Unit 3: Flow of Control

Python Programming (DI02032011)

Milav Dabgar

Lecturer, Dept of ICT
GP Palanpur

April 2, 2026

Unit 3 — Contents

What is Flow of Control?

Definition

Flow of Control refers to the order in which individual statements, instructions, or function calls are executed or evaluated in a program.

Types of Flow of Control:

- **Sequential** — Statements execute one after another (default)
- **Selection / Decision** — Execute different code based on a condition
- **Iteration / Loop** — Repeat a block of code multiple times
- **Jump** — Skip or break out of a block (`break`, `continue`)

Selection Statements — Overview

Selection / Decision Statements

Used to **choose** a path of execution based on a condition.

Statement	Description
`if`	Execute block only if condition is True
`if-else`	One block if True, another if False
`if-elif-else`	Multiple conditions / ladder
`nested if`	`if` inside another `if`

The if Statement

Syntax:

```
if condition:
    # block executed if condition is True
```

Example — check positive number:

```
n = int(input("Enter number: "))
if n > 0:
    print("Number is positive")
print("End of program")
```

Note: Indentation defines the block. If condition is False, the block is skipped.

The if-else Statement (Su'23 Q.3b, Su'25 Q.2b)

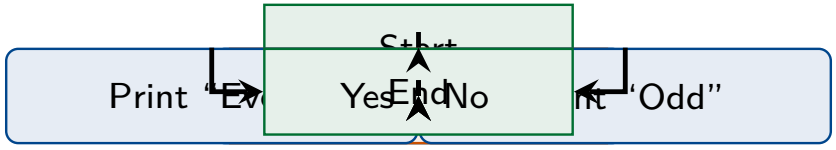
Syntax:

```
if condition:
    # block if True
else:
    # block if False
```

Example — even or odd:

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

Flowchart — `if-else`



Nested `if` Statement (W'23 Q.2b, Su'24 Q.3b)

Syntax:

```
if condition1:
    if condition2:
        # inner block
    else:
        # inner else
else:
    # outer else
```

Example — positive, negative or zero:

```
n = int(input("Enter number: "))
if n >= 0:
    if n == 0:
        print("Zero")
    else:
        print("Positive")
else:
```

The `if-elif-else` Ladder (W'24 Q.3a)

Syntax:

```
if condition1:
    # block 1
elif condition2:
    # block 2
elif condition3:
    # block 3
else:
    # default block
```

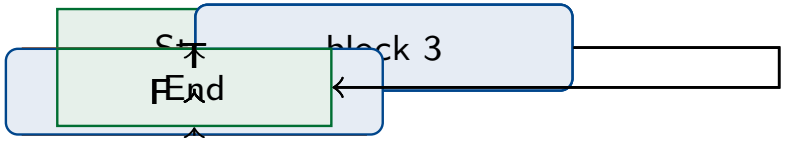
- Conditions are checked **top to bottom**
- First True condition executes its block
- `else` is optional — runs if none match
- Used when there are **multiple exclusive** conditions

if-elif-else — Grade Example (Su'24 Q.2c OR)

Grade Calculator

```
marks = int(input("Enter marks: "))
if marks >= 70:
    print("Grade: Distinction")
elif marks >= 60:
    print("Grade: First Class")
elif marks >= 50:
    print("Grade: Second Class")
elif marks >= 35:
    print("Grade: Pass Class")
else:
    print("Grade: Fail")
```

if-elif-else — Flowchart



Selection Statements — Comparison Table

Statement	When to use	Branches
`if`	Single optional action	1 (or skip)
`if-else`	Two exclusive paths	2
`if-elif-else`	Multiple exclusive conditions	3 or more
`nested if`	Conditions within conditions	Multiple levels

Loops — Overview

What is a Loop?

A loop **repeats** a block of code as long as a condition is True (or for each item in a sequence).

