# Hello World

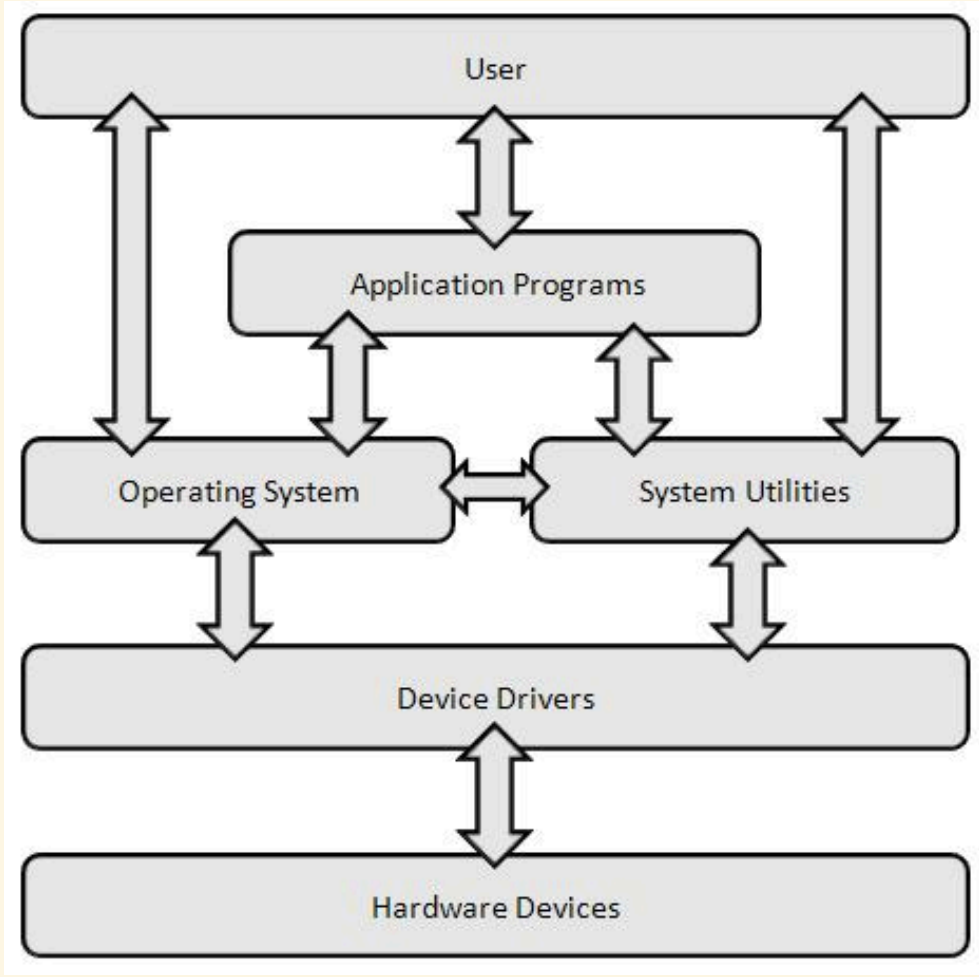
A Quick Introduction to Computers, Programming, and Python

# What Really is a Computer?

- **Electronic machine** that follows instructions
- Takes **input** → processes it → gives **output**
- Only understands **1s and 0s** (binary)
- Everything else is translation

