Computer Skills

Instructor: Dr. Murat Tunc

Why Programming?

Introduction to Computers, Programming, and Python

Instructor: Dr. Murat Tunc

Lecture 1

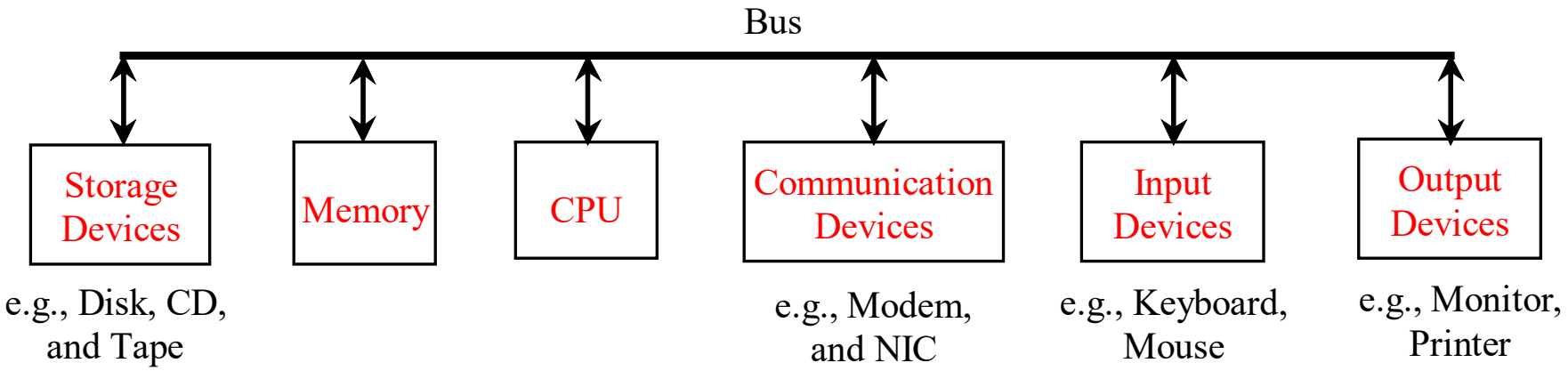
What is a Computer?

- Electronic device that stores and processes data
- Consists of **hardware** and **software**
- Importance of knowing hardware
  - Effect of a program on the computer

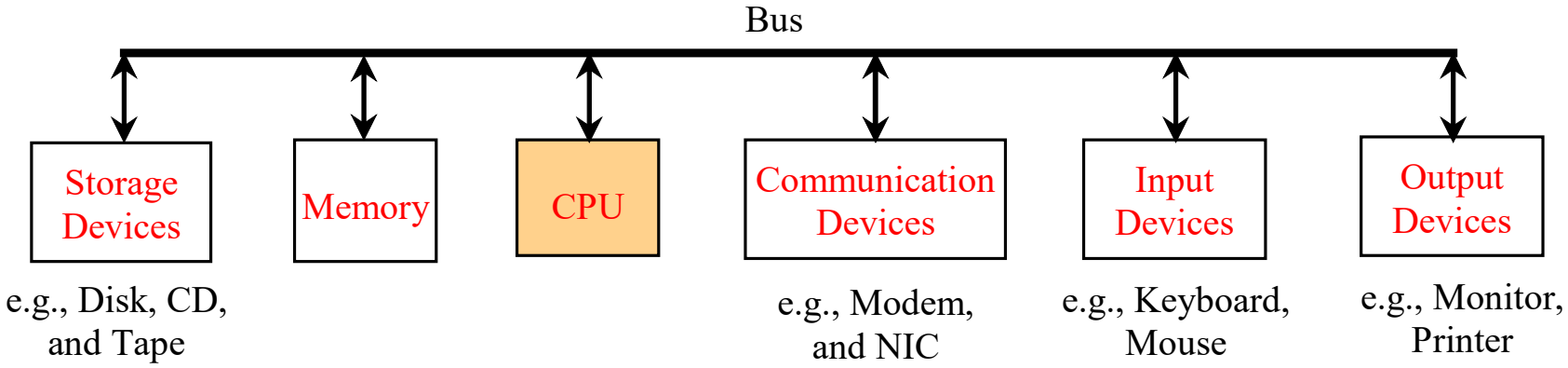
Computer Hardware

Bus

- Computer's components are interconnected by a subsystem called a bus



Central Processing Unit (CPU)



- Computer's brain
- Retrieves instructions from memory and executes them
- 2 units: Control Unit and Arithmetic/Logic Unit

Central Processing Unit (CPU)

- CPU clock speed is measured in gigahertz (GHz)
- Latest CPUs clock around 4 GHz

