

Python Programming

Unit 01 – Lecture 04 Notes

Input/Output, Escape Sequences, Operators, Precedence

Tofik Ali

February 14, 2026

Contents

1	Lecture Overview	1
2	Core Concepts	1
2.1	Input and Output	1
2.2	Escape Sequences	2
2.3	Operator Categories	2
2.4	Precedence and Parentheses	2
2.5	Membership vs Identity	3
3	Demo Walkthrough	3
4	Interactive Checkpoints (with Solutions)	3
5	Practice Exercises (with Solutions)	4
6	Exit Question (with Solution)	4

1 Lecture Overview

This lecture explains how to:

- take input correctly (`input()` returns a string),
- print clean output (escape sequences and `print` options),
- use operators (arithmetic, logical, bitwise, membership, identity),
- and avoid mistakes using precedence and parentheses.

2 Core Concepts

2.1 Input and Output

`input(prompt)` reads a line of text from the user and returns a string. So conversion is necessary when you expect numbers:

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

Useful print options:

- `sep`: separator between printed values
- `end`: what to print at the end (default is newline)

```
print("A", "B", "C", sep=" | ") # A | B | C
print("Hello", end=" ")
print("World") # Hello World
```

2.2 Escape Sequences

Escape sequences start with a backslash `\`:

- `\n`: new line
- `\t`: tab
- `\\`: backslash character
- `\"`: double quote inside a double-quoted string

```
print("Line1\nLine2")
print("A\tB\tC")
print("C:\\\\Users\\\\tofik.ali")
```

2.3 Operator Categories

Category	Examples
Arithmetic	<code>+</code> <code>-</code> <code>*</code> <code>/</code> <code>%</code> <code>//</code> <code>**</code>
Relational	<code><</code> <code><=</code> <code>></code> <code>>=</code> <code>==</code> <code>!=</code>
Logical	<code>and</code> <code>or</code> <code>not</code>
Bitwise	<code>&</code> <code> </code> <code>^</code> <code>~</code> <code><<</code> <code>>></code>
Assignment	<code>=</code> <code>+=</code> <code>-=</code> <code>*=</code> <code>/=</code>
Membership	<code>in</code> , <code>not in</code>
Identity	<code>is</code> , <code>is not</code>

2.4 Precedence and Parentheses

Precedence decides which operator is evaluated first.

```
print(2 + 3 * 4) # 14
print((2 + 3) * 4) # 20
```

Best practice: even if you know the precedence, use parentheses when it improves clarity.

2.5 Membership vs Identity

Membership checks if a value exists in a collection:

```
nums = [10, 20, 30]
print(20 in nums) # True
print(50 not in nums) # True
```

Identity checks whether two variables refer to the same object:

```
a = [1, 2]
b = a
c = [1, 2]
print(a is b) # True (same object)
print(a is c) # False (different object)
```

Important: `==` compares values; `is` compares identity.

3 Demo Walkthrough

File: `demo/operator_precedence_playground.py`

What this demo shows

- precedence examples (with and without parentheses),
- a bitwise truth table for `&`, `|`, and `^`,
- membership and identity checks.

4 Interactive Checkpoints (with Solutions)

Checkpoint 1 Solution

Question: Evaluate: `2 + 3 * 4` and `(2 + 3) * 4`.

Answer:

- `2 + 3 * 4 = 14` because `*` happens before `+`.
- `(2 + 3) * 4 = 20` because parentheses force addition first.

Checkpoint 2 Solution

Question: difference between `==` and `is`?

Answer:

- `==` compares values (equality).
- `is` compares identities (same object or not).

5 Practice Exercises (with Solutions)

Exercise 1: Safe Division

Task: Read two numbers and divide them. Print a message if divisor is 0.

Solution:

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

Exercise 2: Membership Check

Task: Check whether a number is in the sequence (10,20,56,78,89).

Solution:

```
seq = (10, 20, 56, 78, 89)
n = int(input("Enter n: "))
print(n in seq)
```

Exercise 3: Escape Sequences

Task: Print the following in two lines: `Hello` on first line and `World` on second line.

Solution:

```
print("Hello\nWorld")
```

6 Exit Question (with Solution)

Question: give one example each: membership operator, identity operator.

Answer (example):

- membership: `5 in [1,2,3,4,5]`
- identity: `a is b`