Python Programming
Unit 01 – Lecture 05 Notes
Decision Making and Looping Structures

Tofik Ali

February 14, 2026

Contents

1 Lecture Overview

Programs make decisions and repeat work. This lecture covers:

- decision statements: `if/elif/else`, nested `if`,
- `match-case` (Python 3.10+) for clean multi-way branching,
- looping structures: `for` and `while`,
- loop control: `break`, `continue`, `pass`, and loop `else`.

2 Core Concepts

2.1 Decision Making with `if/elif/else`

Basic pattern:

```
marks = int(input("Marks: "))
if marks >= 90:
    print("A+")
elif marks >= 75:
    print("A")
else:
    print("Needs improvement")
```

Tips:

- Conditions are evaluated top to bottom.
- Use `elif` to avoid deeply nested code.
- Always think about boundary conditions (e.g., what about 89, 90?).

2.2 Nested `if`

Nested `if` is valid, but can reduce readability if used too much.

```
age = int(input("Age: "))
if age >= 18:
    if age >= 60:
        print("Senior")
    else:
        print("Adult")
else:
    print("Minor")
```

2.3 `match-case` (Python 3.10+)

`match-case` is useful for menu options and multi-way choices.

```
option = input("Choose A/B/C: ").strip().upper()
match option:
    case "A":
        print("You selected A")
    case "B":
        print("You selected B")
    case "C":
        print("You selected C")
    case _:
        print("Invalid option")
```

2.4 Loops

2.4.1 The `for` loop (with `range`)

Use `for` when you know the count or you are iterating over items.

```
for i in range(1, 6):
    print(i)
```

`range(start, stop, step)` examples:

```
range(5) # 0..4
range(1, 6) # 1..5
range(10, 0, -2) # 10,8,6,4,2
```

2.4.2 The `while` loop

Use `while` when repetition depends on a condition.

```
count = 3
while count > 0:
    print(count)
    count -= 1
```

Warning: if the condition never becomes false, you get an infinite loop.

2.5 Loop Control Statements

- `break`: exits the loop immediately.
- `continue`: skips the rest of the current iteration.
- `pass`: does nothing (placeholder).
- Loop `else`: runs only if the loop completes without `break`.

```
for i in range(1, 6):
    if i == 3:
        break
else:
    print("Runs only if there was no break")
```

3 Demo Walkthrough: Menu-Driven Program

File: `demo/menu_driven_number_analyzer.py`

Idea

This demo uses a loop to keep showing a menu until the user chooses to exit. It uses `match-case` to handle choices cleanly.

How to run

```
python demo/menu_driven_number_analyzer.py
```

4 Interactive Checkpoints (with Solutions)

Checkpoint 1 Solution

Question: When does the `else` part of a loop execute?

Answer: The loop `else` executes only when the loop finishes normally (no `break`). If a `break` happens, `else` is skipped.

Checkpoint 2 Solution

Question: difference between `break` and `continue`?

Answer:

- `break` ends the loop completely.
- `continue` skips the current iteration and moves to the next one.

5 Practice Exercises (with Solutions)

Exercise 1: Even or Odd

Task: Input a number and print whether it is even or odd.

Solution:

```
n = int(input("Enter n: "))
if n % 2 == 0:
    print("Even")
else:
    print("Odd")
```

Exercise 2: Sum 1..n

Task: Input `n` and compute sum of first `n` natural numbers using a loop.

Solution:

```
n = int(input("Enter n: "))
total = 0
for i in range(1, n + 1):
    total += i
print("Sum =", total)
```

Exercise 3: Skip Multiples of 3

Task: Print numbers 1..10 but skip multiples of 3.

Solution:

```
for i in range(1, 11):
    if i % 3 == 0:
        continue
    print(i)
```

6 Exit Question (with Solution)

Question: Write a loop that prints 1..10 but skips multiples of 3.

Solution: see Exercise 3 above.