Fundamentals of Computer Programming

How class is conducted

- Learn basic programming concepts
- Learn Python programming language
- Learn problem-solving skills
- Lab sessions (Monday sessions)
- Lecture sessions (Friday sessions)

Resources

- Textbooks
  - A. Downey, “Think Python.” 2012. O’Reilly.
    url: http://greenteapress.com/wp/think-python/
- Online resources
  - https://www.python.org/doc/

Introduction

- A modern computer:
  "a machine that stores and manipulates **information** under the control of **a changeable program**."

- Computer program:
  - A detailed step-by-step set of instructions telling a computer what to do.

Introduction

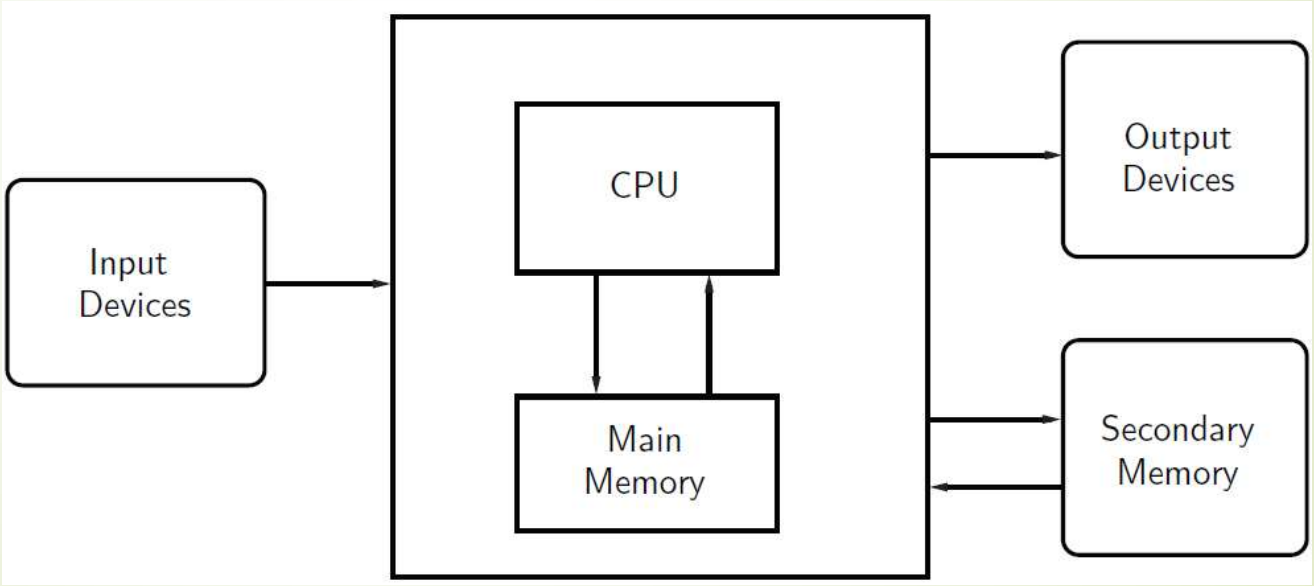
- Computer, The Universal Machine
  - With a suitable programming, any computer can do tasks any other computer can do.
- Program is called software
- Computer (Machine) is called hardware
- Hardware operates under the control of a software

Introduction

- Learn programming
  - Develop problem-solving skills
    - Analyze a problem
    - Simplify it to interaction among simpler tasks
    - Refine them to available resources, i.e., instructions
- Algorithms
  - a step-by-step process to achieve the desired results

Introduction

- Hardware (Von Neumann Machine)



Von Neumann Machine

- Central Processing Unit (CPU)
  - carries out all the computations
    - calculate arithmetic
    - evaluate logic
    - manage a flow of information
  - CPUs manipulate information
  - Programs control CPU
    - A program is also an information!

    Great Idea!

    → Information controls hardware to control information.

Von Neumann Machine

- Memory
  - stores information
    - Information: programs and data

Another paradox?

CPU uses memory.
CPU is controlled by programs.
Programs are stored in memory.
So, who controls who?

No. it's not like who is the supreme leader.
It's more like working together.
Peace and harmony at last.

