

CSI31 Lecture 2

Topics:

- 1.5 Programming languages (page 6)
- 1.6 The Magic of Python (page 9)
- 1.7 Inside a Python Program (page 14)
- 1.8 Chaos and Computers (page 17)

HW1:

- p. 19 read section 1.9 Summary
- p. 20 True/False (5-10)
 - Multiple Choice (5-10)
 - Discussion (5)
- Install Python on your computer
- p.22 (Programming exercises)
 - No 1, 4

1.5 Programming languages

Recall that

computer program

is a detailed, step-by-step set of instructions telling a computer exactly what to do.

So we need to provide those instructions in a language that a computer can understand.

Our natural language is full of ambiguity and imprecision.

Computer scientists designed notations for expressing computations in an *exact* and *unambiguous* way. These special notations are called *programming languages*.

Every structure in a programming language has:

a precise *form* (its *syntax*) and a precise *meaning* (its *semantics*)

Programming language is like a code for writing down instructions that a computer will follow.

Programs written in a programming language are often called *computer code* or *source code*.

The process of writing an algorithm in a programming language is called *coding* (or *implementation*).

A list of programming languages (not full):

C++, Java, Python, Perl, Scheme, Visual Basic, Pascal, ...

- *high-level computer languages* (i.e. close to our natural language)

Computer hardware can only understand a very low-level *machine language*.

Compilers and Interpreters

To add two numbers in Python: $c = a + b$

To add two numbers in machine language: (something like)

load the number from memory location 2001 into the CPU

load the number from memory location 2002 into the CPU

add the two numbers in the CPU

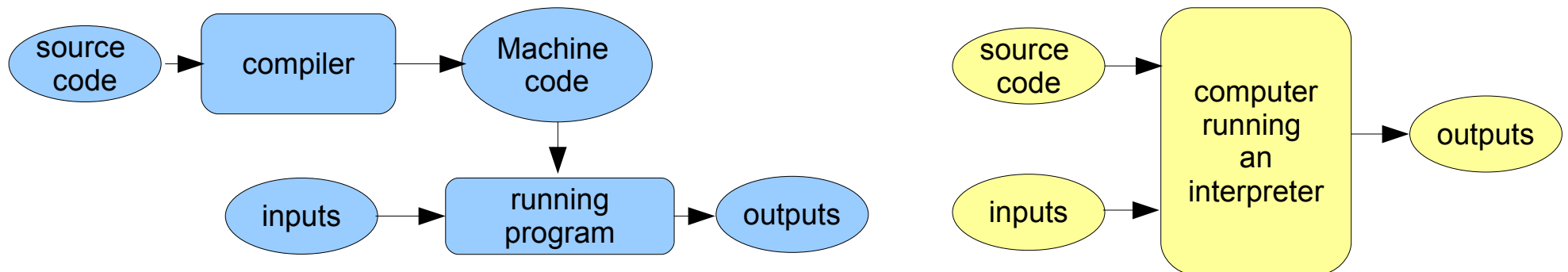
store the result into location 2003

- all these instructions and numbers are represented in *binary notation* as sequences of 0's and 1's.

Thus we need some way to *translate* high-level language into low-level language that the computer can execute. There are two ways to do it: *compile* or *interpret*

Compiler

is a computer program that takes a program written in a high-level language and *translates* it into an equivalent program in the machine language *of some computer*.



Interpreter

is a computer program that simulates a computer that understands a high-level language.

Rather than translating the source program into a machine language equivalent, the interpreter analyses and executes source code instruction by instruction as necessary.

- 1.6 The Magic of Python
- 1.7 Inside a Python Program
- 1.8 Chaos and computers

class work (see programs)