What is computer science?

Computer science combines some of the best features of mathematics, engineering, and natural science.

Computer science uses formal languages to denote ideas (specifically computations).

Software engineering:
Design and assembling of components into systems.
Evaluate advantages/disadvantages among a pool of alternatives.

Science:
Observe the behaviour of complex systems, form hypotheses, and test predictions.

Reference: https://www.greenteapress.com/thinkpython/thinkCSpy/html/chap01.html

Problem Solving

The single most important skill for a computer scientist is **problem solving**.

- Ability to formulate problems.

- Think creatively about solutions.

- Express a solution clearly and accurately.

- The process of learning to program is an excellent opportunity to practice problem-solving skills.

- Learning to program, while using programming as a means to understand physics.

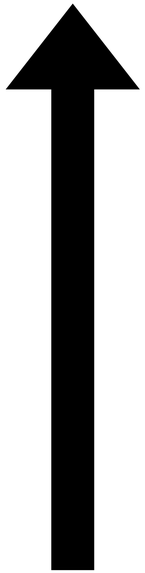
Reference: https://www.greenteapress.com/thinkpython/thinkCSpy/html/chap01.html

High-level vs. Low-level languages

Human

High-level

Python, Julia, IDL/GDL, Mathematica, Matlab



C++, java

C89/90, Fortran77/90

Computer

Low-level

Binary (01001000)

High-level vs. Low-level languages

Roughly speaking, computers **can only execute programs written in low-level languages.**

Programs written in a high-level language **have to be processed** before they can run.

This **extra processing takes some time**, which is a small disadvantage of high-level languages.

Low-level programs can run on only one kind of computer and **have to be rewritten to run on another.**

High-level vs. Low-level languages

The advantages of high-level languages are enormous:

- It is much easier to program in a high-level language.
- Programs written in a high-level language take less time to write.
- They are shorter and easier to read, and they are more likely to be correct.
- They are portable, meaning that they can run on different kinds of computers with few or no modifications.

Due to these advantages, **almost all programs are written in high-level languages.**

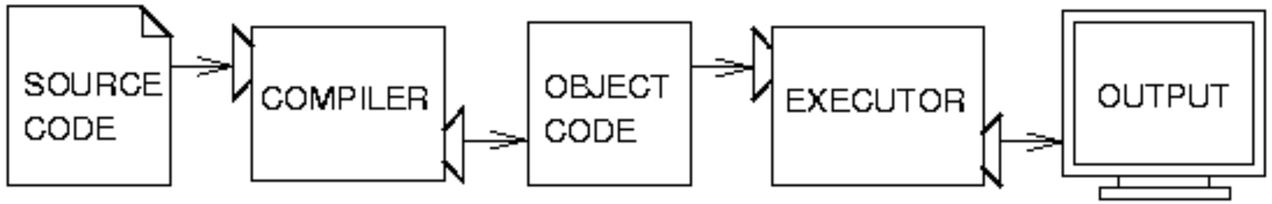
Low-level languages are used only for a few specialised applications.

Interpreter vs. Compiler

Two kinds of programs process high-level languages into low-level languages: **interpreters** and **compilers**.

An **interpreter** reads a high-level program and executes it, meaning that it does what the program says. It processes the program a little at a time, alternately reading lines and performing computations.

A **compiler** reads the program and translates it completely before the program starts running. In this case, the high-level program is called the **source code**, and the translated program is called the **object code** or the **executable**. Once a program is compiled, you can execute it repeatedly without further translation.



Python

Python is considered an interpreted language because Python programs are executed by an interpreter.

There are two ways to use the interpreter:
command-line mode and **script mode**.

In command-line mode, you type Python programs and the interpreter prints the result:

```
$ python
Python 2.4.1 (#1, Apr 29 2005, 00:28:56)
Type "help", "copyright", "credits" or "license" for more information.
>>> print(1 + 1)
2
```

Python

```
$ python
Python 2.4.1 (#1, Apr 29 2005, 00:28:56)
Type "help", "copyright", "credits" or "license" for more information.
>>> print(1 + 1)
2
```

The first line of this example is the command that starts the **Python interpreter.**

The next two lines are **messages from the interpreter.**

The third line starts with **>>>**, which is the **prompt** the interpreter uses to indicate that it is ready.

We typed `print(1 + 1)`, and the interpreter replied `2`.

Python

Alternatively, you can write a program in a file and use the interpreter to execute the contents of the file. Such a file is called a **script**. For example, we used a text editor to create a file named `latoya.py` with the following contents: `print(1 + 1)`

By convention, files that contain Python programs have names that end with `.py`.

To execute the program, we have to tell the interpreter the name of the script:

```
$ python script.py
2
```

Working on the command line is convenient for program development and testing, because you can type programs and execute them immediately.

Once you have a working program, you should store it in a script so you can execute or modify it in the future.