Von Neumann Machine

- Memory
  - CPU can only directly access information stored in main memory (RAM or Random Access Memory).
  - Main memory is fast, but volatile, i.e. when the power is interrupted, the contents of memory are lost.

Von Neumann Machine

- Memory
  - Secondary memory provides more permanent storage: magnetic (hard drive), flash (SSD, USB memory), optical (CD, DVD)
  - CPU accesses secondary memory through main memory.

    "Memory hierarchy."

Von Neumann Machine

- Input
  - Information is passed to the computer through keyboards, mice, etc.
- Output
  - Processed information is presented to the user through the monitor, loudspeaker, printer, etc.

Von Neumann Machine

- Fetch-Execute Cycle
  - CPU retrieves an instruction from memory
  - CPU decodes the instruction to see what it means
  - CPU carries out the action
  - Next instruction gets fetched, decoded, executed.
  - … fetch, decode, execute, repeat, …

Programs/Programming Languages

- Program (in computer memory)
  - machine code
  - specific to hardware
  - inhumane to read
    E.g., 0x0A0911FE
- Program (We will write)
  - more humanize
    - print("Hello")

Programs/Programming Languages

- Program (We will write)
  - more humanize
    - print("Hello")
  - not (or at least less) specific to hardware
    - So, we write (humanized) programs.
    - Computer reads (inhumane) programs.
    - Our programs get translated to computer program by a translator.

Our program is often called a program.
Computer program is often called machine code.

Programs/Programming Languages

- Translator
  - a program to translate another program written in programming language to machine code
  - Machine code: specific to machine
  - Program: specific to programming language

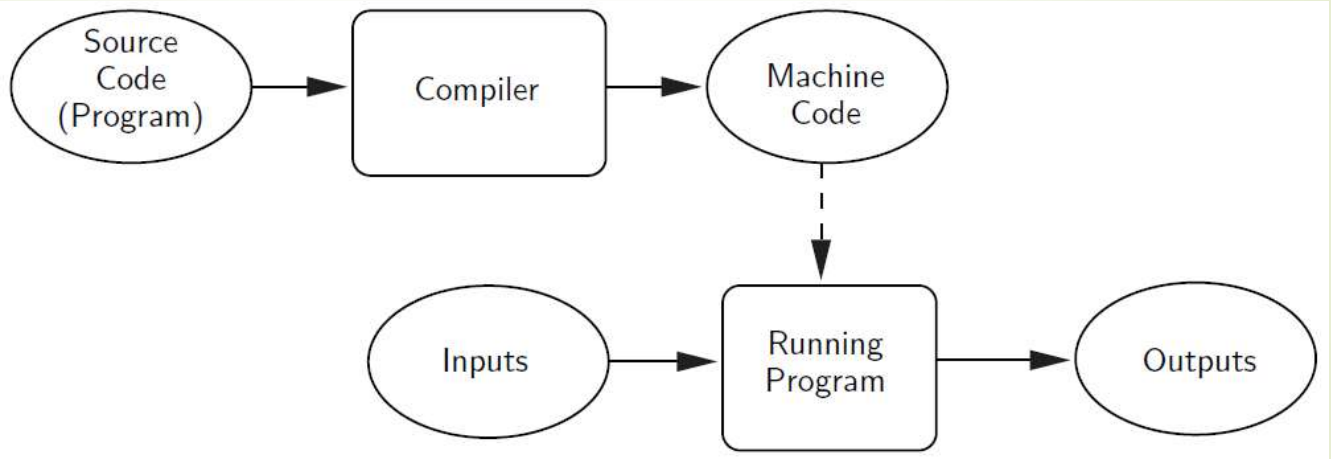
Programming Languages

- Natural language has ambiguity and imprecision
  - not suitable to describe complex algorithms.
- Programs are written in programming languages
- Every structure in programming language has a precise form, called *syntax*
- Every structure has a precise meaning, called semantics

