Python Programming

Unit 01 – Lecture 01: Introduction and Working with Python

Tofik Ali

School of Computer Science, UPES Dehradun

February 14, 2026

Repository: `https://github.com/tali7c/Python-Programming`

Quick Links

Core Concepts Demo Interactive Summary

Agenda

Learning Outcomes

- Explain **interactive (REPL)** vs **scripting** mode

Learning Outcomes

- Explain **interactive (REPL)** vs **scripting** mode
- Run Python from terminal/IDLE/IDE

Learning Outcomes

- Explain **interactive (REPL)** vs **scripting** mode
- Run Python from terminal/IDLE/IDE
- Write a first program using `print()` and `input()`

Learning Outcomes

- Explain **interactive (REPL)** vs **scripting** mode
- Run Python from terminal/IDLE/IDE
- Write a first program using `print()` and `input()`
- Follow the basic Input $\rightarrow$ Process $\rightarrow$ Output pattern

Why Python?

- Simple, readable syntax (good for beginners)

Why Python?

- Simple, readable syntax (good for beginners)
- Huge ecosystem (web, data, AI, automation)

Why Python?

- Simple, readable syntax (good for beginners)
- Huge ecosystem (web, data, AI, automation)
- Cross-platform (Windows, Linux, macOS)

Why Python?

- Simple, readable syntax (good for beginners)
- Huge ecosystem (web, data, AI, automation)
- Cross-platform (Windows, Linux, macOS)
- Great for rapid prototyping and scripting

Two Ways to Run Python

- **Interactive mode (REPL)**: run one line at a time

```
>>> 2 + 3
5
```

Two Ways to Run Python

- **Interactive mode (REPL)**: run one line at a time
- **Scripting mode**: save code in a `.py` file and run it

```
>>> 2 + 3
5
```

Interactive Mode (REPL)

Best for:

- quick calculations and testing small ideas

```
>>> name = "UPES"
>>> name.lower()
'upes'
```

Interactive Mode (REPL)

Best for:

- quick calculations and testing small ideas
- learning syntax (instant feedback)

```
>>> name = "UPES"
>>> name.lower()
'upes'
```

Interactive Mode (REPL)

Best for:

- quick calculations and testing small ideas
- learning syntax (instant feedback)
- checking library functions

```
>>> name = "UPES"
>>> name.lower()
'upes'
```

Scripting Mode (.py files)

Best for:

- programs you want to **save, reuse, and share**

Typical run command:

```
python my_program.py
```

Scripting Mode (.py files)

Best for:

- programs you want to **save, reuse, and share**
- assignments, labs, mini-projects

Typical run command:

```
python my_program.py
```

First Program: Input → Process → Output

```
name = input("Enter your name: ")
print("Hello,", name)
```

- `input()` always returns a `str`

First Program: Input → Process → Output

```
name = input("Enter your name: ")
print("Hello,", name)
```

- `input()` always returns a `str`
- Use type conversion when you need numbers

Type Conversion (Quick Example)

```
a = input("Enter a number: ")
b = input("Enter another number: ")
total = int(a) + int(b)
print("Sum =", total)
```

- Without `int()`, `"2" + "3"` becomes `"23"`

Demo: Mini Calculator

- File: `demo/mini_calculator.py`

Demo: Mini Calculator

- File: `demo/mini_calculator.py`
- Reads two numbers and prints +, -, *, /

Demo: Mini Calculator

- File: `demo/mini_calculator.py`
- Reads two numbers and prints +, -, *, /
- Shows input conversion and division-by-zero handling

Demo Snippet

```
a = float(input("Enter a: "))
b = float(input("Enter b: "))
print("a + b =", a + b)
print("a - b =", a - b)
print("a * b =", a * b)
if b != 0:
    print("a / b =", a / b)
else:
    print("Division by zero is not allowed.")
```

Checkpoint 1

Question: When is interactive mode (REPL) better than a script?

Your answer: write 2 use-cases.

Checkpoint 2

Question: What is the type of `input()`?

```
x = input("Enter something: ")
print(type(x))
```

Think-Pair-Share

In pairs, discuss:

- Where can Python help you in daily life or engineering?
- Give 3 examples (automation / data / web / etc.).

Key Takeaways

- REPL is for quick exploration; scripts are for reusable programs

Key Takeaways

- REPL is for quick exploration; scripts are for reusable programs
- `input()` returns a string; convert to `int/float` when needed

Key Takeaways

- REPL is for quick exploration; scripts are for reusable programs
- `input()` returns a string; convert to `int/float` when needed
- Most programs follow Input $\rightarrow$ Process $\rightarrow$ Output

Exit Question

Write the command to run a Python script named `task.py` from terminal.