Computer Science & Programming

Lecture 2: Programming

Stephen Huang

January 23, 2023

UNIVERSITY of HOUSTON

Contents

1. Problem-Solving
2. Programming Languages
    1. Machine Languages
    2. Assembly Languages
    3. High-level Languages *
    4. Programming Languages *

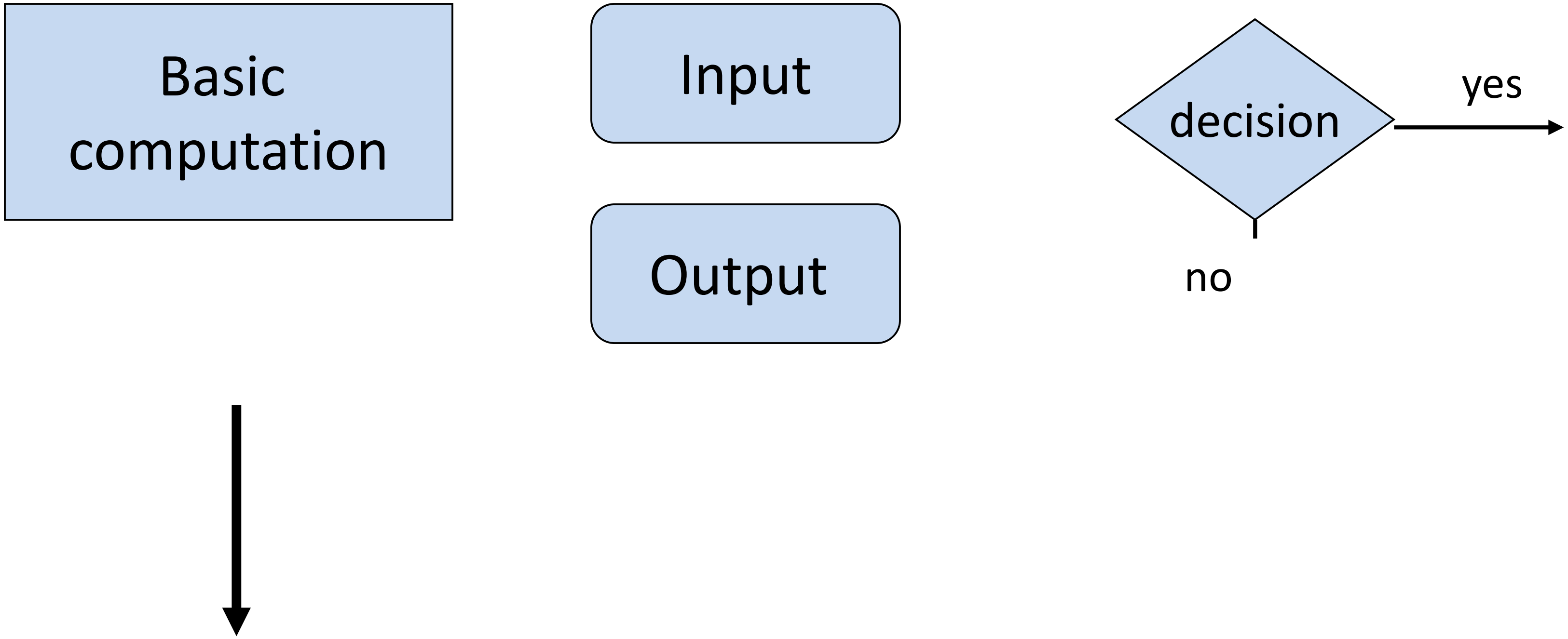
1. Problem-Solving

- The single most important skill for a computer scientist is problem-solving.
- Problem-solving means the ability to
  - formulate problems,
  - think creatively about solutions, and
  - express a solution clearly and accurately.

Syntax

- How do we describe an algorithm as a solution?
  - Clearly
  - Precisely
  - Accurately
- There are many ways to do so, such as BNF, Context-free Grammar, Flowcharts, etc.
- Most Python textbooks do it informally because it is supposedly the obvious way to do it.