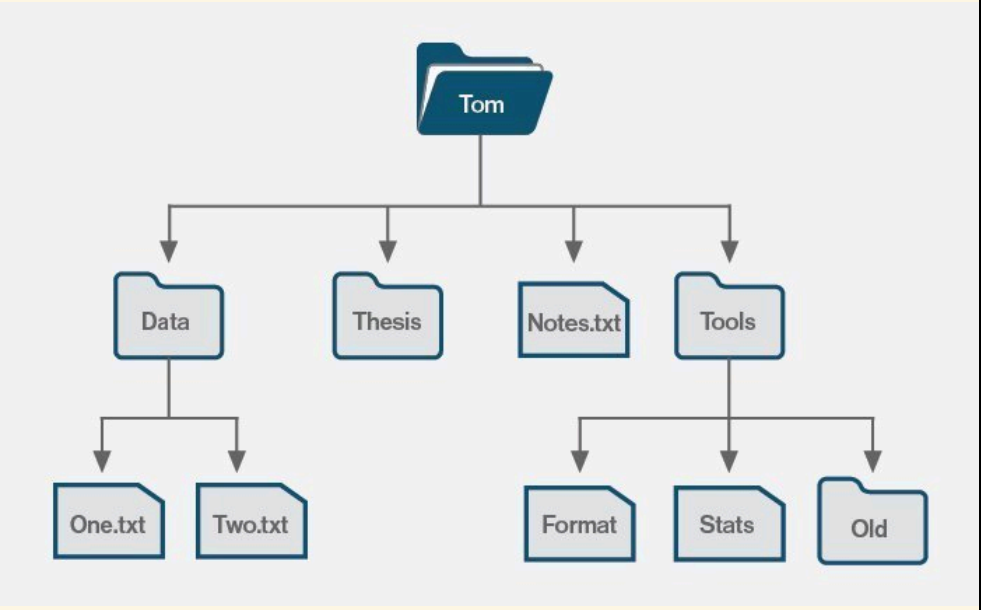
# Modern Computers are Layers of Abstraction

- **Hardware**: Physical components (CPU, RAM, etc.)
- **Operating System**: Manages hardware (Windows, macOS, Linux)
- **Programming Languages**: Human-readable instructions (Python, Java, etc.)
- **Applications**: Software that performs tasks (web browsers, games, etc.)



# What is a File?

- **Collection of data** stored on your computer in a **tree-like** structure
- Has a **path**, **name**, and **extension**
- File extensions indicate type:
  - `.txt` = text
  - `.py` = Python code
  - `.jpg` = image
- Files can be `created`, `read`, `written`, and `deleted`
- Some files can be `executed`, these are programs.



# Source Code vs Machine Code

## Source Code

- **Human-readable** instructions we write
- Looks like:

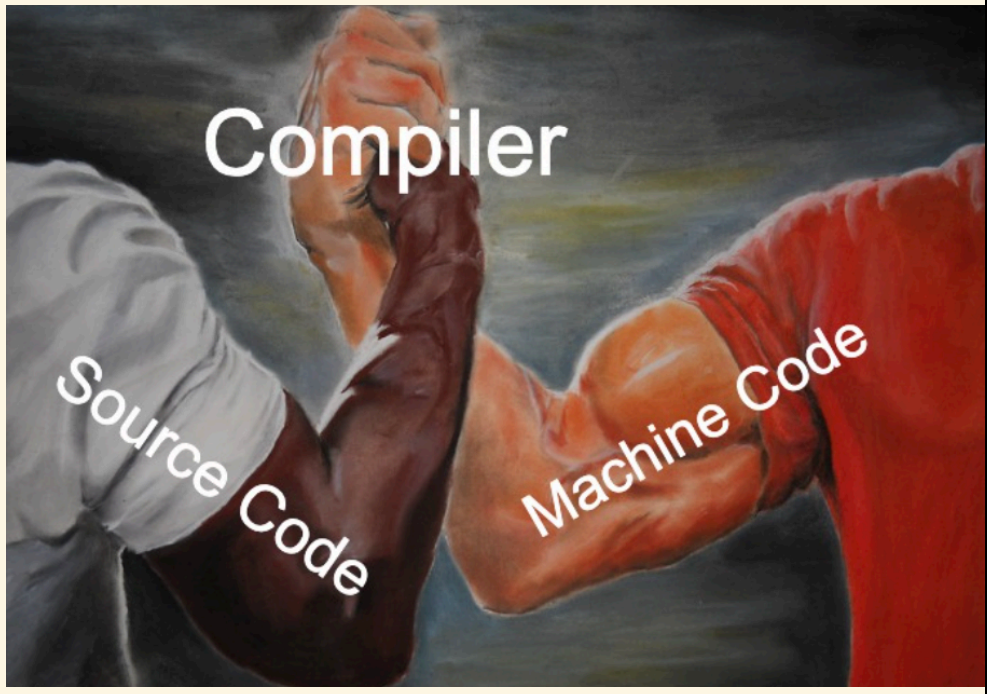
```
print("Hello World")
```

## Machine Code

- **Computer-readable** instructions
- Looks like:

```
48 89 e5     mov    %rsp,%rbp
48 83 ec 10  sub    $0x10,%rsp
bf 01 00 00  mov    $0x1,%edi
```

- **Source code** is translated to **machine code** by a **compiler**.