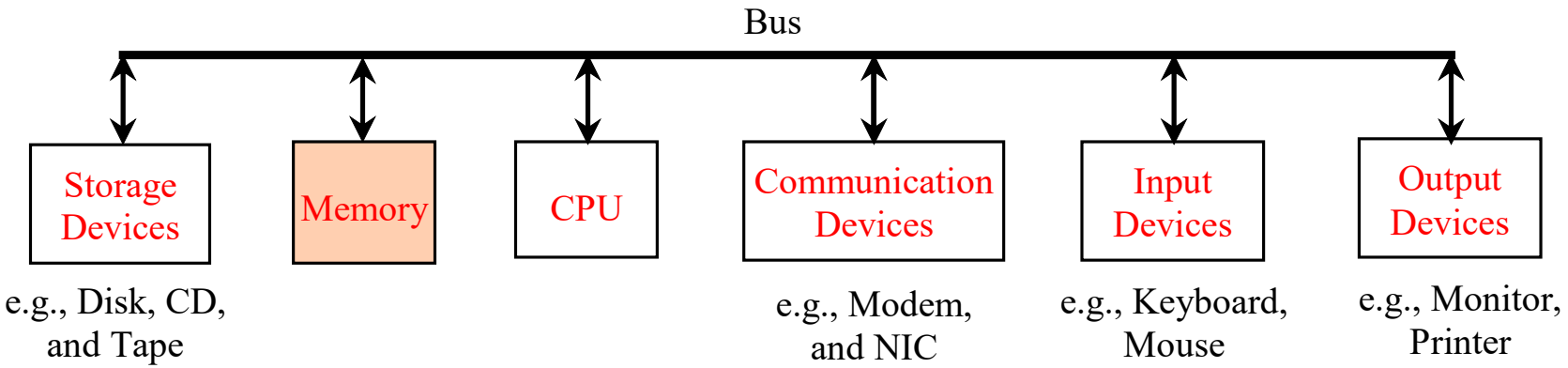
Bits and Bytes

- Computer – series of switches
- **Two stable states**: on (1) or off (0)
- **Bits** (**bi**nary digi**ts**)
- Data of various kinds, such as numbers, characters, and strings, are encoded as a series of bits
- Minimum storage unit in a computer is a byte
- A **byte** is composed of 8 bits
- For example, character 'J' is represented by 01001010 in one byte

Bits and Bytes

- 1 kilobyte (**KB**) is about 1,000 bytes
- 1 megabyte (**MB**) is about 1,000 KB
- 1 gigabyte (**GB**) is about 1,000 MB
- 1 terabyte (**TB**) is about 1,000 GB

Memory


Bus
Storage Devices
Memory
CPU
Communication Devices
Input Devices
Output Devices
e.g., Disk, CD, and Tape
e.g., Modem, and NIC
e.g., Keyboard, Mouse
e.g., Monitor, Printer

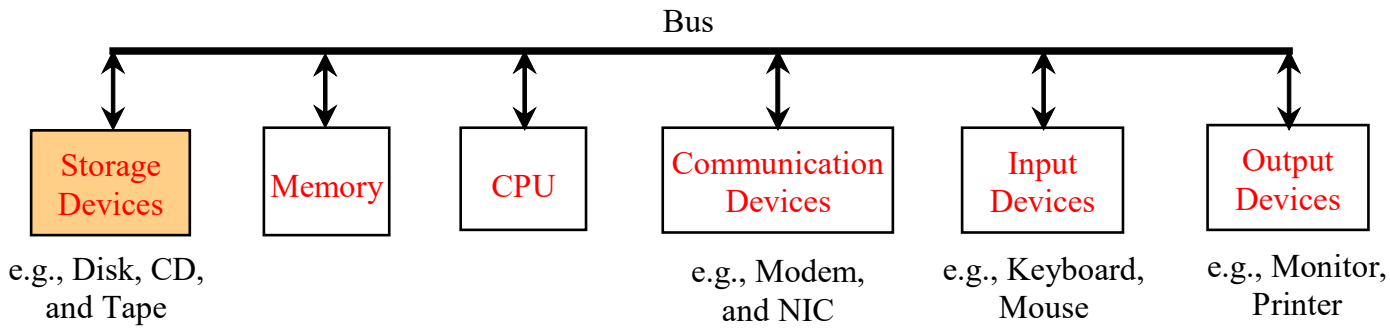
Memory

- Memory is to store data and program instructions for the CPU to execute

- A memory unit is an **ordered** sequence of bytes
- As the bytes in the memory can be accessed in any order, the memory is referred to as **random-access memory** (RAM)

- A program and its data must be brought to memory before they can be executed

Storage Devices



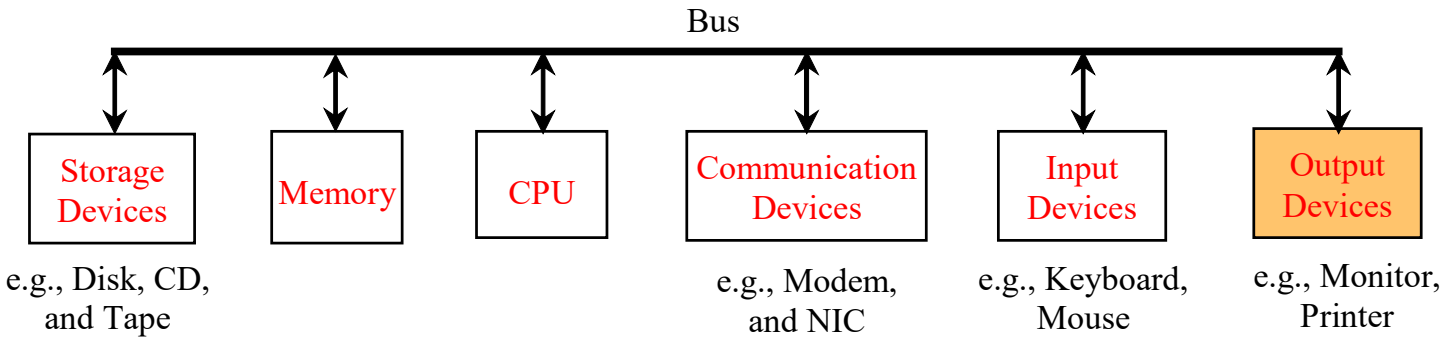
Storage Devices

- Computer's memory (RAM) is **volatile**: Information is lost when the system's power is turned off

- Programs and data are **permanently stored** on storage devices and are moved to memory when the computer actually uses them