Flowchart Components



Basic computation

Input

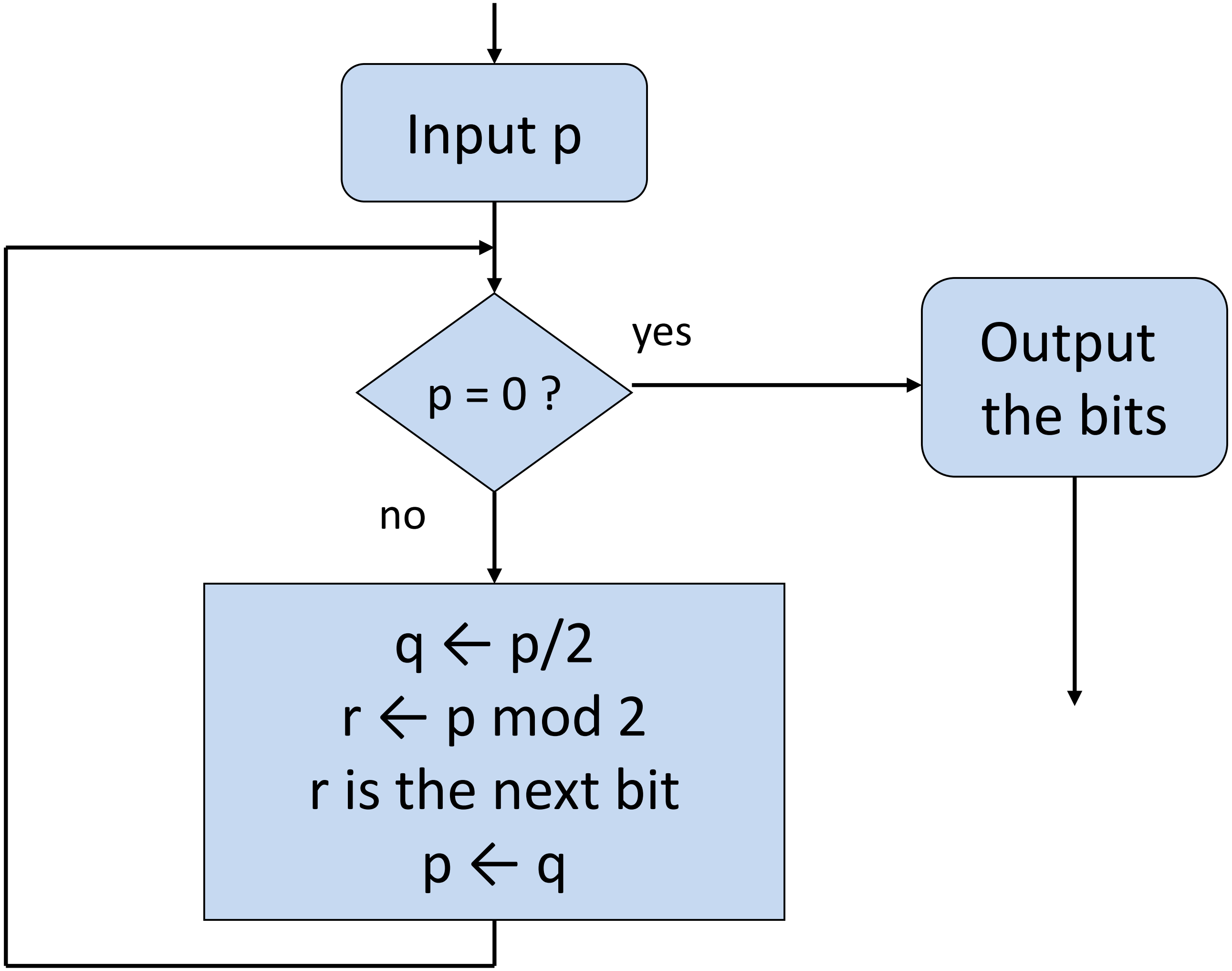
Output

decision

yes

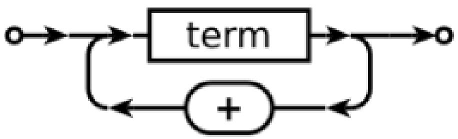
no

A Simple Flowchart

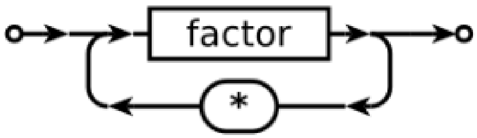


Pascal Syntax Diagrams

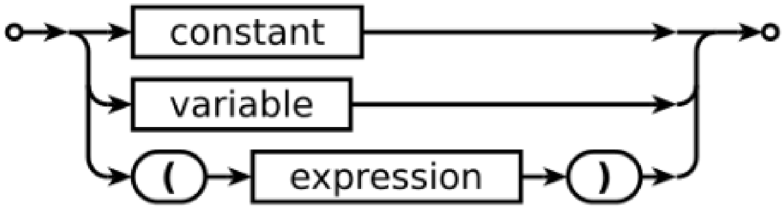
expression:



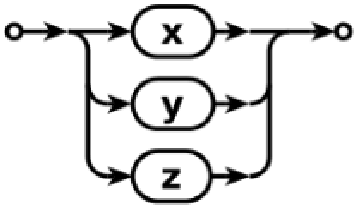
term:



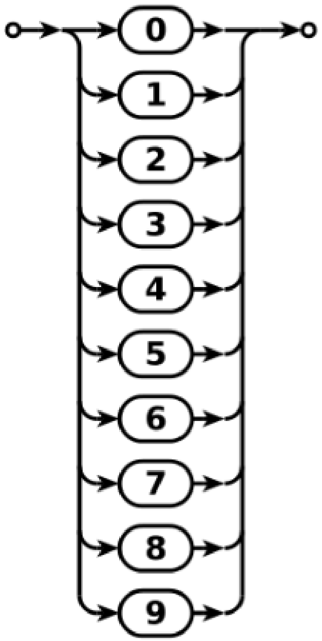
factor:



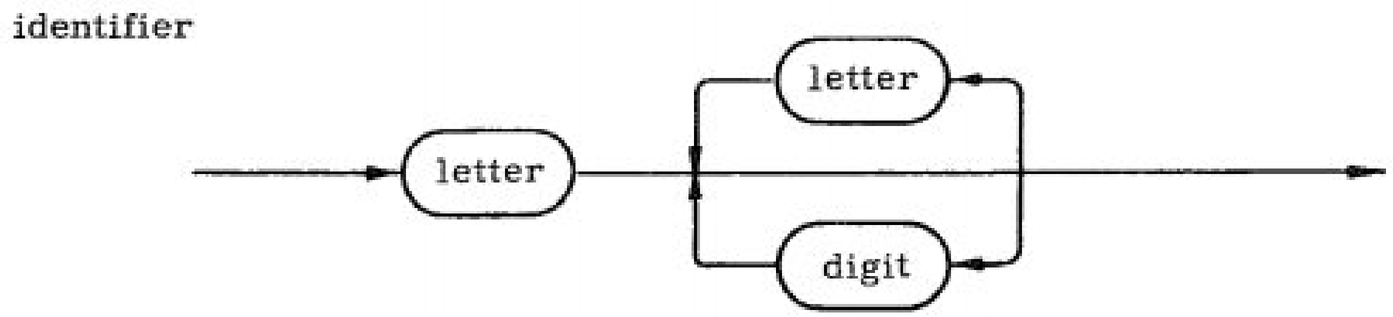
variable:



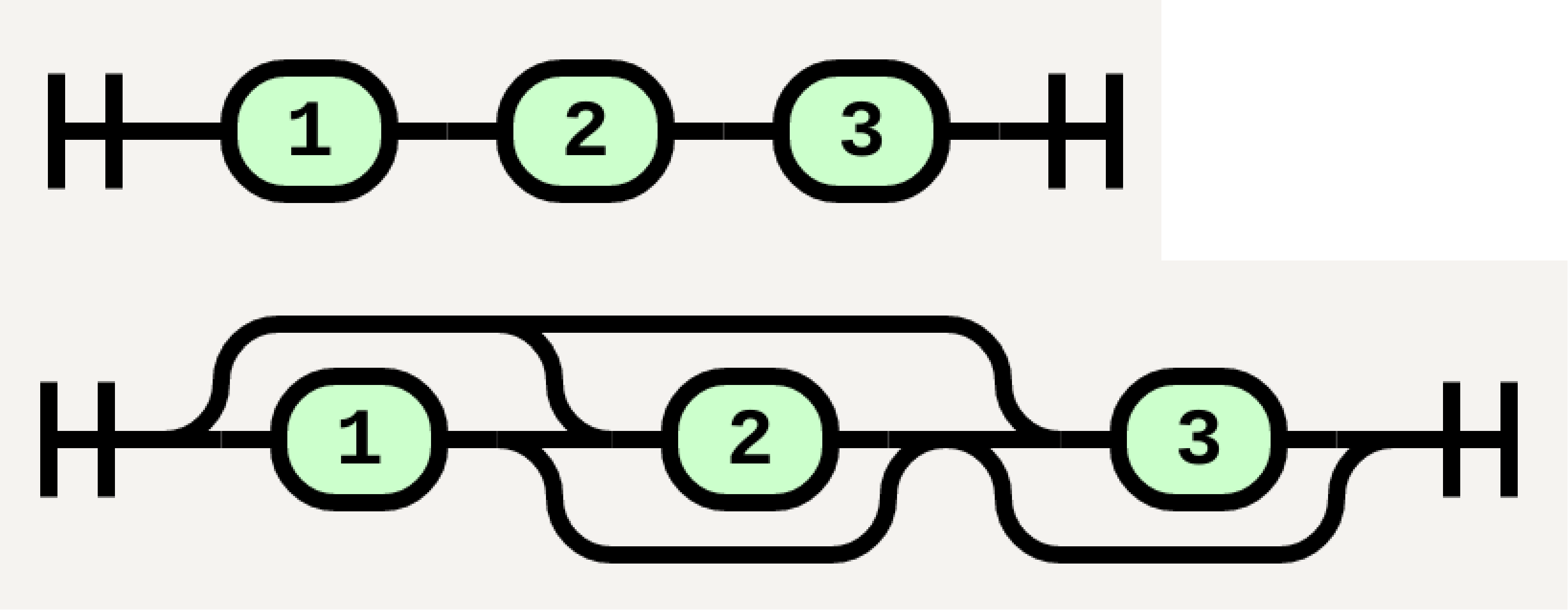
digit:



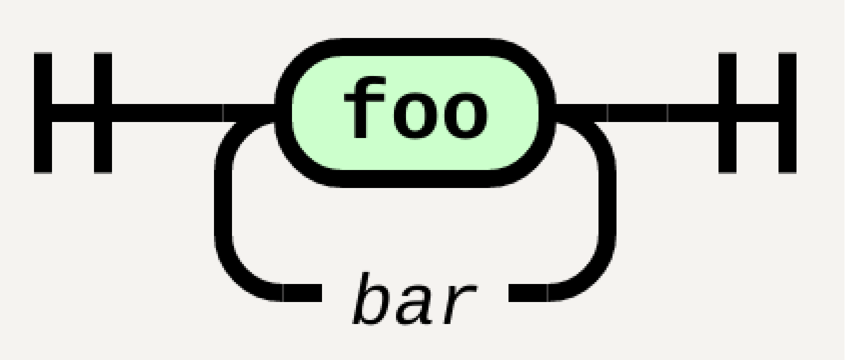
Pascal Syntax Diagrams

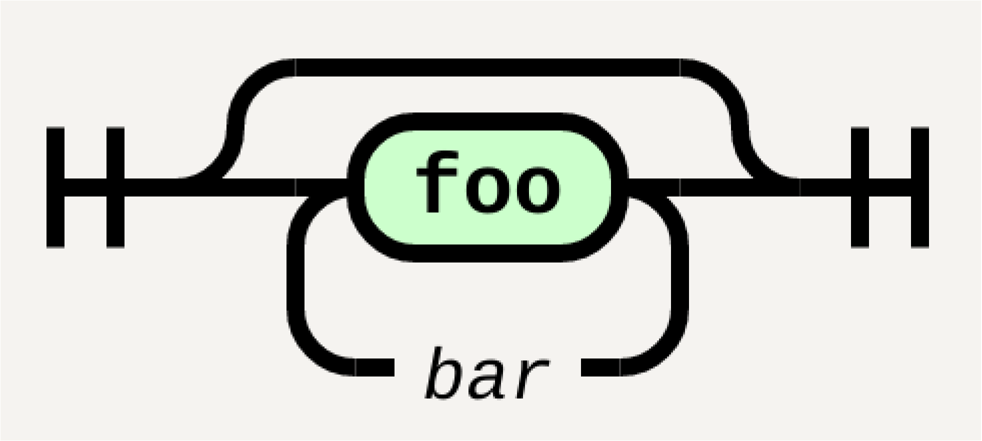


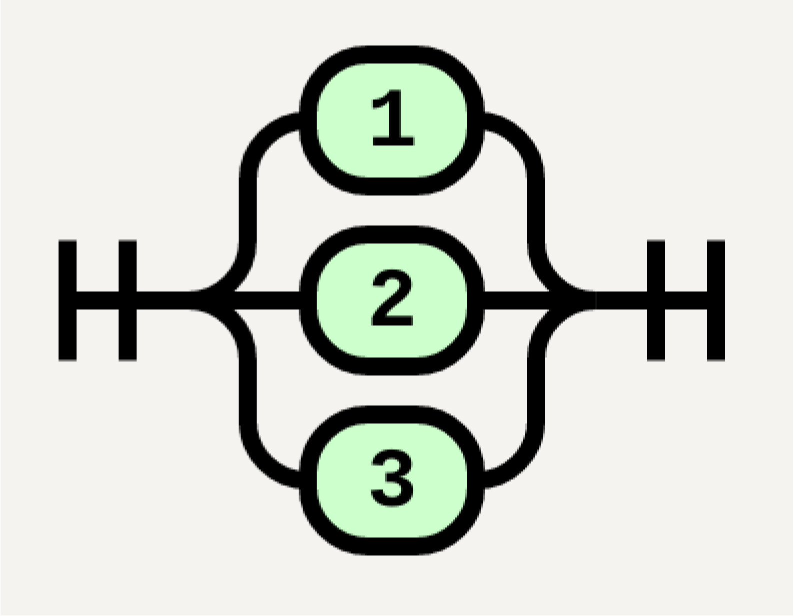
Railroad Diagrams



Railroad Diagrams





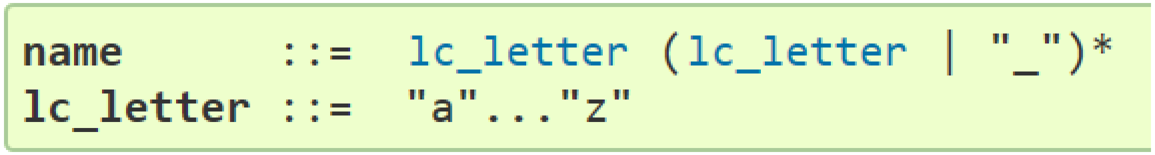


Syntax

- We will not use any particular method to describe Python's syntax formally.
- Understanding the formal syntax is vital for CS majors.
- We will use intuitive examples most of the time.
- When in doubt, you can always check the official Python Language Reference (https://docs.python.org/3/reference/).

Python Reference

- The syntax uses a modified BNF grammar notation:



```
name      ::=  lc_letter (lc_letter | "_")*
lc_letter ::=  "a"..."z"
```

  - Each rule begins with a name (which is the name defined by the rule) and ::=.
  - A vertical bar (|) is used to separate alternatives;
  - A star (*) means zero or more repetitions of the preceding item,
  - a plus (+) means one or more repetitions.

2. Programming Languages

- There are 3 classes of programming languages,
  - machine languages,
  - assembly languages, and
  - high-level languages.

2.1 Machine Languages

- Each computer has its machine language, which is very detailed and specific to the exact details of the computer architecture.
- All instructions in a machine language are in binary codes.
- Since machine languages are machine-dependent, very detailed, and use binary codes, it isn’t easy to write them.

2.2 Assembly Languages

- Assembly languages are symbolic versions of machine languages.
- The instructions use symbolic codes rather than binary codes and, thus, are easier to remember.
- But assembly languages are still machine-dependent and very detailed, therefore difficult to use.

Assembly Languages

- Forexample, an assembly language might have an instruction like

  `Load R4, X`

  loading the value of variable X into the register R4 in the ALU.

- In machine language, this same instruction might be

  `00010111 0100 0010011…0`

  where the first 8 bits are a binary code standing for load, the next 4 bits give the register number in binary, and the last bit sequence shows the address of the variable X.

2.3 High Level Languages

- High-level languages (HLL) are machine-independent and much less detailed than machine/assembly languages.
- HLL makes it easier to write programs and allows the same program to be run on different computers. Most languages have standards defined.
- There are many high-level languages including Fortran, C, C++, C#, Java, and Python.
- We will use Python in this course.