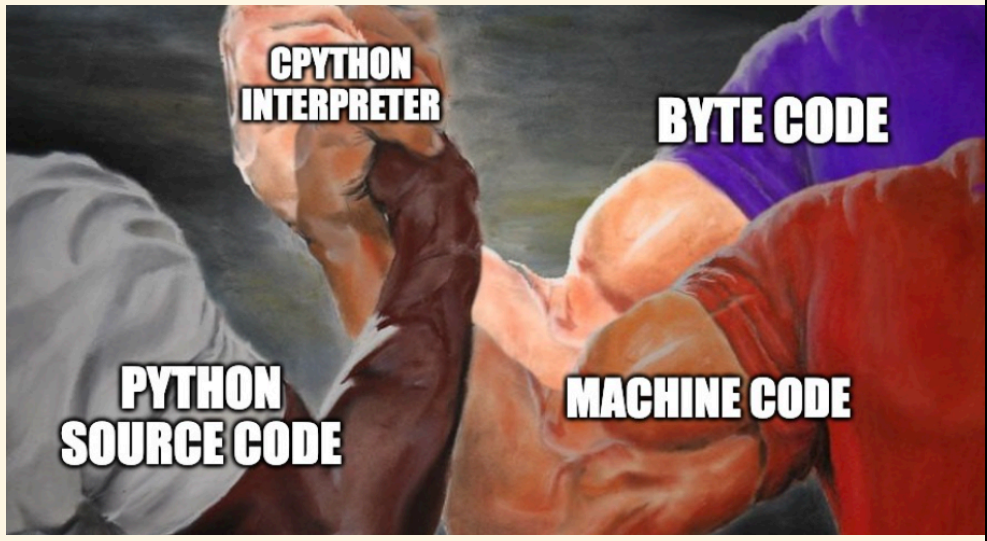


# CPython Interpreter

- What actually runs Python code
- **Two-step process:**
  1. Python source → **bytecode**
  2. Bytecode → **machine code**
- Looks like:

```
0 LOAD_GLOBAL              0 (print)
2 LOAD_CONST               1 ('Hello, World!')
4 CALL_FUNCTION            1
6 RETURN_VALUE
```

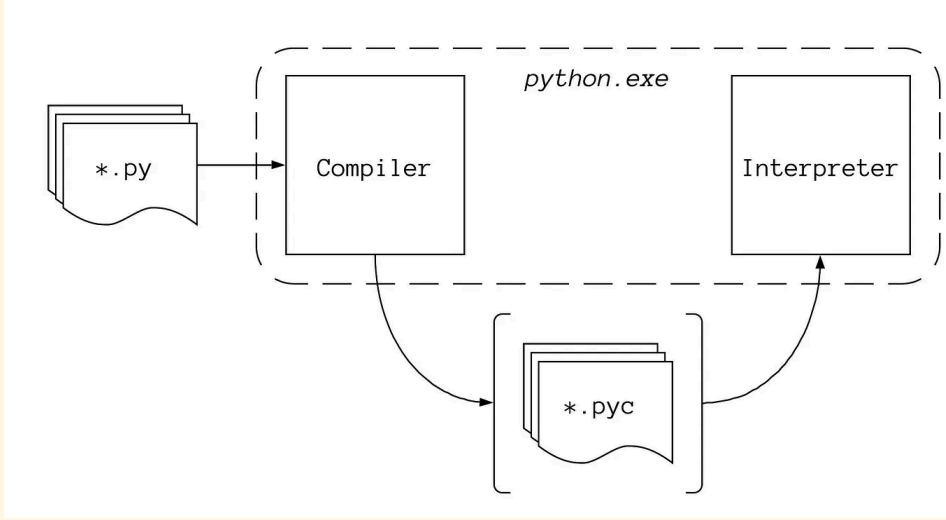
- Why two steps? **Portability** - same Python code runs everywhere.



