Python Programming Lab — Lab Sheet 1 (Foundations)

Introduction, Comments, Variables, Data Types, Arithmetic, and Output

Department of Mechatronics and Robotics Engineering
Instructor: Eng. Elsayed Atif Aner — Semester: Fall 2025

Learning Outcomes

By the end of this lab, you will be able to:

- Explain what Python is and how to run code (REPL vs. script).
- Write readable code using comments and good naming.
- Create variables and understand dynamic typing.
- Use core built-in data types: `int`, `float`, `bool`, `str`, and `None`.
- Perform arithmetic with operators and understand precedence.
- Produce clear output with `print()`, `sep`, `end`, and f-strings.

1 Introduction

Python is a high-level, interpreted, general-purpose language used widely in data science, AI, robotics, web, and automation.

How Python Runs

- **Interactive (REPL):** open a terminal and run `python` or `python3`. Type code and press Enter.
- **Script:** create a file `lab1.py`, then run `python lab1.py`.

Indentation and Style

- Python uses **indentation** to define code blocks (no braces). Use 4 spaces per level.
- Follow **PEP 8** style: `snake_case` for variables, clear names, one statement per line.

First Program

```
1         print("Hello, Python!")    # our first output
```

Explanation:

- `print(...)` is a built-in function that sends text to the console.
- Text is inside quotes (`"..."`) making a `str` (string).

Expected output:

```
Hello, Python!
```

TRY IT

Create a file `hello.py` with the line above and run it using `python hello.py`.

2 Comments

Comments are ignored by Python but essential for readability and documentation.

Single-line and Inline Comments

```
# This is a single-line comment
print("Hi")     # This is an inline comment after code
```

Docstrings vs. Multi-line Comments

```
"""
This triple-quoted string is often used as a docstring.
It documents a module, function, or class.
"""
def greet():
"""Return a friendly greeting (function docstring)."""
return "Hello!"
```

Explanation:

- **Comments** start with `#` and are ignored.
- **Docstrings** (triple quotes) attach documentation to objects and can be viewed via `help()`.

TRY IT:

```
print(greet.__doc__)   # prints the function's docstring
help(greet)            # shows a help page in the console
```

Good Comment Practices

- Explain *why* (intent), not just *what*.
- Keep comments updated; outdated comments are harmful.
- Consider tags like `TODO:`, `FIXME:` for future work.

3 Variables

A **variable** is a name bound to a value in memory.

Assignment and Dynamic Typing

```
course = "Python"      # str
year = 2025            # int
pi = 3.14159           # float

course = 123           # now the same name refers to an int (dynamic typing)
```

Explanation:

- Python infers types at runtime (no type declarations).
- You can rebind a name to a different type (use with care).

Naming Rules & Conventions

- Start with a letter or _; then letters, digits, _.
- Case sensitive: `Score` and `score` are different.
- Avoid Python keywords: `for`, `if`, `class`, `return`, ...
- Use `snake_case` for variables (e.g., `total_marks`).

Multiple Assignment, Swapping, and Augmented Assignment

```
x, y = 10, 20       # multiple assignment
x, y = y, x         # swap without temp variable
x += 5              # augmented assignment (x = x + 5)
```

Inspecting Values

```
name = "Elsayed"
print(type(name))      # <class 'str'>
print(id(name))        # memory identity (implementation detail)
```

TRY IT

Define `a=3`, `b=4`. Swap them without a temporary variable and print the result.

4 Data Types

Core built-in scalar types you will use constantly:

Integers (`int`)

```
n = 42
big = 10_000_000_000       # underscores improve readability
neg = -5
```

Note: Python integers have arbitrary precision (no overflow like C).

Floating-point (`float`)

```
x = 3.14
y = 1.0e-3      # scientific notation = 0.001
```

Precision: Floats are binary approximations; beware rounding.

```
print(0.1 + 0.2)      # 0.30000000000000004
print(round(2/3, 3)) # 0.667
```

Booleans (`bool`)

```
is_ready = True
is_late = False
print(int(True), int(False))  # 1 0
```

Strings (`str`)

```
s = "robotics"
print(len(s))          # length
print(s[0], s[-1])     # indexing: first, last
print(s[0:3])          # slicing: 'rob'
print("Hello " + s)    # concatenation
print("ha" * 3)        # repetition: 'hahaha'
```

Escapes and Raw Strings

```
print("Line1\nLine2\tTabbed")
path = r"C:\Users\Elsayed\Desktop"  # raw string (no escapes)
```

None (absence of value)

```
result = None
print(result is None)   # True
```

Type Checking and Conversion

```
val = "123"
print(type(val))               # str
num = int(val)                 # convert to int
print(isinstance(num, int))    # True
print(float("2.5"))            # 2.5
print(str(3.14))               # "3.14"
print(bool(""))                # False  (empty string is False)
```

Extra: Math Helpers

```
import math
print(math.pi, math.sqrt(16), abs(-7), pow(2, 3))  # 3.1415...
   4.0 7 8
```

TRY IT

Write a program that asks for a `float` radius and prints area using `math.pi`.

