Python Programming
Unit 01 – Lecture 01 Notes
Introduction and Working with Python

Tofik Ali

February 14, 2026

Contents

1 Lecture Overview

This lecture helps you **start** with Python the right way:

- what Python is and why it is popular,
- how to run Python in **interactive mode (REPL)** and **scripting mode**,
- and how to write a first program that follows the common pattern: **Input → Process → Output**.

Repository (course materials): `https://github.com/tali7c/Python-Programming`

2 Core Concepts

2.1 What is Python? (Quick Picture)

Python is a general-purpose programming language known for:

- **readable syntax** (you can focus on logic, not symbols),
- a **large ecosystem** of libraries (web, data analysis, AI, automation),
- and **cross-platform** support.

In this course, Python is used to build programming fundamentals that later support subjects like data structures, databases, and data analysis.

2.2 Two Ways to Run Python

There are two everyday ways to run Python code:

1. **Interactive mode (REPL):** You type a statement and get the result immediately. Good for exploration.
2. **Scripting mode:** You save a program in a `.py` file and execute it. Good for assignments, labs, and reusable programs.

2.2.1 Interactive Mode (REPL)

Example (Python prompt):

```
>>> 2 + 3
5
>>> "UPES".lower()
'upes'
```

Use REPL when you want quick feedback, or when you are learning a new concept.

2.2.2 Scripting Mode (`.py` file)

A script is a plain text file that contains Python statements. Typical run command in a terminal:

```
python my_program.py
```

If your file is `task.py`, the command is `python task.py`.

2.3 The IPO Pattern: Input → Process → Output

Most beginner programs follow this pattern:

- **Input:** read values from the user (keyboard).
- **Process:** compute something using variables and operators.
- **Output:** show the result using `print()`.

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

Important: `input()` always returns a `str`. So if you read numbers, you must convert:

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

If you forget `int(...)` here, then `"2" + "3"` becomes `"23"`, because string concatenation is used.

3 Demo Walkthrough: Mini Calculator

File: `demo/mini_calculator.py`

Goal

Read two numbers and print their sum, difference, product, and quotient. Also handle division by zero safely.

How to run

From the lecture folder:

```
python demo/mini_calculator.py
```

Key idea

Use `float(...)` to allow decimal inputs. Use an `if` condition to avoid dividing by zero.

4 Interactive Checkpoints (with Solutions)

Checkpoint 1 Solution

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

- When you want to quickly test a small expression (e.g., check operator precedence).
- When you want to explore a library function without writing a full program.

Checkpoint 2 Solution

Question: What is the type of `input()`?
Answer: `input()` returns a `str`. You can verify using:

```
x = input("Enter something: ")
print(type(x)) # <class 'str'>
```

5 Practice Exercises (with Solutions)

Try these after the lecture. Solutions are provided so you can self-check.

Exercise 1: Sum of Two Integers

Task: Read two integers and print their sum.

Solution:

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

Exercise 2: Area of a Circle

Task: Read radius r and print area using πr^2 (use $\pi = 3.14159$).

Solution:

```
pi = 3.14159
r = float(input("Enter radius: "))
area = pi * r * r
print("Area =", area)
```

Exercise 3: Print a Clean Output Line

Task: Print `Name: <name>, Age: <age>` in one line.

Solution:

```
name = input("Name: ")
age = int(input("Age: "))
print("Name:", name + ",", "Age:", age)
```

6 Exit Question (with Solution)

Question: Write the command to run `task.py` from terminal.

Solution: `python task.py`