CSI31 Lecture 2

Topics:

1

1.5 Programming languages

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

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*.

1.5 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*).

1.5 Programming languages

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*.

The *machine language* of a computer is created by the designers of the particular CPU. Each kind of computer has its own machine language.

1.5 Programming languanges: Compilers and Interpreters
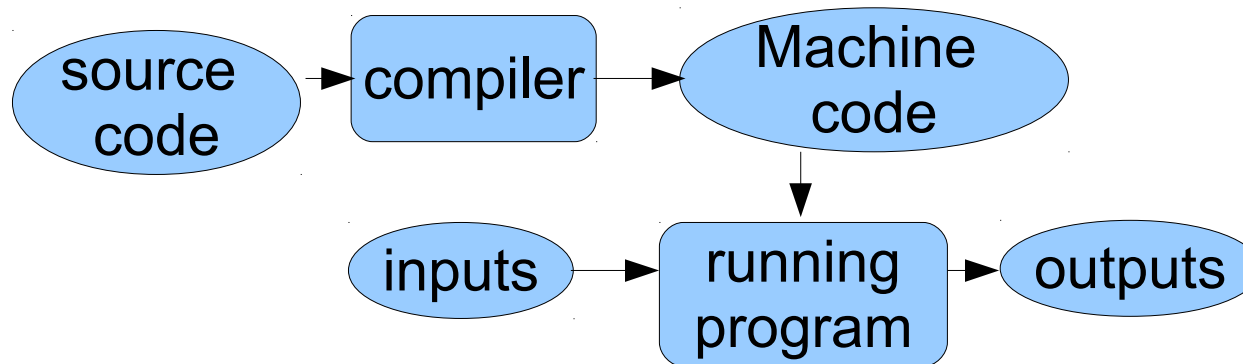
To add two numbers in Python: c = a + b

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 *interprete*

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*.
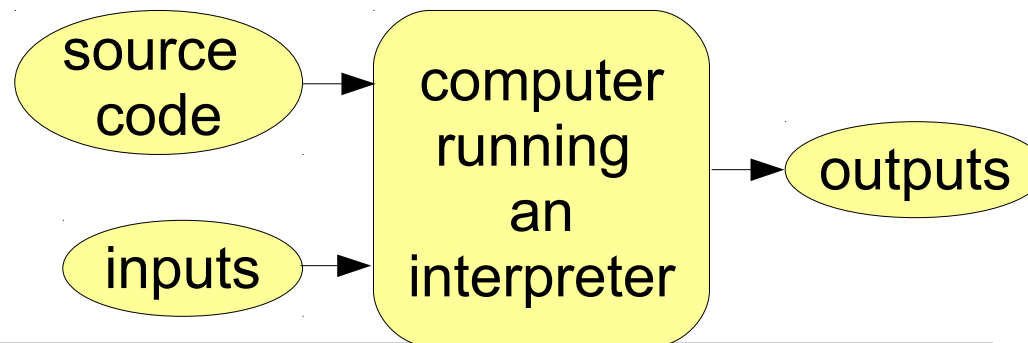
```
  source  ──►  compiler  ──►    Machine
   code                           code
                                    │
                                    ▼
  inputs  ──►   running   ──►   outputs
              program
```

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.

source code → computer running an interpreter → outputs

inputs →

1.6 The Magic of Python

« in some ways programming *feels* like magic»

Python interpreter: Genie - *we can give instructions to the Python interpreter, and it directs the underlying spirits to carry out our demands*.

1.6 The Magic of Python

« in some ways programming *feels* like magic»

Python interpreter: Genie - *we can give instructions to the Python interpreter, and it directs the underlying spirits to carry out our demands.*

First, let's work with Python Shell:

```
>>> print()

>>> print('' Hello World! '')

>>> print(10+11)

>>> print(''10+11='',10+11)
```

1.6 The Magic of Python

Function definition:

defined a *function/method* `hello()` with no *parameters/arguments*

```
>>> def hello():
        print("Hello!")
        print("Computers are fun!!!")

>>> hello()
```

function/method is *invoked/called*

1.6 The Magic of Python

Function definition:

```
>>> def hello():
      print("Hello!")
      print("Computers are fun!!!")
```
defined a *function/method* `hello()` with no *parameters/arguments*

```
>>> hello()
```
← function/method is *invoked/called*

```
>>> def hello(name):
      print("Hello",name)
      print("Computers are fun, don't you agree?")
```
defined a *function/method* `hello()` with one *parameter/argument*

```
>>> person = ''Helen''
>>> hello(person)
```

11

```
>>> hello(''Margaret'')
```

1.7 Inside a Python Program
1.8 Chaos and computers

see programs chaos.py and chaos_comments.py