Preface

Type	Lecture
Reviewed	☐

What is Python:

Python is a high-level, interpreted programming language known for its **simplicity**, **readability**, and **versatility**. It was created by **Guido van Rossum** and first released in **1991**.

Key Features:

- **Easy to Read & Write**: Python has a clean, English-like syntax that makes it beginner-friendly (pseudo-code nature).
- **Interpreted Language**: You don't need to compile your code before running it.
- **Dynamically Typed**: You don't have to declare the type of variables explicitly.
- **Extensive Libraries**: Python has powerful libraries for web development, data science, machine learning, automation, and more.
- **Cross-Platform**: Works on Windows, macOS, Linux, and more.

Used for:

- **Web Development** – with frameworks like Flask and Django
- **Data Science & Machine Learning –** using pandas, NumPy, scikit-learn, TensorFlow
- **Scripting & Automation –** automate files, emails, browsers, and more
- **Game Development** – with libraries like Pygame
- **Desktop GUI Apps** – using Tkinter, PyQt, etc.

Why Learn Python?

- It's one of the most in-demand programming languages today.

- It's great for beginners but powerful enough for professionals.
- It's supported by a massive community and tons of learning resources.

How to Install:

- Installed from python.org.
- Click on the download button.
- Can be installed right after you complete the setup by executing the file for your platform.

What "Interpreted Language" Means:

When we say **Python is an interpreted language**, it means:

> 🧠 You write the Python code in a .py file and run it directly using a program called the Python interpreter, without needing to compile it into machine code first.

In contrast:

- A **compiled language** (like C or Java) requires you to:
  1. Write the code.
  2. Use a **compiler** to translate it into machine code.
  3. Then run the compiled file (like `.exe`).
- An **interpreted language** (like Python):
  1. You write the code.
  2. The **interpreter runs the code line by line**, instantly showing the result or error.

Example:

Let's say you write this in Python:

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

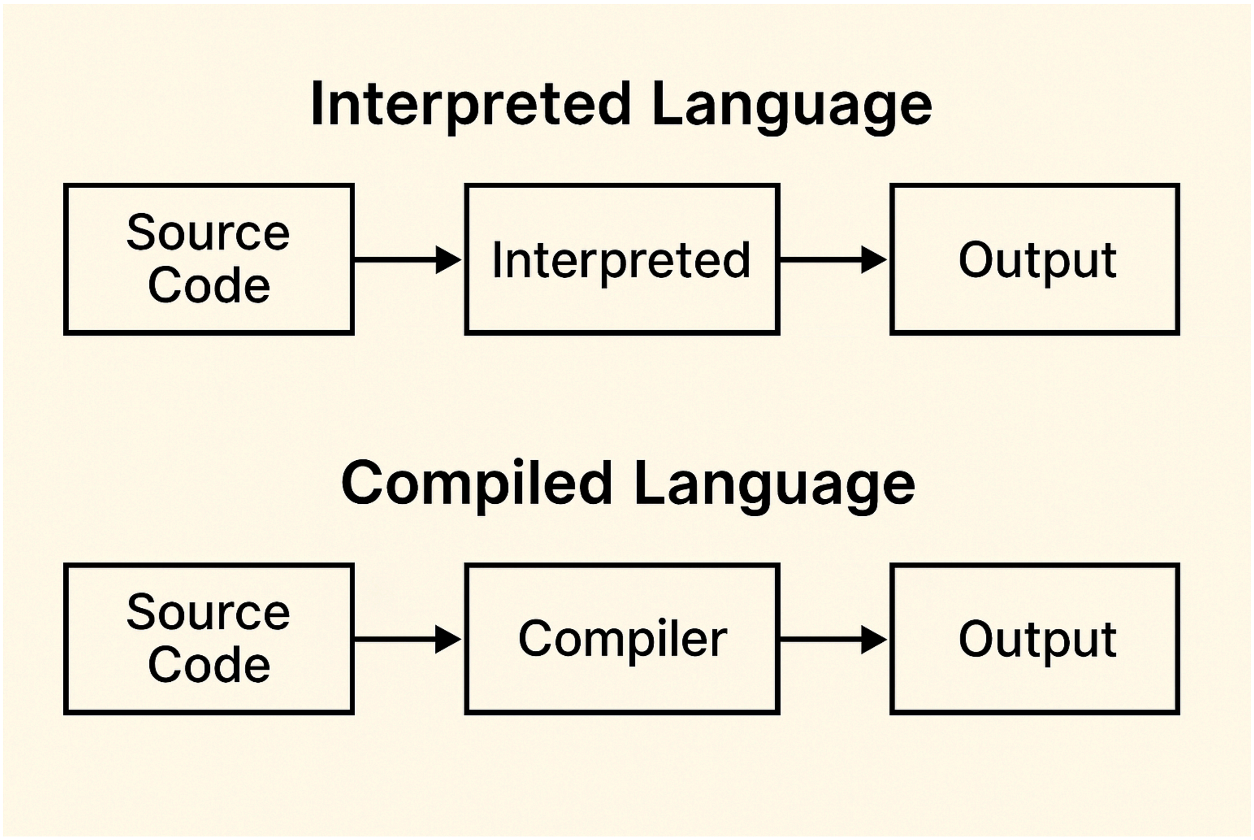
You can run it directly:

```
python hello.py
```

The Python interpreter reads and executes it **line-by-line** right away.

Advantage:

- Fast to test and run code.
- Perfect for beginners.
- Easy to debug since you can run and test small parts quickly.



What "Interpreted Language" Means:

It means:

In Python, **you don't need to say what type a variable is (like `int`, `str`, `float`, etc.) when you create it** — Python figures it out on its own based on the value you assign.

Example:

```
x = 5         # Python knows x is an integer
x = "Hello"    # Now x is a string
```

You never had to write `int x = 5` or `str x = "Hello"` like in some other languages (e.g., Java, C++).

Benefits:

- Faster to write code.
- Easier for beginners.

Downsides:

- You can accidentally change types and create bugs (e.g., mixing strings and numbers).

Here's a
simple visual and code snippet to help you clearly understand how Python handles dynamic typing:

Visual: Python Figures It Out

```
x = 42      → Python sees a number → x becomes an integer
x = "Hi"    → Python sees a string → x becomes a string
x = 3.14    → Python sees a float  → x becomes a float
```

Python dynamically updates the

type of `x` based on the assigned **value** — no need to tell it.

Code Snippet: See It in Action

```
x = 10
print(x, type(x))   # Output: 10 <class 'int'>
x = "Python"
print(x, type(x))   # Output: Python <class 'str'>
x = 3.14
print(x, type(x))   # Output: 3.14 <class 'float'>
```

Each time you assign a new value, Python
automatically adjusts the variable's type behind the scenes.