  - Hard Disks
  - CDs and DVDs
  - USB Flash Drives

Input and Output Devices



Output Device: Monitor

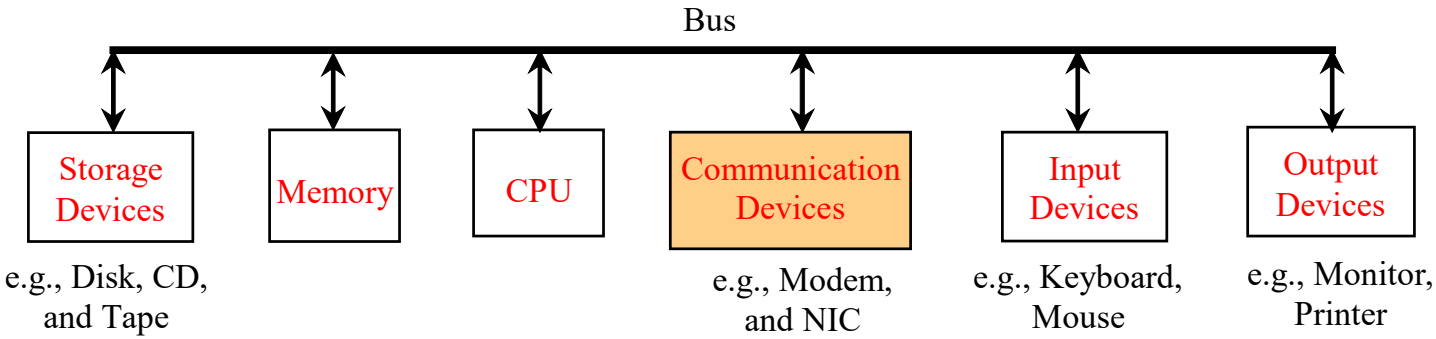
- The monitor displays information (text and graphics)
- Pixels are tiny dots that form an image on the screen
- The **resolution** and **dot pitch** determine the quality of the display

Output Device: Monitor

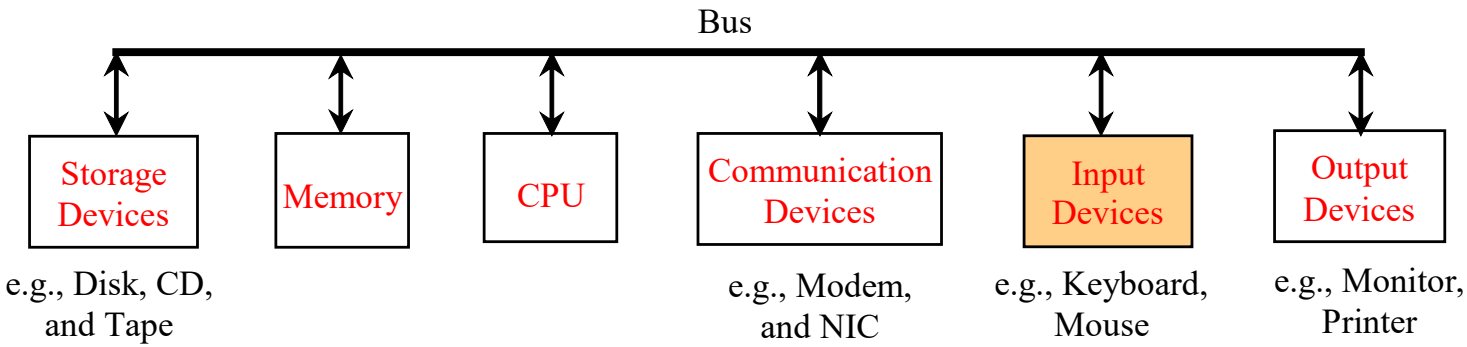
- **Resolution** specifies the number of pixels in horizontal and vertical dimension of the display device
  - The higher the resolution, the sharper and clearer the image is
  - A common resolution for a 17-inch screen, for example, is 1,024 pixels wide and 768 pixels high

- **Dot pitch**: Amount of space between pixels
  - The smaller the dot pitch, the sharper the display

Communication Devices



Input Devices



Computer Software

Programming Languages

- Computer programs, known as software, are **instructions** that tell a computer what to do
- Without programs, a computer is an empty machine
- Programs are written using **programming languages**

Programming Languages

- Machine Language
- Assembly (Low-Level) Language
- High-Level Language

Programming Languages

Machine Language

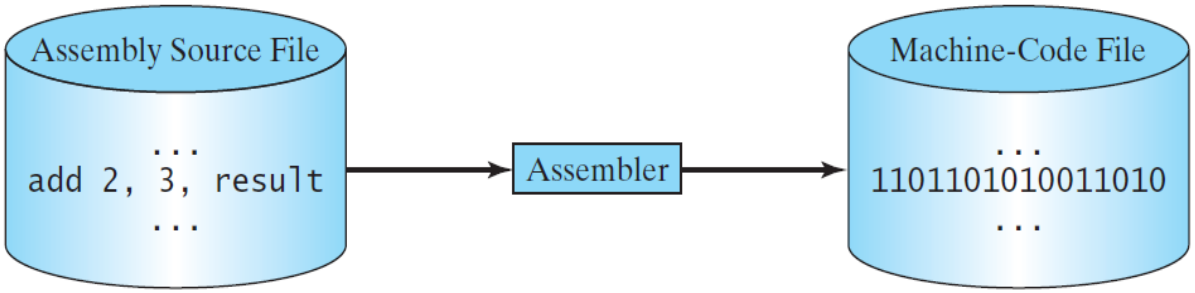
- The instructions are in the form of **binary code**, so you have to enter binary codes for various instructions
- The programs are highly **difficult to read** and modify
- For example, to add two numbers, you might write an instruction in binary like this:

1101101010011010

Programming Languages

Assembly Language (Low-Level Language)

- Assembly languages were developed to make programming easy
  - Example: add 2, 3, result
- The computer cannot understand assembly language, however, a program called **assembler** is used to convert assembly language programs into machine code



