PROGRAMMING LANGUAGES

Lecture no.1

OBJECTIVES

At the end of the lesson, the student must be able to:

1. Define the term programming language
2. Language design and implementation
3. Identify the reasons of studying programming language
4. Determine the criteria for a good p.l.
5. Language paradigms
PROGRAMMING LANGUAGE

- A **programming language** is an artificial **language** designed to express **computations** that can be performed by a **machine**, particularly a **computer**. Programming languages can be used to create **programs** that control the behavior of a machine, to express **algorithms** precisely, or as a mode of human communication. (Wikipedia

**EX.** PASCAL, C, JAVA, C++, VISUAL BASIC, ADA, LISP, FORTRAN, COBOL, PROLOGUE

---

Language design and implementation

- Syntax – What a program looks like
- Semantics – What a program means
- Implementation – How a program executes
WHY STUDY PROGRAMMING LANGUAGES?

1. To improve your understanding of the language you are using. Ex. Use of array and single variable data storage
2. To increase your vocabulary of useful programming constructs.
3. To allow better choice of programming language. Ex. C and Java
4. To make it easier to learn a new language
5. To make it easier to design a new language

WHAT MAKES A GOOD LANGUAGE?

1. Clarity, simplicity, and unity - provides both a framework for thinking about algorithms and a means of expressing those algorithms
2. Naturalness for the application - program structure reflects the logical structure of algorithm
3. Support for abstraction - program data reflects problem being solved
4. Orthogonality - every combination of features is meaningful
WHAT MAKES A GOOD LANGUAGE? (Continued)

5. Programming environment - external support for the language
6. Portability of programs - transportability of the resulting programs from the computer on which they are developed to other computer systems

7. COST OF USE
   - COST OF PROGRAM EXECUTION
   - COST OF PROGRAM TRANSLATION
   - COST OF PROGRAM CREATION, TESTING, AND USE
   - COST OF PROGRAM MAINTENANCE

Language paradigms

Imperative languages
- Goal is to understand a machine state (set of memory locations, each containing a value)
- Statement oriented languages that change machine state (C, Pascal, FORTRAN, COBOL)

Applicative (functional) languages
- Goal is to understand the function that produces the answer
- Function composition is major operation (ML, LISP)
- Programming consists of building the function that computes the answer
Language paradigms (continued)

Rule-based languages
- Specify rule that specifies problem solution (Prolog, BNF Parsing)
- Other examples: Decision procedures, Grammar rules (BNF)
- Programming consists of specifying the attributes of the answer

Object-oriented languages
- Imperative languages that merge applicative design with imperative statements (Java, C++, Smalltalk)