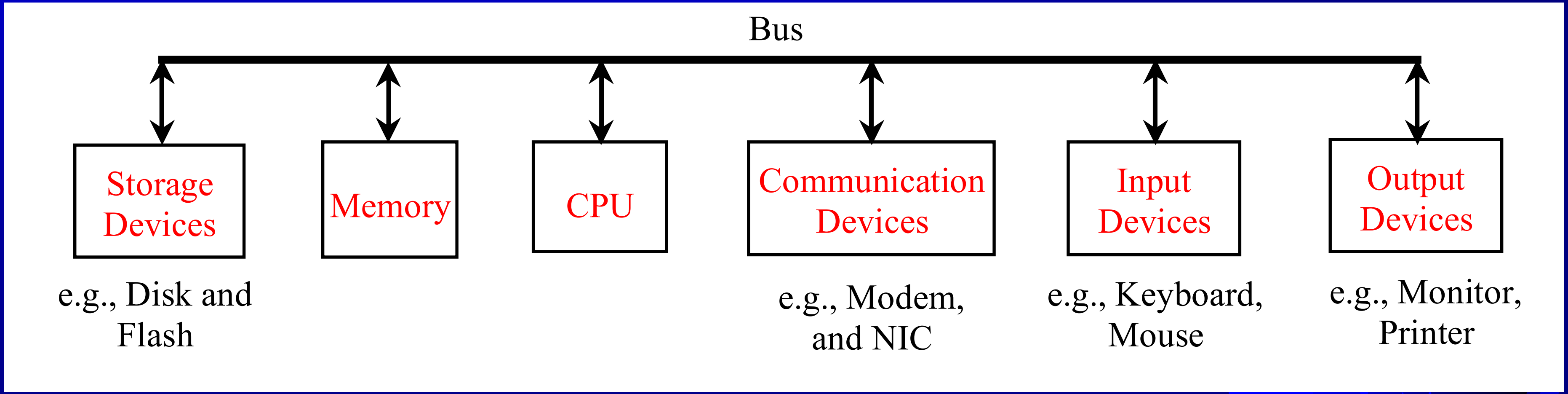
Chapter 1
Introduction to Computers, Programs, and Python

Objectives

- To understand computer basics, programs, and operating systems (§§1.2-1.4).
- To write and run a simple Python program (§1.5).
- To explain the basic syntax of a Python program (§1.5).
- To describe the history of Python (§1.6).
- To explain the importance of, and provide examples of, proper programming style and documentation (§1.7).
- To explain the differences between syntax errors, runtime errors, and logic errors (§1.8).
- To create a basic graphics program using Turtle (§1.9).

What is a Computer?

A computer consists of a CPU, memory, disk, input, output, and communication devices.