Programming Languages

High-Level Language

- 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;
```

Programming Languages

High-Level Language

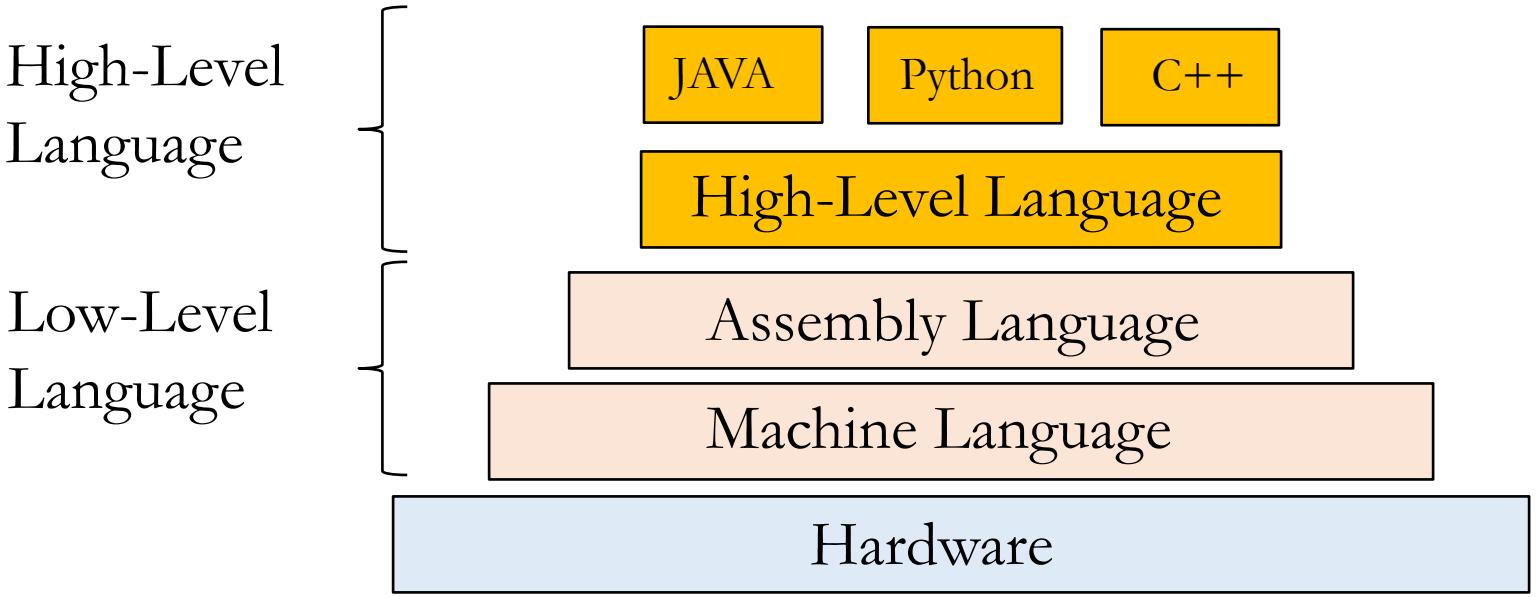
- Popular high-level programming languages
  - Java
  - Visual Basic
  - Python
  - FORTRAN
  - Pascal
  - C
  - C++
  - C#

Programming Languages

Distinction between programming languages

- There is **no formal distinction** between high- and low-level language
- A low-level language is characterized by its closeness to hardware
- A **low-level language** directly interacts with hardware, whereas a **high-level language** needs many intermediaries to interact with hardware

Programming Languages



Programming Languages

High-Level Languages

- **Benefit**: Easy to use
- **Drawback**: Memory and speed optimization are handled by compiler. Thus, less flexible
- Ex: Java, Python

Low-Level Languages

- **Drawback**: Difficult to use
- **Benefit**: Memory and speed optimization are handled by the programmer. Thus, more flexible
- Ex: x86 assembly language

Programming Languages

Why are high-level languages becoming popular?

- Memory and CPU power (clock speed) are getting cheaper
- Compilers are now smart enough to perform memory and speed optimization