# How Python Gets Executed

1. **You write** Python code in a `.py` file
2. **CPython interpreter** reads your code
3. **Converts** it to bytecode (behind the scenes)
4. **Computer executes** the bytecode
5. **You see** the result

*Python handles the translation for you!*



# Basic Input/Output

## Input

- **Getting information** from the user
- Keyboard, mouse, files, internet

## Output

- **Showing information** to the user
- Screen, speakers, files, internet

## In Python Code

- `input()` - get text from user
- `print()` - show text to user

## Output

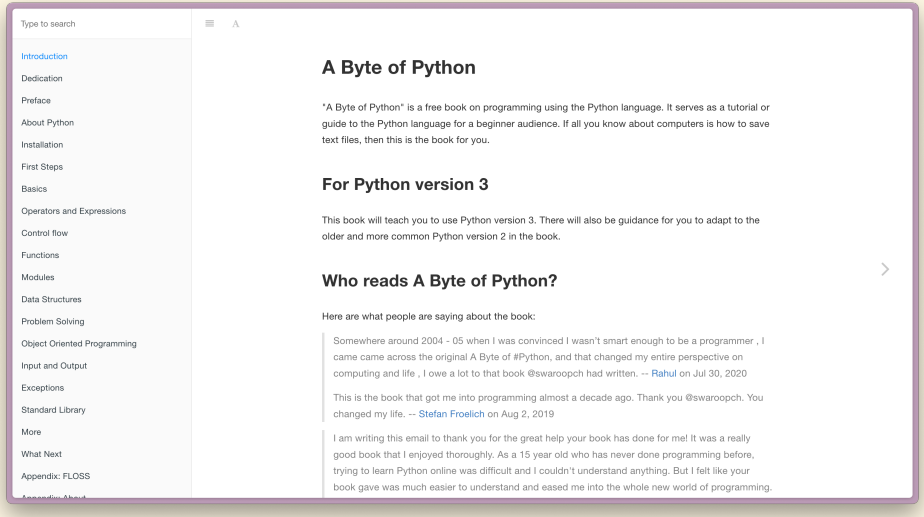
```
print("Hello, World!")
```

## Input and Output

```
name = input("Enter your name: ")
print(f"Hello, {name}!")
```

# A Byte of Python

- Written by **Swaroop Chitlur**
- All my slides for teaching Python are adapted from this book!
- Licensed under a **Creative Commons License**
- Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License
- https://python.swaroopch.com/



# Comments

- Use `#` for comments
- They can be above:

```
# Note that print is a function
print('hello world')
```

- Or at the end

```
print('hello world')  # Note that print is a function
```

**Well commented code is easier to understand for future readers (including yourself).**

# Literal Constants

- Numbers: `5`, `1.23`
- Strings: `'This is a string'`, `"It's a string!"`
- Boolean: `True`, `False`
- Nothing!: `None`

# Numbers

Numbers are mainly of two types - **integers** and **floats**.

## Integers

- An example of an integer is `2` which is just a whole number.

## Floats

- Examples of floating point numbers ("floats") are `3.23` and `52.3e-4`.
- The `e` notation indicates powers of 10. `52.3e-4` => `52.3 * 10^-4^`.

# Strings

- Strings are used to represent text.
- You can use single quotes, double quotes, or triple quotes for multi-line strings.

```
'Hello'
"Hello"
'''
This is a
multi-line string
'''
```

- Strings are **immutable**

An error occurred on this slide. Check the terminal for more information.

An error occurred on this slide. Check the terminal for more information.

An error occurred on this slide. Check the terminal for more information.

# Variables Rules

- The first character of the identifier must be a letter of the alphabet or an underscore (_).
- The rest of the identifier name can consist of letters (uppercase ASCII or lowercase ASCII or Unicode character), underscores (_) or digits (0-9).
- Identifier names are case-sensitive. For example, `myname` and `myName` are not the same.

