CMPT 120

Topic: Programming Languages and their Evolution
+ Intro to Scratch
Last Lecture

- Computing Science -> Problem Solving
- Problem Solving Process
  - Algorithm
  - Decomposition
- Algorithm
  - 5 ways of expressing algorithm
- Why do we need them?
- Computer Program -> Activity

Steps of Problem Solving Process:
1. Problem Statement
2. a) Solution(s) Design
   -> solution is expressed as an algorithm
   b) Identify data involved in problem:
      input data, internal data and output data (result)
3. Selection
   Select “best” solution by analyzing efficiency of algorithms
4. Implementation
   Implement an algorithm into a computer program
5. Testing
   Does program execute and solve problem?
Learning outcomes

At the end of this lecture, a student will be able to:

• Describe fundamental concepts pertaining to computing science
  • Programming language
  • Evolution of programming languages
Today’s Menu

• Definition of Programming language
• Evolution of programming languages
• Compiled versus interpreted programming languages
• Introduction to Scratch
• Illustrate the Problem Solving Process using Scratch

Step 1: State problem
Step 2: Design algorithm + Identify data
Step 4: Implement algorithm into computer program
Step 5: Test it
Programming Language

• A programming language, like a natural language, is made of

  1. Vocabulary ("building blocks")
     • In English:
     • In Scratch:
     • In Python:

  2. Syntax rules (grammar)

• **Definition:** is a language that allows us to specify detailed instructions that a computer can understand and execute

*CMPT Programming languages are also called formal languages as opposed to natural languages such as English, Mandarin, Spanish,...*
What does a computer understand?
Evolution of Programming Languages

1st generation

1. Machine language -> M.L.
   • CPU dependent – each family or series of CPU has its own M.L.
     • Example:

   • Machine languages’ instructions:
     • Instructions expressed as a series of
     • Example:

   • Advantage:
     • Fastest execution since there is no need to translate the instructions of the program into M.L.

   • Disadvantage:
     • Tedious to learn, use, read and debug
Evolution of programming languages

2nd generation

2. Assembly languages -> low level languages
   - 1 level of abstraction away from M.L.
   - Still CPU dependent (like M.L.)
     - Example:

   - Assembly languages’ instructions
     - Example:

   - Translation required:

   - Advantage:
     - Easier to read than M.L.

   - Disadvantage:
     - But still tedious to learn, use, read and debug + error-prone
Evolution of programming languages

3. High level languages: visual and text-based
   • Abstraction: Further away from M.L.
   • CPU independent
     • Example:

   • High level languages’ instructions
     • Example:

   • Translation:
     • See next slide
   • Advantage:
     • Easier for humans to learn, use, read and debug
   • Disadvantage:
We only need to translate, i.e., compile our program (in high-level language) once.

Every time we want to execute our program, we simply “run” its *executable* file.
Translation – Interpreter

- Every time we want to execute our program, we must “interpret” it.
- This “interpretation” process is repeated until the whole program has executed.
Examples of Programming language – Hello World!

• This web site shows the “Hello World!” program written in a multitude of programming languages http://helloworldcollection.de/
Let’s introduce Scratch

• Where to find Scratch?
  • [https://scratch.mit.edu/](https://scratch.mit.edu/)

• What is Scratch?
  • “Scratch is a block-based imperative, event-driven, dynamically-typed and interpreted programming language.”
  Source: [https://wiki.scratch.mit.edu/wiki/Programming_Language](https://wiki.scratch.mit.edu/wiki/Programming_Language)

• Why starting with a visual programming language such as Scratch?
  • Fun to learn and use
  • Easily accessible
Summary

- Defined programming language
- Outlined evolution of programming languages through first 3 generations
- Described compiler and interpreter
- Introduced to Scratch
- Demo’ed of the Problem Solving Process using Scratch

Step 1: State problem
Step 2: Design algorithm + Identify data
Step 4: Implement algorithm into computer program
Step 5: Test it
Next Lecture

• i-clicker Session 1
• Introduce Python
• Illustrate the Problem Solving Process using Python