Interpreting/Compiling Source Code

Source Code (or Source Program):

- A program written in a high-level language is called a **source code** or a source program

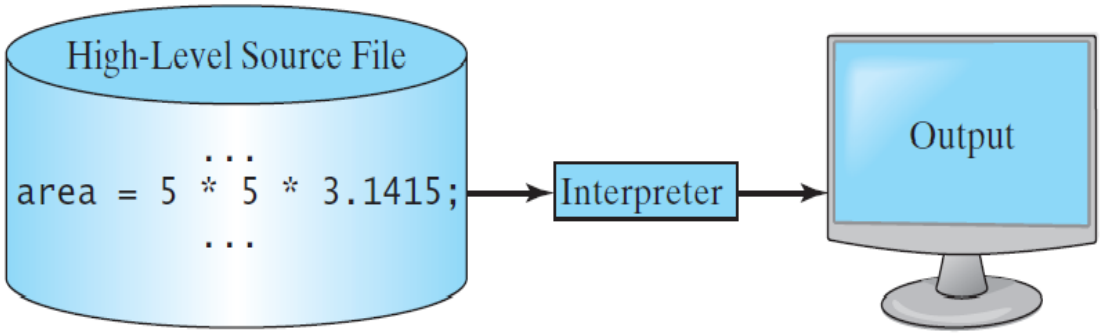
Compiler/Interpreter:

- Computer cannot understand source code
- Needs to be translated into machine code for execution
- The translation can be done using another programming tool called a **compiler** or an **interpreter**

Interpreting/Compiling Source Code

Interpreter:

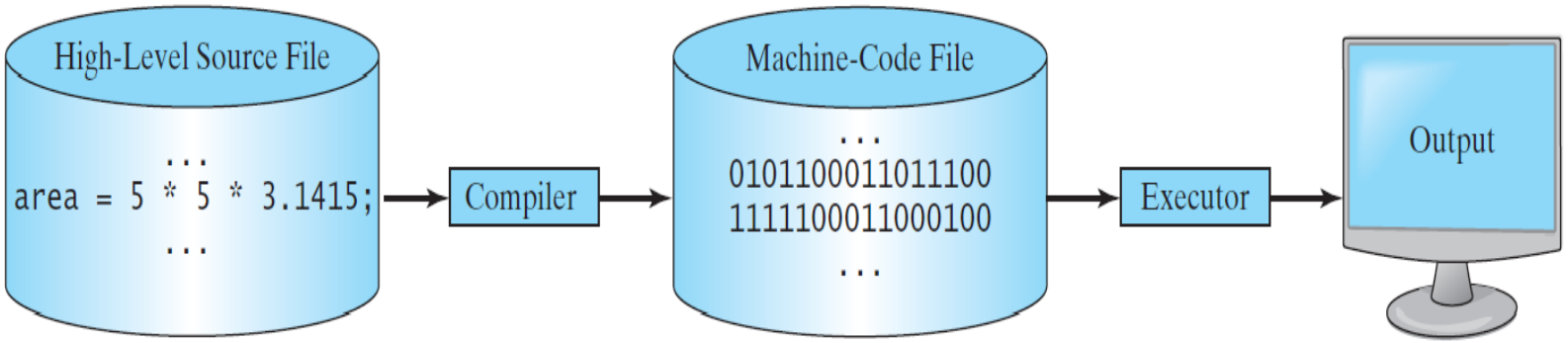
- An interpreter reads **one statement** from the source code, translates it to the machine code, and then executes it right away



Interpreting/Compiling Source Code

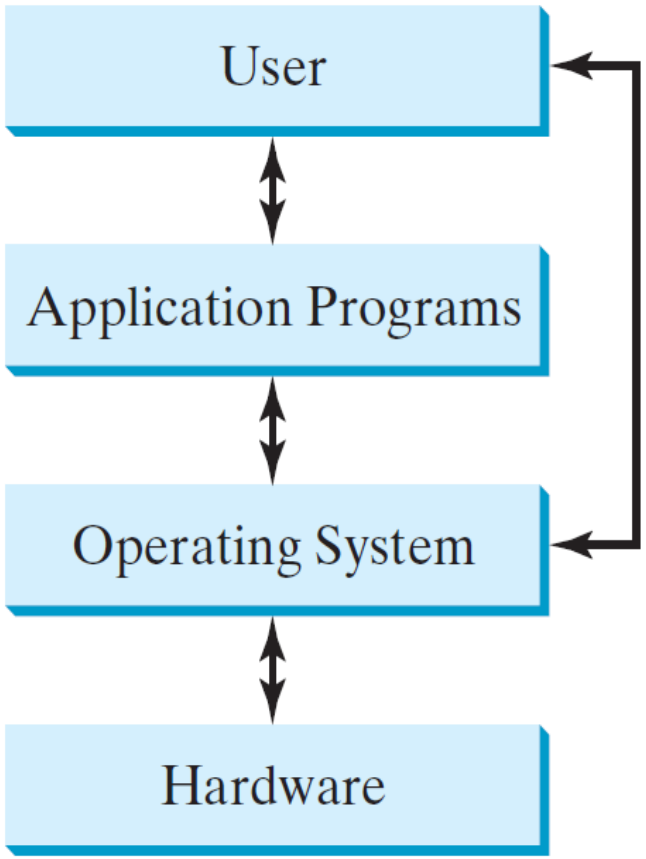
Compiler:

- A compiler translates **the entire source code** into a machine-code file, and the machine-code file is then executed



Operating System (OS)

- Program that manages and controls a computer's activities
- Popular OS: Microsoft Windows, Mac OS, and Linux
- Application programs, such as a Web browser or a word processor, cannot run unless an OS is installed and running on the computer



Review

- **Q:** A bit is a sequence of 8 bytes? (T/F)
- **A:** False

- **Q:** We can store data permanently in storage devices? (T/F)