Loop	Description
`for`	Iterates over a sequence or range
`while`	Repeats while condition is True
`nested loop`	A loop inside another loop

Jump statements inside loops:

- `break` — exit the loop immediately
- `continue` — skip to next iteration

The for Loop (Su'23 Q.3b OR, W'23 Q.2b OR)

Syntax:

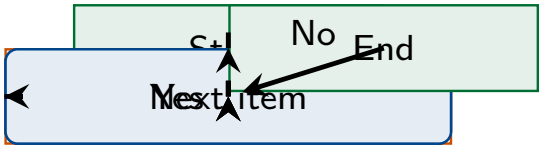
```
for variable in sequence:
    # loop body
```

Common uses with range():

```
# Print 1 to 10
for i in range(1, 11):
    print(i, end=" ")
# Output: 1 2 3 4 5 6 7 8 9 10

# range(start, stop, step)
for i in range(1, 20, 2):
    print(i, end=" ")  # odd numbers
```

`for` Loop — Flowchart



The `while` Loop (Su'24 Q.2c)

Syntax:

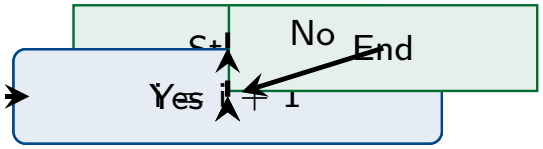
```
while condition:
    # loop body
    # update condition variable
```

Example — print 1 to 10:

```
i = 1
while i <= 10:
    print(i, end=" ")
    i = i + 1
# Output: 1 2 3 4 5 6 7 8 9 10
```

Warning: Forgetting to update the condition variable causes an **infinite loop**!

`while` Loop — Flowchart (Su'24 Q.2c)



`for` vs `while` — Comparison (W'25 Q.3b)

for loop	while loop
Fixed number of iterations	Indefinite iterations
Iterates over a sequence	Runs while condition True
Loop variable managed by loop	Programmer manages variable
Compact syntax with `range()`	More flexible for input loops
Best for lists, strings, ranges	Best for user-input validation

Nested Loops (W'24 Q.3a OR)

Definition

A loop inside another loop is called a **nested loop**. The inner loop runs **completely** for every iteration of the outer loop.

Example:

```
for i in range(1, 4):          # outer
    for j in range(1, 4):      # inner
        print(i * j, end="   ")
    print()
# Output:
# 1   2   3
# 2   4   6
# 3   6   9
```

Program: Numbers 1–10 & Odd/Even 1–N (W'23 Q.3a)

Print 1 to 10 using for loop

```
for i in range(1, 11):
    print(i, end=" ")
```

Odd and Even from 1 to N

```
n = int(input("Enter N: "))
for i in range(1, n + 1):
    if i % 2 == 0:
        print(i, "is Even")
    else:
        print(i, "is Odd")
```

Program: Sum 1 to 100 (Su'24 Q.3a OR)

Sum of numbers from 1 to 100

```
total = 0
for i in range(1, 101):
    total = total + i
print("Sum =", total)
# Sum = 5050
```

Using while loop

```
total = 0
i = 1
while i <= 100:
    total += i
    i += 1
print("Sum =", total)
```

`break` Statement (Su'23 Q.3a)

Definition

`break` **immediately exits** the loop, regardless of the loop condition.

Syntax:

```
for/while ...:
    if condition:
        break
```

Example — find first even number:

```
for i in range(1, 20):
    if i % 2 == 0:
        print("First even:", i)
        break
```

`continue` Statement (Su'23 Q.3a OR)

Definition

`continue` **skips** the rest of the current iteration and moves to the next one.

Example — skip even numbers:

```
for i in range(1, 11):
    if i % 2 == 0:
        continue        # skip even
    print(i, end=" ")
# Output: 1 3 5 7 9
```

Key difference:

- `break` — exits the loop entirely
- `continue` — skips current iteration, loop continues

Star Triangle Pattern (W'23 Q.3b)

Print star triangle (5 rows):

```
for i in range(1, 6):
    for j in range(1, i + 1):
        print("*", end=" ")
    print()
```

Output:

```
*
* *
* * *
* * * *
* * * * *
```

Inverted Star Triangle Pattern (Su'23 Q.2c OR)

Print inverted star triangle (5 rows):

```
for i in range(5, 0, -1):
    for j in range(1, i + 1):
        print("*", end=" ")
    print()
```

Output:

```
* * * * *
* * * *
* * *
* *
*
```

Number Triangle Pattern (Su'23 Q.2c, Su'24 Q.3b OR)

Problem: Print `1 / 1 2 / 1 2 3 / 1 2 3 4 / 1 2 3 4 5`

```
for i in range(1, 6):
    for j in range(1, i + 1):
        print(j, end=" ")
    print()
```

Output:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

& Character Triangle Pattern (Su'25 Q.1c)

Print & triangle (5 rows):

```
for i in range(1, 6):
    for j in range(1, i + 1):
        print("&", end=" ")
    print()
```

Inverted @ triangle (W'25 Q.2c):

```
for i in range(5, 0, -1):
    for j in range(1, i + 1):
        print("@", end=" ")
    print()
```

Alphabet Triangle Pattern (W'24 Q.3c OR)

Print A / AB / ABC / ABCD / ABCDE:

```
for i in range(1, 6):
    for j in range(i):
        print(chr(65 + j), end="")
    print()
```

Output:

```
A
AB
ABC
ABCD
ABCDE
```

Note: chr(65) = 'A', chr(66) = 'B', etc.

