Lecture 2. Programs & Programming Language

Program(Software)

set of instructions that tell a computer what to do

— We use programs to interact(互动)/talk with computers

—To write program we use programming languages

Programming Language

Languages used to write programs

— Computers are machines → They don't understand human languages

— Programs are written in a language a computer can understand → programming language

Machine Language

A computer's native language

— Use zeros & ones (0|1) → Binary Language → Vary hard to use

— Machine language is machine dependent, so it's differs among different types of machines (机器语言是机器相关的，因此不同类型的机器之间的语言有所不同)

—Every instruction should be written in machine language before it can be executed

Assembly Language

Was developed to make programming easier

- Machine dependent
- Introduced keywords: add, sub, ...
- To add 2 and 3 get the result: add 2, 3, result (can't be executed)
- A program called 'assembler' translates assemble code to machine code

High-Level Language

A new generation of programming language

- Uses English words | Easy to learn & use.
- Machine independent | your program will run on different machines.
- Instructions are called 'statements'.
- A program written in a high-level language is called a 'source program' or 'source code'. (编程语言编写的程序称为程序源或源代码)
- A 'compiler(编译器)' or an 'interpreter(解释器)' is used to translate source code to machine code.

compiler: Translates all the source code into machine code(executable)

(A compiler translates the entire source code program into machine code or an intermediate representation in a single pass.)

source code → **compiler** → machine code → **executor** → output

interpreter: translates each statement into machine code and executes it right away

*(An interpreter processes the source code **line-by-line or statement-by-statement**, executing each line immediately after it is translated.)*

statement → interpreter → output

compiler vs interpreter

- **翻译流程**

  编译器：编译器将整个源代码程序一次性翻译成机器代码或中间表示。 它生成可执行的二进制文件或字节码，可以独立于源代码运行。
  解释器：解释器逐行或逐语句处理源代码，在翻译后立即执行每一行。 没有生成单独的输出文件，代码直接从源代码执行。

- **执行**

  编译器：编译是在程序执行之前完成的。 这意味着整个程序是一次性编译的，代码中的任何错误或问题都必须在执行之前解决。 编译的程序往往执行速度更快，因为它们已经被翻译为机器代码。
  解释器：解释是在运行时完成的。 每行或语句都会即时翻译和执行。 如果代码的某一部分发生错误，解释器仍然可以执行程序的其余部分，为程序员提供更即时的反馈。

- **可移植性**

  编译器：由编译器编译的程序通常是特定于平台的，因为生成的机器代码是根据目标体系结构定制的。 要在不同平台上运行相同的程序，通常需要为每个平台重新编译它。
  解释器：解释程序更加可移植，因为它们依赖解释器来执行代码。 只要解释器适用于特定平台，就可以执行相同的源代码而无需修改。

- **调试**

  编译器：使用编译语言进行调试可能更具挑战性，因为错误通常是在编译过程后检测到的。 调试信息可能有限，并且更难查明问题的根源。
  解释器：解释型语言通常在调试过程中提供更即时的反馈，因为在执行过程中遇到错误。 这可以更轻松地定位和修复代码中的问题。