and inefficient

Influences on Language Design: Software Development Methods (1)

- In mid-to-late 1960s, programs were handling larger, more complex tasks
  - Structured programming, top-down design & stepwise refinement
  - Inadequate control statements and incomplete type checking

Influences on Language Design: Software Development Methods (2)

- In late 1970s, another shift from procedure-oriented to data-oriented design
  - Support for data abstraction
  - Object-oriented programming

Categories of Languages

- Imperative languages
  - Visual languages
  - Scripting languages
- Functional languages
- Logic Languages
- Mark-up/programming hybrids

Language Design Trade-Offs

- Reliability and run-time type checking and cost of execution
  - No array bounds checking in C
  - Array bounds checking in Java
- Readability vs writability
  - Example: APL
- Writability vs reliability
  - Pointers in C/C++ vs references in Java

Language Implementations (1)

- Execution environment
  - Machine language
  - Operating system
- Compilation
  - Phases of a compiler
  - Executable image
  - Linking and loading
  - The von Neumann bottleneck

Language Implementations (2)

- Pure interpretation
  - Effect on run time
  - Run-time checks and error messages
- Hybrid systems
  - Compile to intermediate code, which is then interpreted
  - JIT compilation
- Preprocessors