## Examples

- `age`, `name`, `foo`, `bar`
- `age2`, `name_2`, `foo_bar`
- `_age`, `_name` - starting with `_` denotes "private"
- `AGE`, `NAME` - ALL_CAPS denotes "constant"
- `i`, `j`, `k` - common for iterators in loops (more on this later)
- `df`, `x`, `y`, `X`, `Y` - bad variables but common in data science

An error occurred on this slide. Check the terminal for more information.

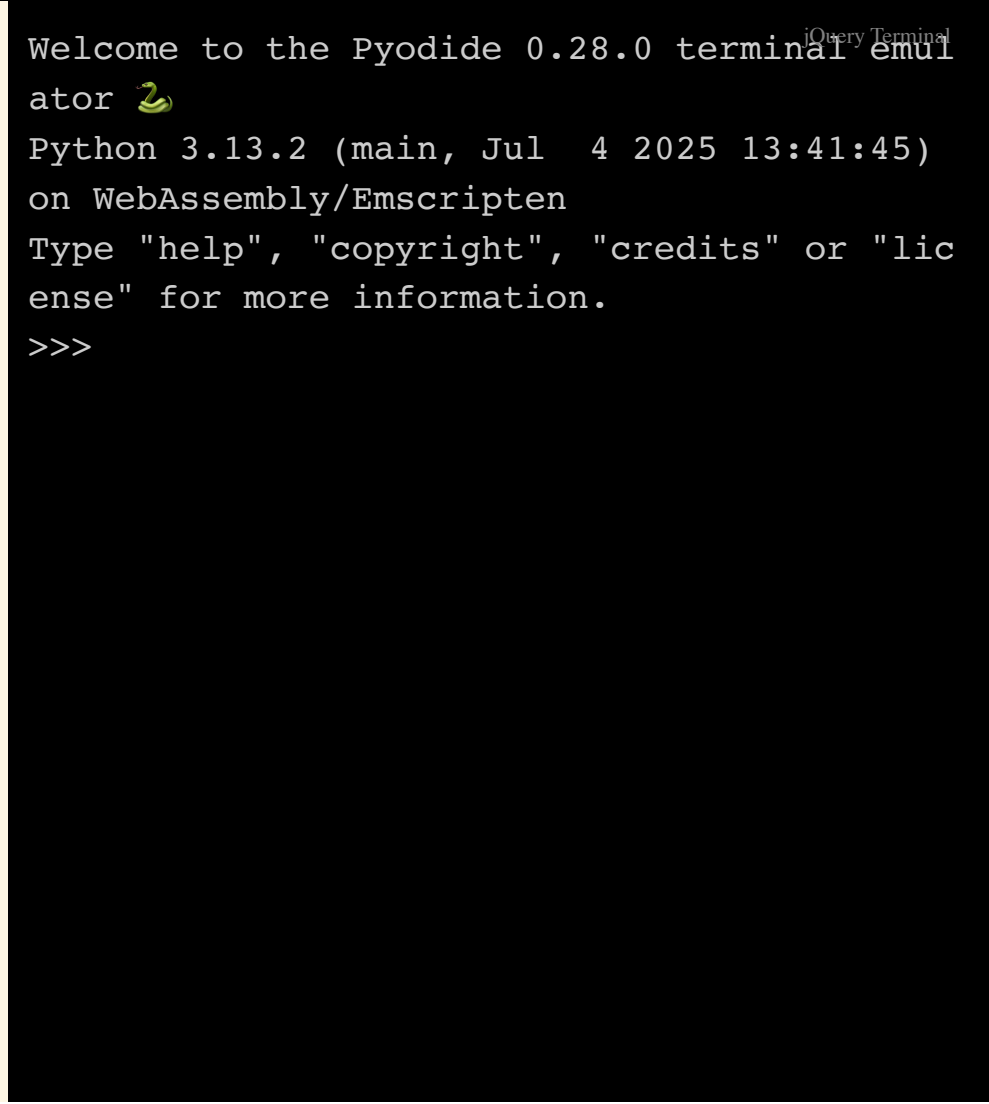
# Logical Operators

```
True and False
## False

True or False
## True

not True
## False
```

An error occurred on this slide. Check the terminal for more information.