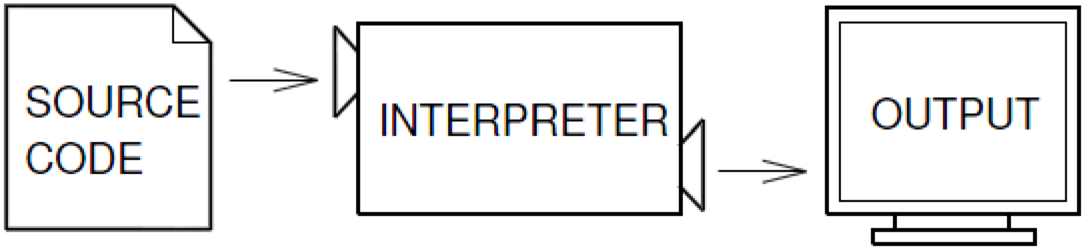
Type of PLs

- Two kinds of programs process high-level languages into low-level languages:
  - Interpreters, and
  - Compilers.

Interpreter

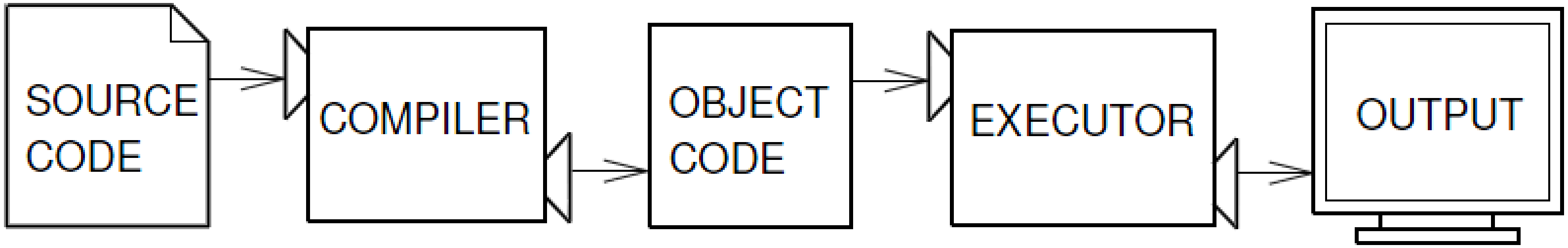
- An interpreter reads a high-level program and executes it, meaning it does what the program says.
  - It processes the program a little at a time, alternately reading lines and performing computations.

One instruction at a time



Compiler

- A compiler reads the program and translates it entirely before it starts running.
  - In this context, 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.



persistent

A sample C++ program

```cpp
/* Hello World program */
#include <iostream>
using namespace std;
int main()
{
   cout<<"Hello World!"<<endl;
   return 0;
}
```