5 Arithmetic Operators

The most common operators:

`+`	addition
`-`	subtraction
`*`	multiplication
`/`	true division (returns float)
`//`	floor division (truncates toward $-\infty$)
`%`	modulo (remainder)
`**`	exponent (power)

Examples and Precedence

```
a, b = 7, 3
print(a + b, a - b, a * b, a / b)   # 10 4 21 2.333...
print(a // b, a % b)                # 2 1
print(a ** b)                       # 343
print(2 + 3 * 4)                    # 14 (multiplication first)
print((2 + 3) * 4)                  # 20 (parentheses first)
```

Useful Built-ins

```
print(divmod(7, 3))   # (2, 1) -> quotient, remainder
print(round(3.14159, 2))  # 3.14
```

Mixing Types

```
print(1 + 2.0)    # 3.0 (int + float -> float)
# Beware with strings:
print("3" + "4")  # "34" (concatenation), not 7
# print("3" + 4)  # TypeError: must be str, not int
```

TRY IT

Compute the BMI given `weight (kg)` and `height (m)`:

$$\text{BMI} = \frac{\text{weight}}{(\text{height})^2}$$

Format the output to 2 decimal places (hint: f-strings).

6 Output (Printing)

`print()` sends human-readable text to the console.

Basic Printing

```
name, age = "Ali", 21
print("Name:", name, "Age:", age)    # multiple args
```

`sep` and `end` Parameters

```
print("A","B","C", sep="-")    # A-B-C
print("Hello", end=" ")        # no newline, space instead
print("World")
```

f-Strings (Formatted Strings)

```
pi = 3.14159265
r = 5
area = pi * r**2
print(f"Area = {area:.2f}")          # 2 decimal places
velocity, t = 12.5, 3
print(f"s = v*t = {velocity*t:.1f} m")
```

Explanation:

- Inside `f"..."` use {`expression:format`}.
- `:.2f` means fixed-point with 2 decimals.

`str()` vs. `repr()`

```
x = "robotics"
print(str(x))    # human-friendly
print(repr(x))   # developer representation -> "'robotics'"
    including quotes
```

Printing to a File (optional)

```
with open("out.txt", "w", encoding="utf-8") as f:
print("Hello file!", file=f)
```

TRY IT

Ask the user for their name and a score (float). Print:
`Student <name> scored <score> / 100.00` (format to 2 decimals).

Mini Worked Examples

Example 1 — Rectangle Area (I/O + arithmetic + formatting)

```
width  = float(input("Width (m): "))
height = float(input("Height (m): "))
area = width * height
print(f"Area = {area:.3f} m^2")
```

Example 2 — Celsius to Fahrenheit (formula + cast)

```
c = float(input("Celsius: "))
f = (c * 9/5) + 32
print(f"{c:.1f} C = {f:.1f} F")
```

Example 3 — Simple Physics (using math)

```
import math
v0 = float(input("Initial speed (m/s): "))
t  = float(input("Time (s): "))
g  = 9.81
s  = v0*t - 0.5*g*t**2
print(f"Displacement after {t:.1f}s: {s:.2f} m")
```

Practice Tasks (Submit)

1. **Profile Card:** store your full name, department, year, and a boolean `is_enrolled=True`. Print a single line with all fields.
2. **Circle:** input radius (float). Print circumference and area using `math.pi`, formatted to 2 decimals.
3. **Converter:** input a distance in meters. Print it in centimeters and kilometers.
4. **Mini Calculator:** input two numbers; print sum, difference, product, true division, floor division, remainder, and power.
5. **BMI+:** input weight (kg) and height (m). Print BMI to 2 decimals and a clean message.

Submission

Submit one Python file named `lab1_solutions.py` containing your solutions to all tasks, and include screenshots showing successful runs.

Common Pitfalls

- Forgetting to convert `input()` (which returns `str`) to `int`/`float`.
- Mixing strings and numbers without conversion (e.g., `"3" + 4` is an error).
- Relying on exact float equality (use rounding or tolerances).
- Missing parentheses in formulas; not using `math.pi`.

Instructor Notes (Remove or hide before sharing)

Sample Outputs / Hints

- Rectangle: width=3, height=2.5 ⇒ area=7.500 m^2.
- Circle: radius=2 ⇒ circumference ≈ 12.57, area ≈ 12.57.
- Converter: 1234 m ⇒ 123,400 cm; 1.234 km.
- Mini Calc (a=7, b=3): sum 10, diff 4, prod 21, / 2.3333..., // 2, % 1, ** 343.
- BMI: 70 kg, 1.75 m ⇒ 22.86 (approx).

Reference Solutions (sketches)

```
# 1 Profile Card
name = "Student Name"; dept = "Mechatronics"; year = 1; is_enrolled = True
print(f"{name} | {dept} | Year {year} | Enrolled={is_enrolled}")

# 2 Circle
import math
r = float(input("r: "))
C = 2*math.pi*r
A = math.pi*r**2
print(f"Circumference={C:.2f}, Area={A:.2f}")

# 3 Converter
m = float(input("Meters: "))
print(f"cm={m*100:.1f}, km={m/1000:.3f}")

# 4 Mini Calculator
a = float(input("a: ")); b = float(input("b: "))
print("sum", a+b, "diff", a-b, "prod", a*b)
print("div", a/b, "floordiv", a//b, "mod", a%b, "pow", a**b)

# 5 BMI+
w = float(input("kg: ")); h = float(input("m: "))
bmi = w/(h*h)
print(f"BMI={bmi:.2f}")
```