# REPL - Read-Eval-Print Loop

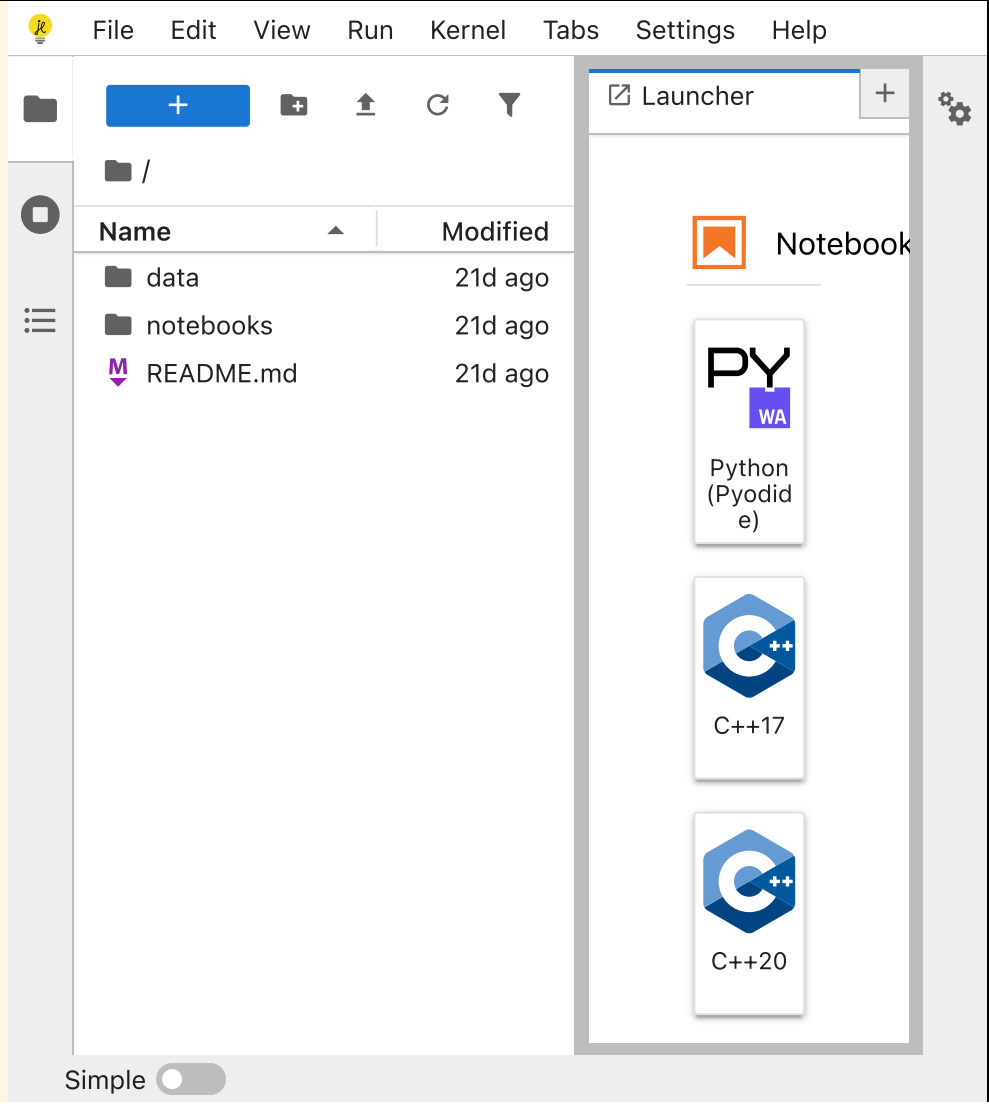
- **Interactive** way to run code
- **R**ead your code → **E**valuate it → **P**rint the result → **L**oop back
- Type code, hit Enter, see result immediately
- Perfect for **experimenting** and **learning**



# Jupyter Notebook

- **Web-based** coding environment
- Combines **code**, **text**, and **output** in one document
- Code goes in **cells** - run one cell at a time
- Like a **digital notebook** for programming
- Perfect for **data science** and **learning**

*Think: Google Docs + Python code*



# Exercise: Variables

bigd103.link/variables