- **A:** True

- **Q:** Assembly language is a low-level programming language? (T/F)
- **A:** True

- **Q:** Only numeric data is stored as binary? (T/F)
- **A:** False

- **Q:** We can store data permanently in RAM? (T/F)
- **A:** False

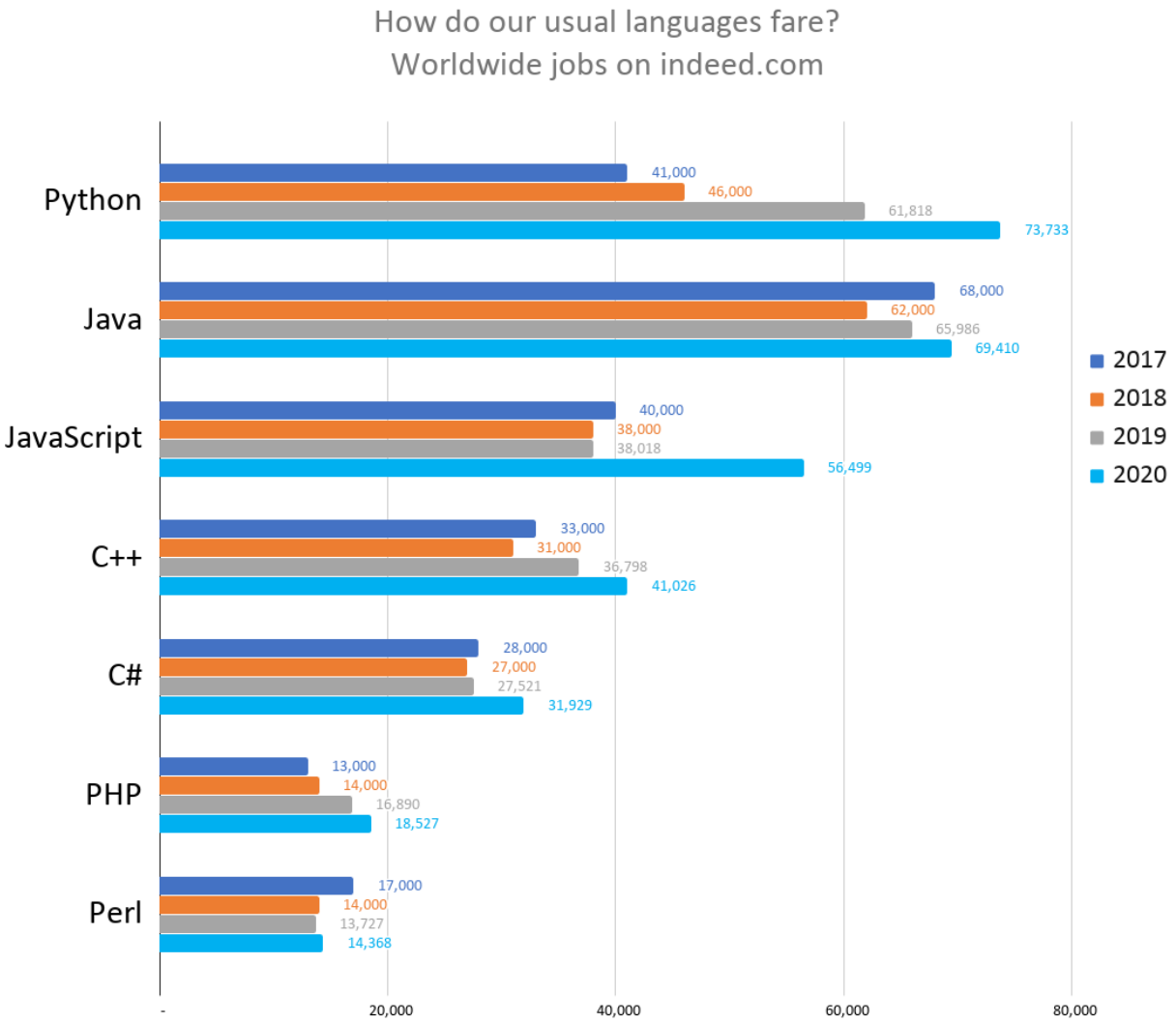
- **Q:** A high-level language is closer to hardware in terms of direct interaction than a low-level language? (T/F)
- **A:** False

- **Q:** Which is an example of a code in machine language?
  - area = 5 * 5 * 3.1415
  - 1101101010011010
  - Both of the above
  - None of the above

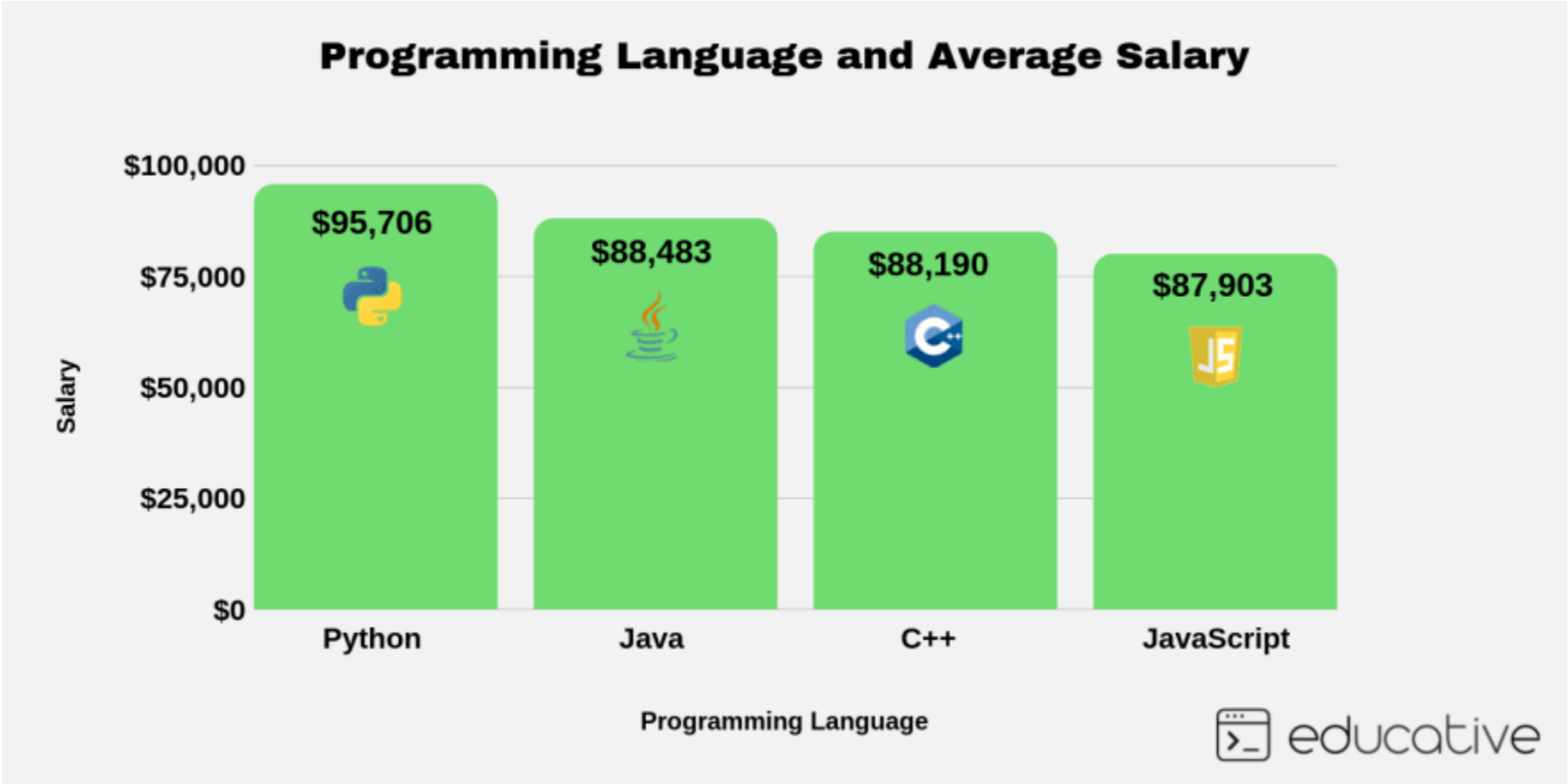
- **A:** B

Why Python?

In-demand


How do our usual languages fare?
Worldwide jobs on indeed.com
Python
41,000
46,000
61,818
73,733
Java
68,000
62,000
65,986
69,410
JavaScript
40,000
38,000
38,018
56,499
C++
33,000
31,000
36,798
41,026
C#
28,000
27,000
27,521
31,929
PHP
13,000
14,000
16,890
18,527
Perl
17,000
14,000
13,727
14,368
2017
2018
2019
2020
-
20,000
40,000
60,000
80,000

Job prospects: $$$



Download and Installation

Where to write Python programs?

- Any text editor can be used to create and edit Python programs
- We will use an **integrated development environment** (IDE) to write Python programs