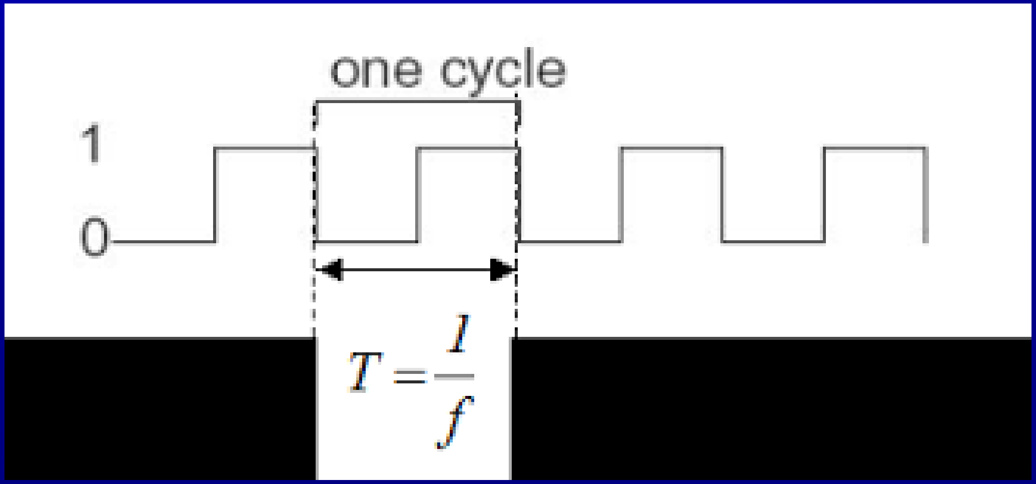


CPU

- Central processing unit (CPU)
- The "brain" of a computer
- Retrieves instructions from memory and executes them
- Speed is controlled by the computer clock

Clock

- Clock speed is measured in
  - megahertz (MHz) or 1 million pulses per second
  - gigahertz (GHz) or 1 billion pulses per second
- The speed of the CPU has been improved over time but is now plateauing at ~5 GHz

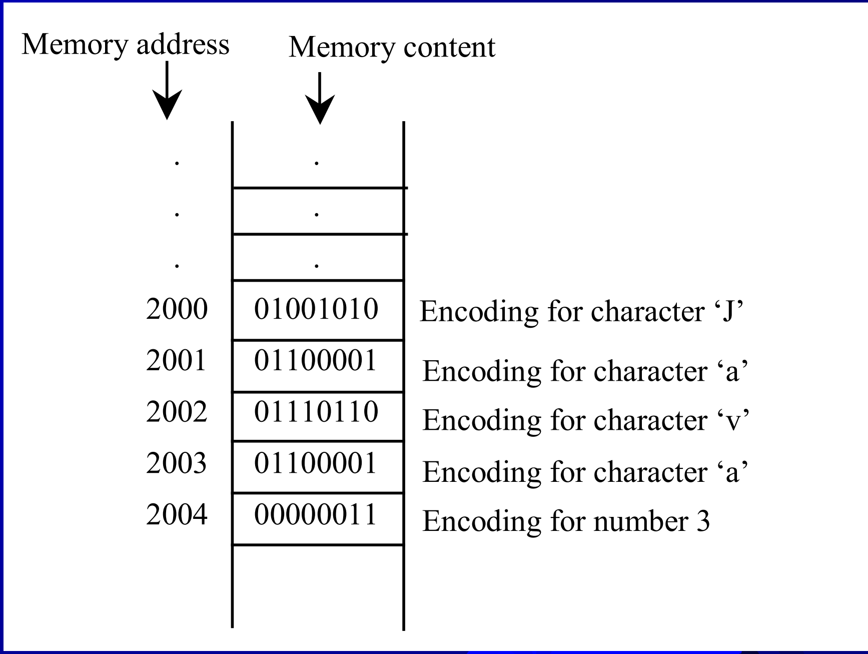


Memory (RAM)

- *S*tores data and program instructions for CPU to execute.
- An ordered sequence of bytes (eight bits).
- Each byte has an address
- Can be read or modified (written)
- A program and its data must be written into memory before they can be executed.