Number Series: Row Numbers (Su'23 Q.2c)

Print: 1 2 3 / 4 5 6 / 7 8 9 10 (triangular number series):

```
n = 1
for i in range(1, 5):
    for j in range(1, i + 1):
        print(n, end=" ")
        n += 1
    print()
```

Output:

```
1
2 3
4 5 6
7 8 9 10
```

Squared Number Series (W'25 Q.2c)

Print: 1 / 1 4 / 1 4 9 / 1 4 9 16:

```
for i in range(1, 5):
    for j in range(1, i + 1):
        print(j * j, end=" ")
    print()
```

Output:

```
1
1 4
1 4 9
1 4 9 16
```

Program: Prime Number Check (Su'23 Q.2c, W'25 Q.3b OR)

Check if number is prime

```
n = int(input("Enter number: "))
is_prime = True
if n < 2:
    is_prime = False
else:
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            is_prime = False
            break
if is_prime:
    print(n, "is Prime")
else:
    print(n, "is Not Prime")
```

Program: Odd/Even + Average 1–n (Su'25 Q.1c OR)

Check odd or even

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

Average of numbers 1 to n

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

Program: Print 1–25 with `for` Loop (W'25 Q.3b)

Print numbers 1 to 25 using for loop

```
for i in range(1, 26):
    print(i, end=" ")
print()
```

Print odd numbers 1 to 20 using while loop

```
i = 1
while i <= 20:
    if i % 2 != 0:
        print(i, end=" ")
    i += 1
```

PYQ Practice — 3-Mark Questions

3-Mark PYQ Bank

1. Describe the use of `break` statement with examples *(Su'23)*
2. Describe the use of `continue` statement with examples *(Su'23 OR, Su'24 OR)*
3. Develop Python code to identify if number is even or odd *(Su'23)*
4. Create Python code to check if number is positive or negative *(Su'23 OR)*
5. Explain if-elif-else statement with flowchart and example *(W'24)*
6. Develop Python program to print numbers 1 to 10 using loops *(W'23)*
7. Develop Python code to find odd and even numbers from 1 to N *(W'23 OR)*
8. Write Python program to print odd numbers between 1 to 20 *(Su'24)*
9. Write Python program to find sum of numbers from 1 to 100 *(Su'24 OR)*
10. Explain nested loop with a suitable example *(W'24 OR)*

PYQ Practice — 4-Mark Questions

4-Mark PYQ Bank

1. Explain if...else statement with suitable example *(Su'23, Su'25, W'25)*
2. Explain Nested if statement with Python code example *(W'23, Su'24, W'25 OR)*
3. Explain `for` loop statement with example *(Su'23 OR, W'23 OR, W'24 OR, Su'25)*
4. Explain the need for `continue` and `break` statements *(W'24)*
5. Explain `while` loop with example *(Su'25 OR)*
6. Explain `break` statement with example *(Su'25 OR)*
7. Differences between `for` and `while` loop; print 1–25 using `for` *(W'25)*
8. Develop Python program to find if the entered number is prime or not *(W'25 OR)*
9. Print star pattern (5 rows) *(W'23)*
10. Print number triangle: 1 2 3 / 4 5 6 / 7 8 9 10 *(Su'24 OR)*

PYQ Practice — 7-Mark Questions

7-Mark PYQ Bank

1. Write Python code to check whether entered number is prime *(Su'23)*
2. Display patterns: (A) number triangle (B) inverted star triangle *(Su'23 OR)*
3. Explain different types of selection/decision making structures *(W'23)*
4. Describe `break` and `continue` statements in Python *(W'23 OR)*
5. Explain `while` loop with syntax, flowchart and Python code *(Su'24)*
6. Explain if-elif-else ladder with syntax, flowchart and Python code *(Su'24 OR)*
7. Patterns: (A) `&` character triangle (B) number triangle 1/12/123 *(Su'25)*
8. Programs: (1) odd or even (2) average of 1 to n *(Su'25 OR)*
9. Patterns: (A) `@` inverted triangle (B) squared number series *(W'25)*
10. Explain `for` loop with syntax, flowchart and Python code *(W'25 OR)*
11. Pattern: A / AB / ABC / ABCD / ABCDE *(W'24 OR)*