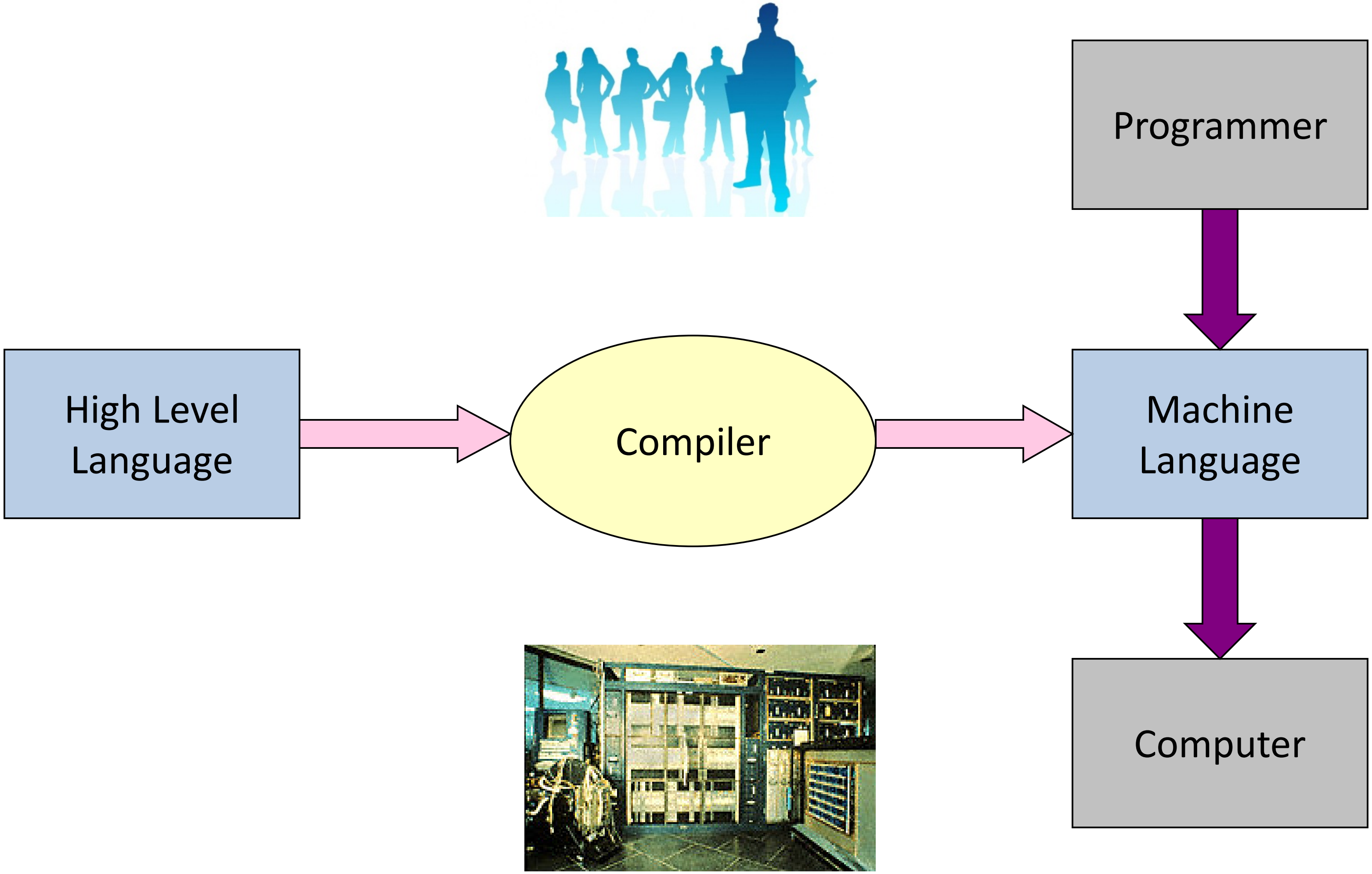
Python Equivalent

```
print("Hello World!")
```

Compilers

- When the computer executes a program, it is always in the machine language of that computer.
- However, it is difficult and tedious to program in machine language. Most programs are written in a high-level, machine-independent programming language like C++.
- Before the computer can execute a program written in such a language, it must be translated into the computer's machine language.