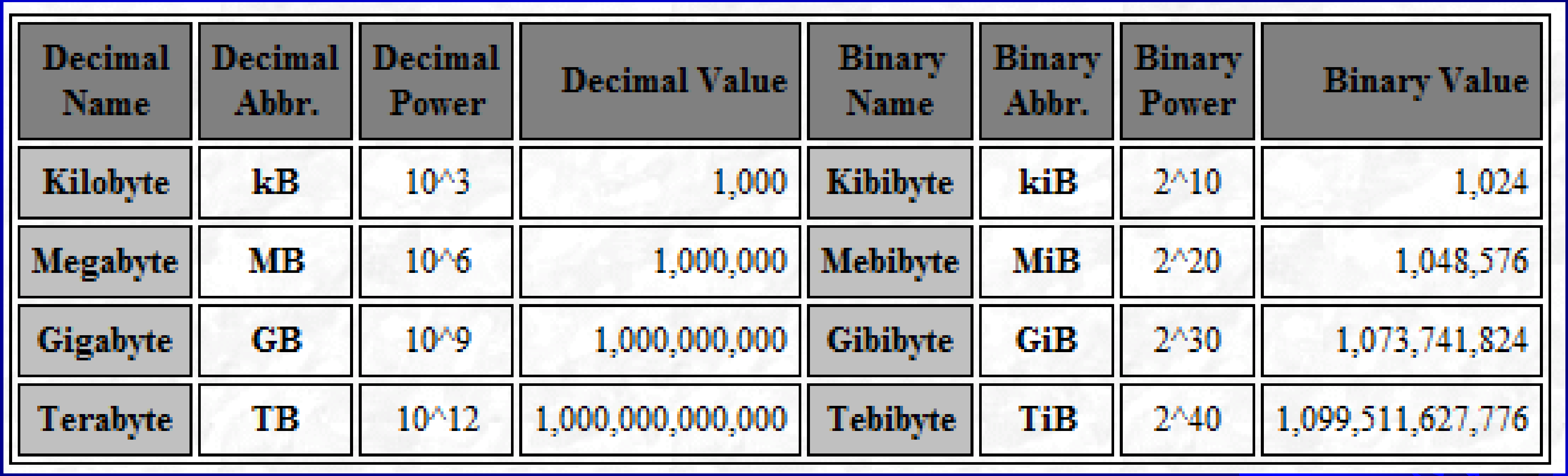
How Data is Stored?

- Physical or Circuitry Level
  - Absence or Presence of a Voltage
- Software or Logical Level
  - Binary Numbers
    - Strings of Bits
    - Zeros and Ones (0 and 1)
  - Byte
    - 8 bits
    - Minimum unit of storage
  - Letters
    - ASCII encoding



Data Units

- Bit (binary digit): A one or a zero
- Nibble: 4 bits
- Byte: 8 bits



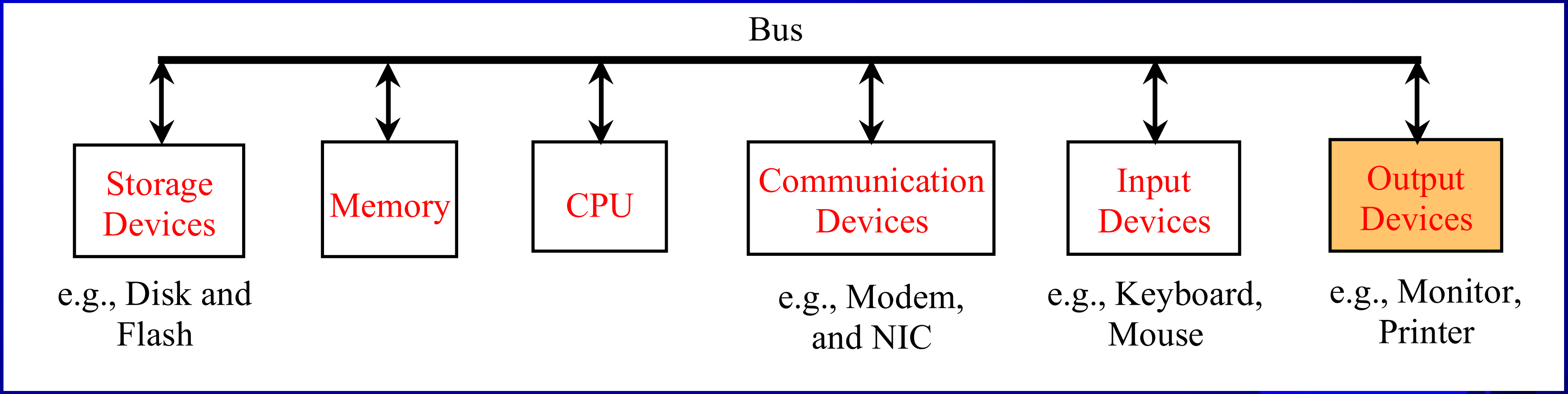
Decimal Name	Decimal Abbr.	Decimal Power	Decimal Value	Binary Name	Binary Abbr.	Binary Power	Binary Value
Kilobyte	kB	10^3	1,000	Kibibyte	kiB	2^10	1,024
Megabyte	MB	10^6	1,000,000	Mebibyte	MiB	2^20	1,048,576
Gigabyte	GB	10^9	1,000,000,000	Gibibyte	GiB	2^30	1,073,741,824
Terabyte	TB	10^12	1,000,000,000,000	Tebibyte	TiB	2^40	1,099,511,627,776