Worked Solution: `break` & `continue` (W'23 Q.2c OR)

break statement

```
# break: exit loop when 5 is found
for i in range(1, 11):
    if i == 5:
        print("Found 5, breaking!")
        break
    print(i, end=" ")
# Output: 1 2 3 4 Found 5, breaking!
```

continue statement

```
# continue: skip multiples of 3
for i in range(1, 11):
    if i % 3 == 0:
        continue
    print(i, end=" ")
# Output: 1 2 4 5 7 8 10
```

Worked Solution: Selection Structures (W'23 Q.2c)

All three selection types in one program

```
# if statement
x = 10
if x > 0:
    print("Positive")

# if-else
if x % 2 == 0:
    print("Even")
else:
    print("Odd")

# if-elif-else
if x >= 90:
    print("A")
elif x >= 75:
    print("B")
elif x >= 50:
    print("C")
else:
    print("F")
```

Worked: `while` Loop — Full Solution (Su'24 Q.2c)

`while` loop syntax:

```
while condition:
    # body
    # update
```

Example — sum of digits:

```
n = int(input("Enter number: "))
total = 0
while n != 0:
    digit = n % 10
    total += digit
    n = n // 10
print("Sum of digits:", total)
```

Worked: `for` Loop — Full Solution (W'25 Q.2c OR)

Syntax and examples:

```
# Syntax
for variable in sequence:
    body

# Iterate over a list
fruits = ["apple", "banana", "cherry"]
for f in fruits:
    print(f)

# Iterate over range
for i in range(1, 6):
    print(i ** 2, end=" ")
# Output: 1 4 9 16 25

# Iterate over string
for ch in "PYTHON":
    print(ch, end="-")
```

The pass Statement (Su'25 OR, W'25 — 3M)

What is pass?

pass is a **null operation** — it does nothing. Used as a placeholder when a statement is syntactically required but no code is needed.

```
# Placeholder in if
x = 10
if x > 5:
    pass            # do nothing for now
else:
    print("small")

# Placeholder in loop
for i in range(5):
    pass            # empty loop body

# Placeholder in function
def future_function():
    pass            # to be implemented later
```

range() Function (Su'24, W'24 OR — 3M)

Syntax of range()

`range(stop)` | `range(start, stop)` | `range(start, stop, step)`

```
for i in range(5):          # 0 1 2 3 4
    print(i, end=" ")

for i in range(1, 6):       # 1 2 3 4 5
    print(i, end=" ")

for i in range(0, 11, 2): # 0 2 4 6 8 10
    print(i, end=" ")

for i in range(10, 0, -2):# 10 8 6 4 2
    print(i, end=" ")

# Countdown
for i in range(5, 0, -1): # 5 4 3 2 1
    print(i, end=" ")
```

`else` Clause with Loops (Su'25 OR, W'25 OR — 3M)

Loop-else Concept

The `else` block after a loop executes when the loop completes **normally** (not interrupted by `break`).

```
# for-else: search example
nums = [1, 3, 5, 7, 9]
target = 6
for n in nums:
    if n == target:
        print("Found!")
        break
else:
    print("Not found")    # prints if no break

# while-else
i = 0
while i < 3:
    print(i)
    i += 1
else:
    print("Loop done")    # runs when i==3
```

Program: Multiplication Table (W'23, Su'24 — 4M)

```python
# Multiplication table of N
n = int(input("Enter number: "))
print(f"Multiplication Table of {n}:")
for i in range(1, 11):
    print(f"{n} x {i:2d} = {n*i:3d}")

# Output for n=5:
# 5 x  1 =   5
# 5 x  2 =  10
# ...
# 5 x 10 =  50

# Nested: All tables from 1 to 5
for n in range(1, 6):
    for i in range(1, 11):
        print(f"{n}x{i}={n*i}", end="  ")
    print()
```