- IDE provides **comprehensive facilities** to computer programmers – source code editor, automation tools, debugger, etc.
- Popular IDEs
  - PyCharm
  - **Anaconda** ✓

Anaconda

- **Download:** Anaconda Distribution (includes Python 3.8 + popular packages and modules)
- Download link: https://docs.anaconda.com/anaconda/install/
  - For Windows: https://docs.anaconda.com/anaconda/install/windows/
  - For Mac OS X: https://docs.anaconda.com/anaconda/install/mac-os/
  - For Linux / UNIX: https://docs.anaconda.com/anaconda/install/linux/

Anaconda Navigator

- **Anaconda Navigator** is automatically installed with Anaconda Distribution
- Open Anaconda Navigator:
  - On **Windows**, the installer will create a Start menu shortcut for Navigator
  - On **macOS**, if using the GUI (.pkg) installer, you'll get an icon for Navigator in Launchpad
  - On **Linux** or macOS installed via .sh installer, open a terminal and enter this command: anaconda-navigator

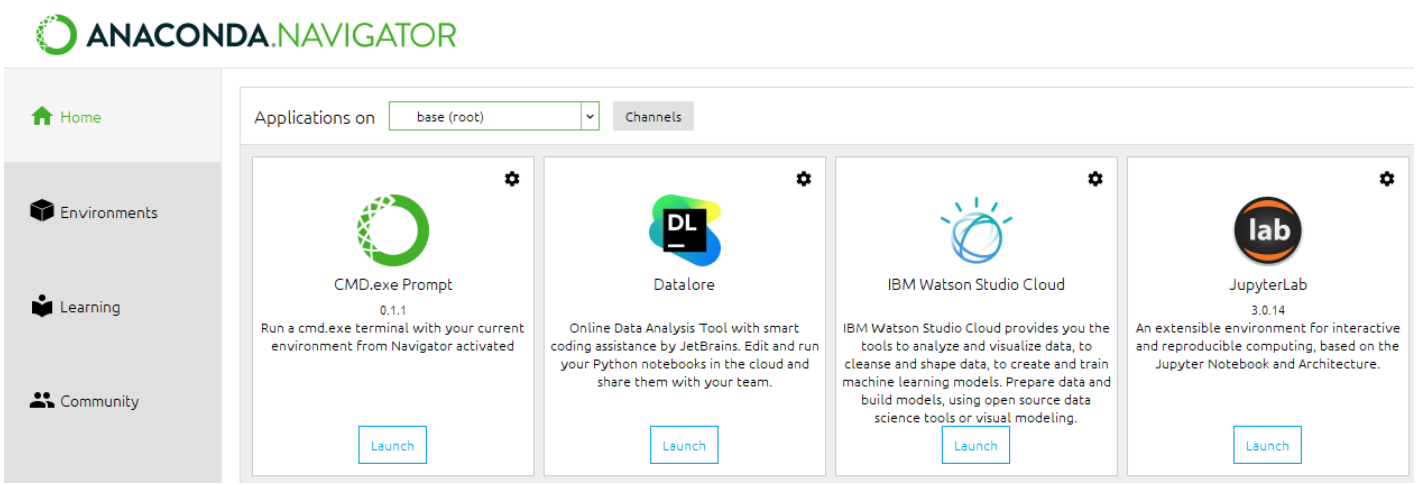
“Hello World!”

Writing a program to display “Hello World!” in the console

“Hello World!” Program

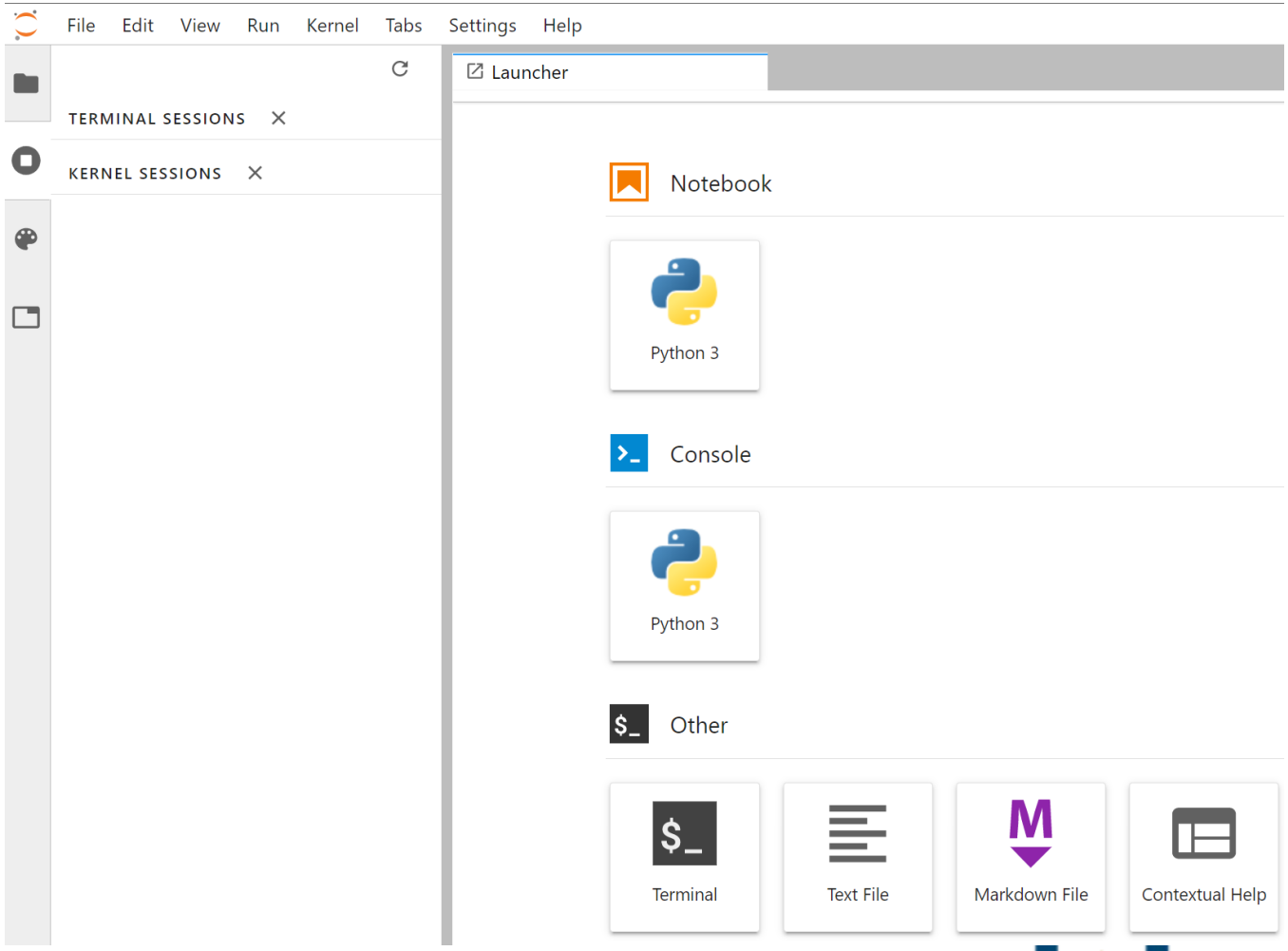
Open → Anaconda Navigator

Click on → JupyterLab



“Hello World!” Program

Under Notebook → **Click** on → Python 3



“Hello World!” Program