Storage Devices

Memory is volatile, because information is lost when the power is off. Programs and data are permanently stored on storage devices and are moved to memory when the computer actually uses them. There are two modern types of storage devices: disk drives (SSD and hard disks), and Flash drives.

Output Devices: Monitor

The monitor displays information (text and graphics). The resolution and dot pitch determine the quality of the display.

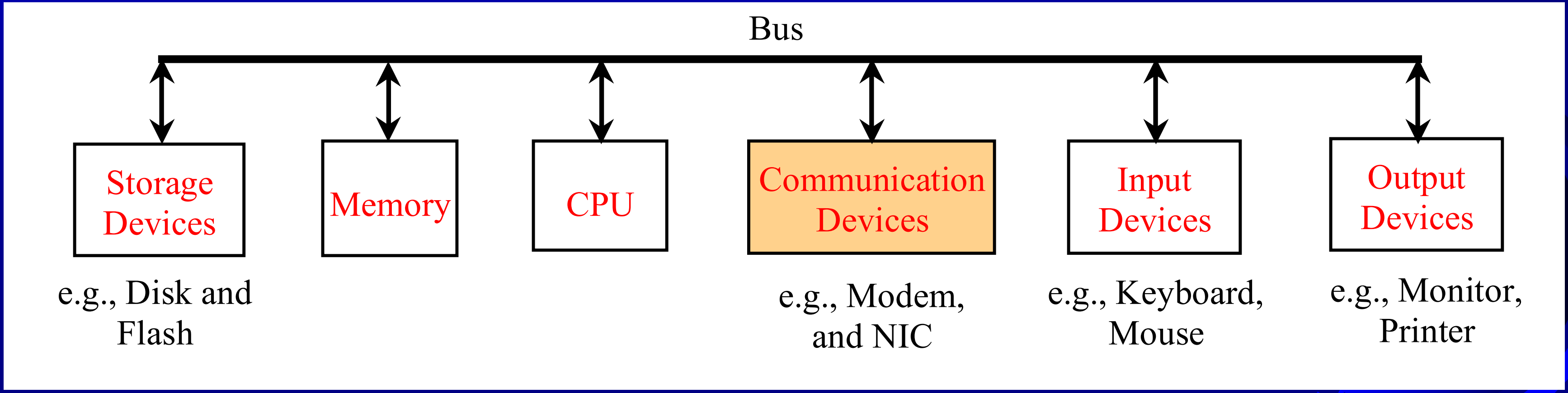


Monitor Resolution and Dot Pitch

resolution	The *resolution* specifies the number of pixels per square inch. Pixels (short for “picture elements”) are tiny dots that form an image on the screen. The higher the resolution, the sharper and clearer is the image. For example a 4K monitor 3840×2160 (8,294,400 pixels).
dot pitch	The *dot pitch* is the amount of space between pixels. The smaller the dot pitch, the better the display.

Communication Devices

A *DSL* (digital subscriber line) also uses a phone line and can transfer data in a speed 20 times faster than a regular modem. A *cable modem* uses the TV cable line maintained by the cable company. A cable modem is faster than DSL. Network interface card (*NIC*) is a device to connect a computer to a local area network (LAN). The LAN is commonly used in business, universities, and government organizations. A typical type of NIC, can transfer data at 1 Gbps (billion bits per second).