Program: Factorial & Sum of Digits (W'24, Su'25 — 4M)

```python
# Factorial using while loop
n = int(input("N: "))
fact, i = 1, 1
while i <= n:
    fact *= i
    i += 1
print(f"{n}! = {fact}")

# Sum of digits
num = int(input("Number: "))
total = 0
while num > 0:
    total += num % 10
    num //= 10
print("Sum of digits =", total)
```

Program: Fibonacci Series (W'25, Su'25 — 7M)

```python
# Fibonacci series up to N terms
n = int(input("How many terms? "))
a, b = 0, 1
count = 0
print("Fibonacci series:")
while count < n:
    print(a, end=" ")
    a, b = b, a + b
    count += 1

# Alternative: for loop version
a, b = 0, 1
for _ in range(n):
    print(a, end=" ")
    a, b = b, a + b
# Output (n=8): 0 1 1 2 3 5 8 13
```

Program: Reverse Number & Palindrome (Su'24 OR, W'23 — 4M)

```python
# Reverse a number
n = int(input("Number: "))
original = n
rev = 0
while n > 0:
    rev = rev * 10 + n % 10
    n //= 10
print("Reversed:", rev)
if original == rev:
    print(f"{original} is a Palindrome")
else:
    print(f"{original} is NOT a Palindrome")
```

Program: Sum of Series (W'25, Su'24 — 7M)

```python
# Series: 1 + 1/2 + 1/3 + ... + 1/N
n = int(input("N: "))
total = 0.0
for i in range(1, n+1):
    total += 1/i
print(f"Sum = {total:.4f}")

# Series: 1^2 + 2^2 + ... + N^2
total2 = 0
for i in range(1, n+1):
    total2 += i*i
print(f"Sum of squares = {total2}")

# Series: 1 + 3 + 5 + ... (N odd numbers)
total3, i, cnt = 0, 1, 0
while cnt < n:
    total3 += i; i += 2; cnt += 1
print(f"Sum of {n} odd numbers = {total3}")
```

Diamond Star Pattern (W'25 OR, Su'25 — 7M)

```
n = int(input("Rows: "))    # e.g. n=4
# Upper half (including middle)
for i in range(1, n+1):
    print(" "*(n-i) + "*"*(2*i-1))
# Lower half
for i in range(n-1, 0, -1):
    print(" "*(n-i) + "*"*(2*i-1))

# Output (n=4):
#     *
#    ***
#   *****
#  *******
#   *****
#    ***
#     *
```

Floyd's Triangle & Right Angle (Su'24 OR, W'25 — 4M)

```python
# Floyd's Triangle (consecutive numbers)
n = 5
num = 1
for i in range(1, n+1):
    for j in range(i):
        print(num, end=" ")
        num += 1
    print()
# 1
# 2 3
# 4 5 6
# 7 8 9 10
# 11 12 13 14 15

# Right-angle with row numbers
for i in range(1, n+1):
    for j in range(1, i+1):
        print(j, end=" ")
    print()
# 1  / 1 2  / 1 2 3 ...
```

Nested Loop: Multiplication Table Grid (W'25 — 7M)

```python
# Print 5x5 multiplication table grid
n = 5
print("    ", end="")
for j in range(1, n+1):
    print(f"{j:4d}", end="")
print()
print("-" * (4*n+5))
for i in range(1, n+1):
    print(f"{i:3d}|", end="")
    for j in range(1, n+1):
        print(f"{i*j:4d}", end="")
    print()
# Output:
#       1   2   3   4   5
# ----------------------
#   1|   1   2   3   4   5
#   2|   2   4   6   8  10
```

Program: Count Digits & Armstrong Number (Su'25, W'24 — 4M)

```python
# Count digits in a number
n = int(input("Number: "))
count = 0
temp = n
while temp != 0:
    temp //= 10
    count += 1
print(f"Digits: {count}")

# Armstrong Number (3-digit: sum of cubes)
n = int(input("Number: "))
digits = [int(d) for d in str(n)]
p = len(digits)
arm_sum = sum(d**p for d in digits)
if arm_sum == n:
    print(f"{n} is Armstrong")
else:
    print(f"{n} is NOT Armstrong")
```

Worked: break, continue, pass (W'24 OR — 7M)

```python
# Find first multiple of 7 between 1-50
for i in range(1, 51):
    if i % 7 == 0:
        print(f"First multiple of 7: {i}")
        break        # exit loop immediately

# Print only even numbers 1-10 (skip odd)
for i in range(1, 11):
    if i % 2 != 0:
        continue     # skip rest of this iteration
    print(i, end=" ")  # 2 4 6 8 10

# pass: future implementation placeholder
for i in range(5):
    if i == 3:
        pass         # do nothing special for 3
    else:
        print(i, end=" ")  # 0 1 2 4
```