The Role of a Compiler



Programmer

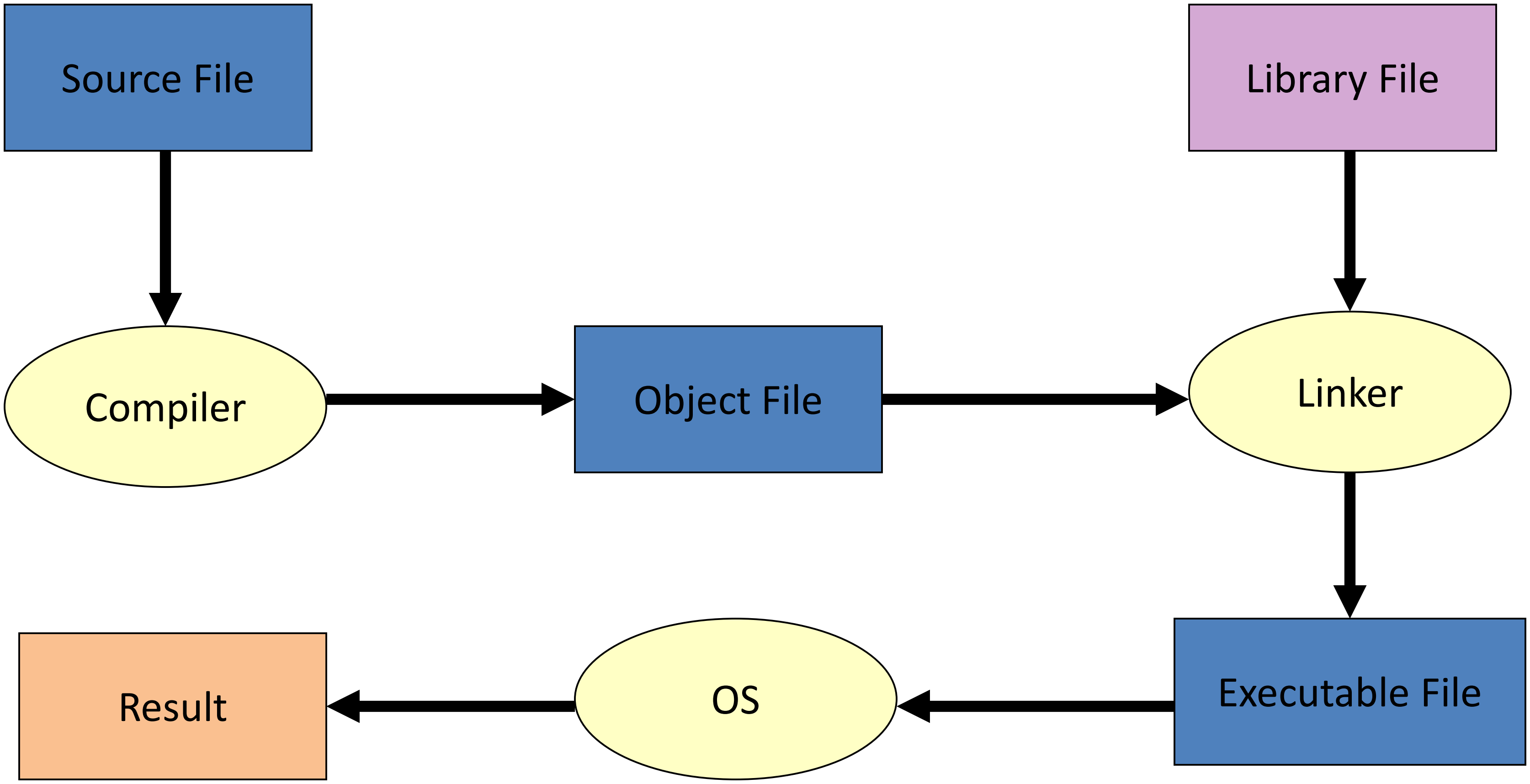
High Level Language

Compiler

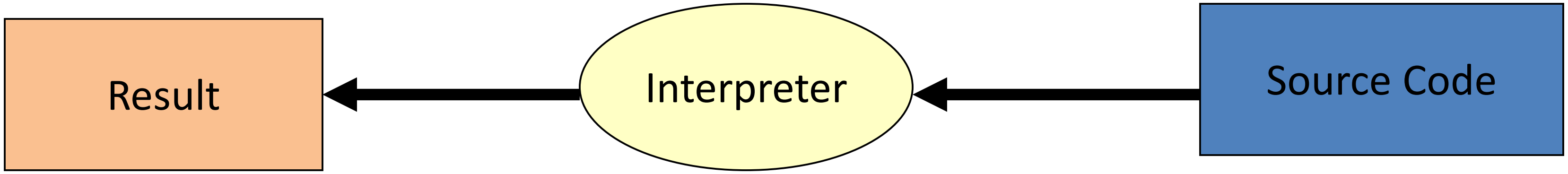
Machine Language

Computer

Steps of Compilation



Steps of Interpretation



2.4 Programming Languages

- We want programming languages to support writing programs that are
  - concise,
  - clear and precise,
  - simple and natural
- Designing programming languages is an art, not just a science.
- So is using them to write code.

Elements of PLs

- According to Abelson and Sussman, a good programming language must have four elements.
  - Primitives (built-in stuff such as integer numbers and functions)
  - Means of Combination (Expression, containment)
  - Means of Abstraction (naming, function)
  - Means of Capturing Common Patterns

Essential Elements

- Essential elements of a programming language:
  - Some primitive data types.
  - Expressions that allow one to compute new values.
  - Using variables to store data.
  - Various statements.
  - Abstraction (function).

Essential Elements

- Statements that contain instructions describing what a program does
  - Assignment
  - Flow control constructs (conditional & loops)
  - Input and Output
  - Data Types

Four concepts in programming

- If you can do these four things* you can write every program that has ever existed. The code won’t be pretty, but it will solve the problem.
  - Process data (assignment)
  - Make decision (if)
  - Loop (while, for)
  - Use indexed storage (arrays, lists)

* Platform independent