Programs

Computer *programs*, known as *software*, are instructions to the computer.

You tell a computer what to do through programs. Without programs, a computer is an empty machine. Computers do not understand human languages, so you need to use computer languages to communicate with them.

Programs are written using programming languages.

Programming Languages

Machine Language   Assembly Language   High-Level Language

Machine language is a set of primitive instructions built into every computer. The instructions are in the form of binary code, so you have to enter binary codes for various instructions. Program with native machine language is a tedious process. Moreover the programs are highly difficult to read and modify. For example, to add two numbers, you might write an instruction in binary like this:

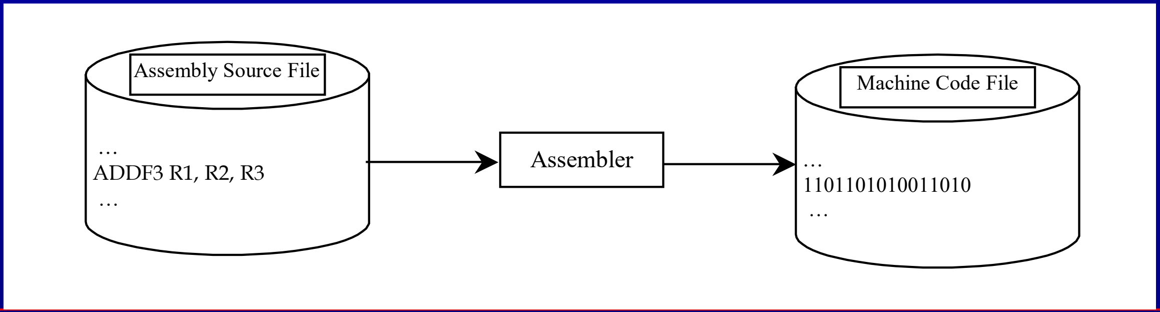
```
1101101010011010
```

Programming Languages

Machine Language Assembly Language High-Level Language

Assembly language is a human readable version of machine code and are rather cryptic. The computer cannot understand assembly language directly, so a program called assembler is used to convert assembly language programs into machine code. For example, to add two numbers, you might write an instruction in assembly code like this:

ADD EAX,EBX



Programming Languages

Machine Language   Assembly Language   High-Level Language

The high-level languages are English-like and easy to learn and program. For example, the following is a high-level language statement that computes the area of a circle with radius 5:

```
area = 5 * 5 * 3.1415;
```

Compiled vs Interpreted

- Interpreted languages (such as Python) need a layer of software that converts to machine code line by line as the program runs
- Compiled languages use a compiler that converts all the code to machine code in one step.
- Compiled languages are generally much faster than interpreted ones.

Popular High-Level Languages

- Python (interpreted language named after Monty Python)
- C (compiled language)
- Java (interpreted language
- JavaScript (interpreted language)
- C++ (compiled language, based on C)
- C# (C++ like language developed by Microsoft)