Worked: while Loop with Validation (W'25, Su'25 — 7M)

```
# Grade entry until 'done'
total, count = 0, 0
while True:
    grade = input("Grade (or 'done'): ")
    if grade == 'done':
        break
    score = float(grade)
    total += score
    count += 1
if count > 0:
    print(f"Average: {total/count:.2f}")

# Count positive/negative/zero
n = int(input("How many numbers? "))
pos = neg = zer = 0
for _ in range(n):
    x = int(input())
    if x > 0: pos += 1
    elif x < 0: neg += 1
    else: zer += 1
print(f"Pos={pos}, Neg={neg}, Zero={zer}")
```

Unit 3 — Rapid Revision Sheet

Statement	Key Points
`if-elif-else`	Check multiple conditions in sequence
`for` loop	Fixed iterations using `range()` or sequence
`while` loop	Repeat while condition is True
`break`	Exit loop immediately
`continue`	Skip current iteration, go to next
`pass`	Null statement; placeholder
`range(s,e,st)`	Generate sequence from s to e-1 step st
loop `else`	Runs when loop completes without `break`
Nested loops	Use for 2D patterns, tables, matrix
Indentation	Mandatory in Python (4 spaces or 1 tab)

GTU Pattern (Unit 3 — 14%)

3-mark: definitions, comparisons 4-mark: programs 7-mark: patterns + programs

Common Mistakes in Loops (Exam Guide)

Mistakes

× Infinite while (missing update)
× Off-by-one (`range(1,n)` vs `range(1,n+1)`)
× Wrong indentation
× Using = instead of == in `if`
× Modifying loop variable inside
× Missing `break` in when needed

Correct Patterns

`while i < n: …; i += 1`
`range(1, n+1)` for 1 to n
Always 4 spaces indent
`if x == 5:` not `if x = 5:`
Update counter outside body
Use `break` to exit search

Worked: Pattern Analysis Method (7M Exam Guide)

How to Design Any Pattern Loop

Step 1: Draw the pattern and number each row (starting from 1).
Step 2: Identify what is printed per row (spaces + characters).
Step 3: Find the formula for spaces and characters in terms of row i.
Step 4: Outer loop: `for i in range(1, n+1)`
Step 5: Inner loop: use formula from Step 3.

```
# Right-angle triangle (n=4)
# Row 1: *              (1 star)
# Row 2: * *            (2 stars)
# Row 3: * * *          (3 stars)
# Row 4: * * * *        (4 stars)
# Formula: row i has i stars
n = 4
for i in range(1, n+1):
    print("* " * i)
```

Worked: GCD & LCM Using Loop (Su'25, W'24 — 4M)

```
# GCD using Euclidean algorithm
a = int(input("A: "))
b = int(input("B: "))
original_a, original_b = a, b
while b != 0:
    a, b = b, a % b
gcd = a
print(f"GCD of {original_a} and {original_b} = {gcd}")

# LCM using GCD
lcm = (original_a * original_b) // gcd
print(f"LCM = {lcm}")

# Output for A=12, B=8:
# GCD = 4
# LCM = 24
```

Worked: Comprehensive Loop Programs (W'25 — 7M)

```python
# Sum of N terms: 1 - 1/2 + 1/3 - 1/4 ...
n = int(input("N: "))
s, sign = 0, 1
for i in range(1, n+1):
    s += sign * (1/i)
    sign *= -1
print(f"Sum = {s:.4f}")

# Check perfect number
n = int(input("Number: "))
total = sum(i for i in range(1, n) if n % i == 0)
if total == n:
    print(f"{n} is Perfect")
else:
    print(f"{n} is NOT Perfect")

# Count odd numbers between A and B
a, b = int(input("A: ")), int(input("B: "))
count = sum(1 for i in range(a, b+1) if i % 2 != 0)
print(f"Odd count between {a} and {b}: {count}")
```

Unit 3 — Summary

Selection

if — single condition
if-else — two branches
if-elif-else — ladder
nested if — multi-level

Loops

for — fixed iterations
while — condition-based
nested — loop inside loop

Jump

break — exit loop
continue — skip iteration

Patterns

Star / Number / Character
Triangle / Inverted / Series
Use nested `for` loops