```
# This program prints Hello World!

print("Hello World!")
```

Anatomy of a Python Program

- Statements
- Comments
- Special Symbols

Statement

- A **statement** represents an action or a sequence of actions
- The statement **print**(**"Hello World!"**) in the program is a statement to display the greeting "Hello World!"

```
# This program prints Hello World!

print("Hello World!")
```

Comments

- Line 1 (in **green color**) is a **comment** that documents what the program is and how it is constructed

- They are not programming statements, and thus are **ignored** by the compiler

```
# This program prints Hello World!

print("Hello World!")
```

Comments

- Line Comment
  - In Python, comments are preceded by pound sign (#) on a line, called a line comment
- Example: **# This program prints Hello World!**

- Block Comment (or Paragraph Comment)
  - In Python, select multiple lines and press **ctrl** and **/**
- Example:

 # This program prints Hello World!

 # This program …..

 # This program….

Special Symbols

- ( ) i.e. Opening and closing parentheses
  - Used with functions and methods
- # i.e. Pound sign
  - Precedes a comment line
- “ ” i.e. Opening and closing double quotation marks
  - Enclosing a string (i.e. a series of characters)

```
# This program prints Hello World!

print("Hello World!")
```

Special Symbols

(…)

```
# This program prints Hello World!

print("Hello World!")
```

Special Symbols

#

```
# This program prints Hello World!

print("Hello World!")
```

Special Symbols

" ... "

```
# This program prints Hello World!

print("Hello World!")
```

Programming Style and Documentation

- Appropriate Comments
  - A summary at the beginning of the program and other appropriate positions to **explain what the program does**, its key features, its supporting data structures, and any unique techniques it uses
  - Good habit to include your name, class section, date, and a brief description at the beginning of the program
- Naming Conventions
  - Choose **meaningful and descriptive** names
- Proper Indentation and Spacing Lines

Programming Errors

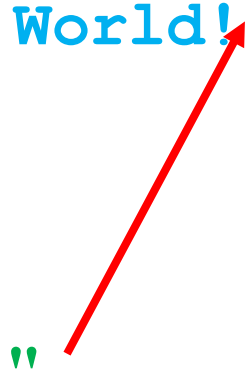
- 1) Syntax Errors
  - **Detected** by the compiler
- 2) Logic Errors
  - Produce **incorrect results**

Programming Errors

Syntax Error

```
# This program prints Hello World!

print("Hello World!)
```



"

Programming Errors

Logic Error

```
# This program prints the average of 3 + 4

print("Average of 3 and 4 is ")

print(3 + 4 / 2)
```

Output: Average of 3 and 4 is 5
Correct output: 3.5
Correct way: (3+4)/2 = 3.5

Review

```
# This program prints Hello World!

print("Hello World!"
```

Ans: Missing a paranthesis

print

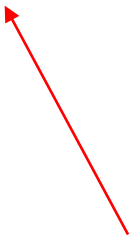
```
# This program prints Hello World!

pirnt("Hello World!")
```

Ans: Misspelling Names

```
# This program prints Hello World!

print("Hello Wrld!")
```



Logic Error

Ans: Prints a misspelled sentence