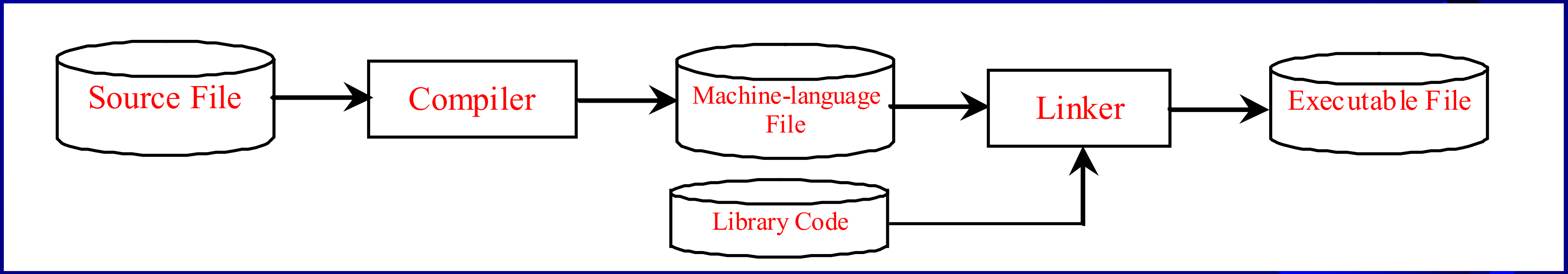
Language Popularity

TIOBE Index

Dec 2021	Dec 2020	Change	Programming language	Ratings	Change
1	3	↑	Python	12.90%	+0.69%
2	1	↓	C	11.80%	-4.69%
3	2	↓	Java	10.12%	-2.41%
4	4		C++	7.73%	+0.82%
5	5		C#	6.40%	+2.21%
6	6		Visual Basic	5.40%	+1.48%
7	7		JavaScript	2.30%	-0.06%
8	12	↑↑	Assembly language	2.25%	+0.91%
9	10	↑	SQL	1.79%	+0.26%
10	13	↑	Swift	1.76%	+0.54%
11	9	↓	R	1.58%	-0.01%
12	8	↓↓	PHP	1.50%	-0.62%
13	23	↑↑	Classic Visual Basic	1.27%	+0.56%
14	11	↓	Groovy	1.23%	-0.30%
15	15		Ruby	1.16%	-0.01%
16	18	↑	Delphi/Object Pascal	1.14%	+0.27%
17	32	↑↑	Fortran	1.04%	+0.59%
18	14	↓↓	Perl	0.96%	-0.24%
19	16	↓	Go	0.95%	-0.19%
20	17	↓	MATLAB	0.92%	-0.18%