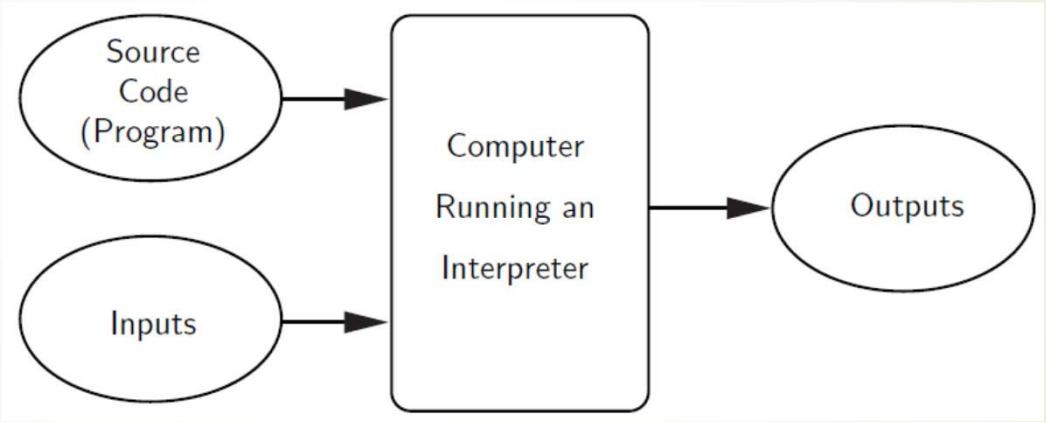
Translator: Compiler v.s. Interpreter

- Translation: Program --> Machine code
- Compiler
  - translates a program to machine code
- Interpreter
  - simulates a computer that understands a high-level language
  - the source program is not translated into machine code all at once
  - An interpreter analyses and executes the source code instruction by instruction

Compiler


Source Code (Program)
Compiler
Machine Code
Inputs
Running Program
Outputs

Interpreter



Compiler v.s. Interpreter

- Compiler
  - Once program is compiled, it can be executed over and over without the source code or compiler.
- Interpreter
  - The source code and interpreter are needed each time the program runs

Compiler v.s. Interpreter

- Compiled programs generally run faster since the translation of the source code happens only once.
- Interpreted languages are part of a more flexible programming environment since they can be developed and run interactively.
- Interpreted programs are more *portable*, meaning the executable code produced from a compiler for a Pentium won't run on a Mac, without recompiling. If a suitable interpreter already exists, the interpreted code can be run with no modifications.

Python

- "Python is an interpreted, interactive, object-oriented programming language. … Python is portable: it runs on many Unix variants, on the Mac, and on Windows 2000 and later." – python.org
- Why do we learn Python?
  - "Python is a powerful, flexible, open source language that is easy to learn, easy to use, and has **powerful libraries** for data manipulation and analysis. …
    Python has a unique combination of being both a capable **general-purpose programming language** as well as being easy to use for analytical and quantitative computing." -- https://www.continuum.io/why-python

Glimpse of Python

```
print("Hello")
x = 8
print("x")
print(x)
```

- Each command is called a *statement*
- print(…) is a command to invoke a built-in function *print*
- what in the parentheses are function's *arguments*.
  - `"Hello"` is an argument of `print("Hello")`
  - `"x"` is an argument of `print("x")`
  - `x` is an argument of `print(x)`

Glimpse of Python

```
print("Hello")
x = 8
print("x")
print(x)
```

- `print` function
  - `help(print)`
- built-in functions
  - *help(__builtins__)*
  - type <*q*> to quit
  - or better, see `https://docs.python.org/3/library/functions.html`

Glimpse of Python

```
print("Hello")
x = 8
print("x")
print(x)
```

- `x = 8`
  - `x` is an example of a *variable*
  - A variable is used to assign a name to a value so that we can refer to it later.
  - `x = 8` is an *assignment* statement
  - Variable is assigned a value of `8` (the right-hand side of the "=")

Milestones

- Introduction ✓
- First program in python
- Basic console output/input & variables
- Selection / flowchart
- Functions
- Loops
- Collection variables
- Files
- Advanced topics

For a curious mind

- A program in Python is compiled into a bytecode, then executed by the interpreter in a similar manner to Java.
- What are advantages of bytecode over native code?
  - "Hank Shiffman from SGI said:
    There are three advantages of Java using byte code instead of going to the native code of the system … **Portability**, **Security**, **Size**…"
- src: `https://stackoverflow.com/questions/48144/what-are-advantages-of-bytecode-over-native-code`