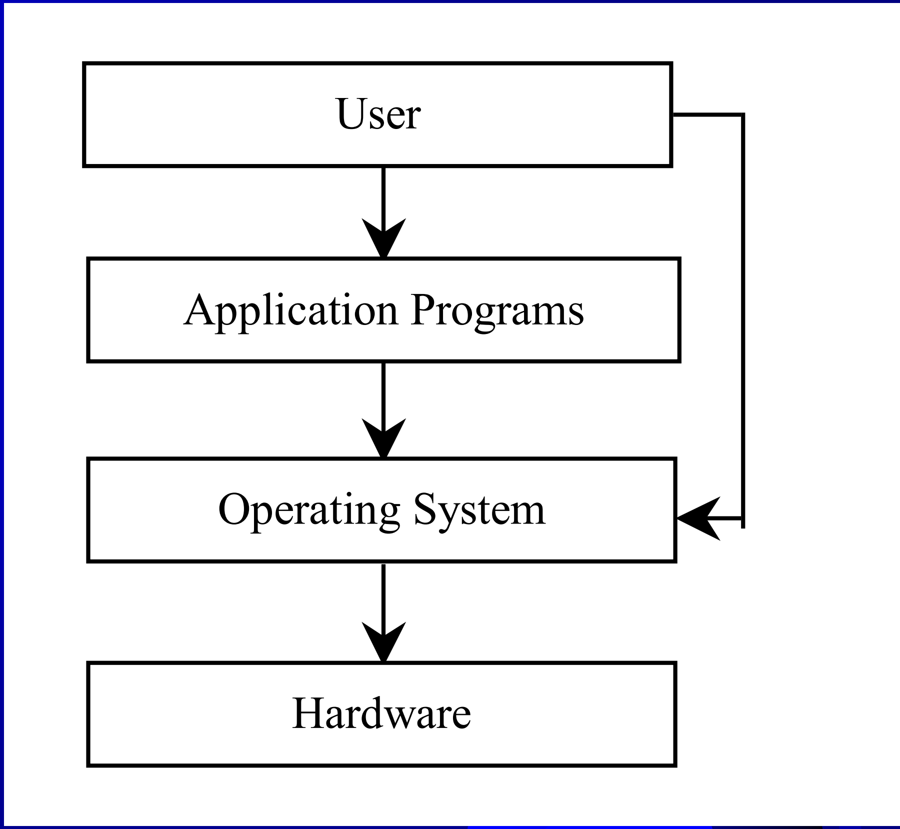
Compiling Source Code

A program written in a high-level language is called a *source program*. Since a computer cannot understand a source program. Program called a *compiler* is used to translate the source program into a machine language program called an *object program*. The object program is often then linked with other supporting library code before the object can be executed on the machine.



Operating Systems

The *operating system* (OS) is a program that manages and controls a computer's activities. Windows is currently the most popular PC operating system. Application programs such as an Internet browser and a word processor cannot run without an operating system.



What is Python?

General Purpose Interpreted Object-Oriented

Python is a general purpose programming language. That means you can use Python to write code for any programming tasks. Python are now used in Google search engine, in mission critical projects in NASA, in processing financial transactions at New York Stock Exchange.

What is Python?

General Purpose Interpreted Object-Oriented

Python is interpreted, which means that python code is translated and executed by an interpreter one statement at a time. In a compiled language, the entire source code is compiled and then executed altogether.

What is Python?

General Purpose   Interpreted   Object-Oriented

Python is an object-oriented programming language. Data in Python are objects created from classes. A class is essentially a type that defines the objects of the same kind with properties and methods for manipulating objects. Object-oriented programming is a powerful tool for developing reusable software.

Python's History

- created by Guido van Rossum in Netherlands in 1990
- Open source
- Continually being updated with new features

Python 2 vs. Python 3

Python 3 is a newer version, but it is not backward compatible with Python 2. That means if you write a program using Python 2, it may not work on Python 3.

A Simple Python Program

Listing 1.1

```
# Display two messages
print("Welcome to Python")
print("Python is fun")
```

Welcome

Run

animation

Trace a Program Execution

Execute a statement

```
# Display two messages
print("Welcome to Python")
print("Python is fun")
```

animation

Trace a Program Execution

Execute a statement

```
# Display two messages
print("Welcome to Python")
print("Python is fun")
```

Two More Simple Examples

WelcomeWithThreeMessages

Run

ComputeExpression

Run

Anatomy of a Python Program

- Statements
- Comments
- Indentation

Statement

A statement represents an action or a sequence of actions. The statement print("Welcome to Python") in the program in Listing 1.1 is a statement to display the greeting "Welcome to Python“.

```
# Display two messages
print("Welcome to Python")
print("Python is fun")
```

Indentation

The indentation matters in Python. Note that the statements are entered from the first column in the new line. It would cause an error if the program is typed as follows:

```
# Display two messages
 print("Welcome to Python")
print("Python is fun")
```

Special Symbols

Character	Name	Description
()	Opening and closing parentheses	Used with functions.
#	Pound sign	Precedes a comment line.
" "	Opening and closing quotation marks	Enclosing a string (i.e., sequence of characters).
''' '''	Opening and closing quotation marks	Enclosing a multiline paragraph comment.

Programming Style and Documentation

- Appropriate Comments
- Proper Indentation and Spacing Lines

Appropriate Comments

Include the entire problem description at the top of the code page.

Comment significant lines of code. Therefore, most lines should have a comment after them. You need to make it clear that you understand what is happening in the code.

Proper Indentation and Spacing

- Indentation
  - Indent four spaces.
  - A consistent spacing style makes programs clear and easy to read, debug, and maintain.

- Spacing
  - Use blank line to separate segments of the code.

Programming Errors

- Syntax Errors
  - Error in code construction
- Runtime Errors
  - Causes the program to abort
- Logic Errors
  - Produces incorrect result