Module 3: Control Flow Statements

Control flow statements are used to control the flow of execution of the program. They fall into two main categories: **Conditional** (Decision Making) and **Looping** (Iteration).

3.1 Conditional Statements (Decision Making)

Conditional statements execute a block of code only if a specified condition is `True`.

The `if` statement

The simplest decision structure. The code block beneath `if` is executed only if the condition is true.

Real-Time Example: Checking for a positive bank balance.

Python
```
balance = 5000.00
withdrawal_amount = 6000.00

if balance >= withdrawal_amount:
    balance -= withdrawal_amount
    print(f"Withdrawal successful. New balance: ${balance:.2f}")

print("Transaction completed.")
# Since 6000 > 5000, the 'if' block is skipped.
```

The `if-else` statement

Provides two possible paths of execution. If the `if` condition is `True`, the `if` block executes; otherwise, the `else` block executes.

Real-Time Example: User authentication check (login).

Python
```
stored_password = "password123"
user_input = input("Enter password: ")

if user_input == stored_password:
    print("✅ Login successful. Welcome back!")
    is_authenticated = True
else:
    print("❌ Login failed. Incorrect password.")
    is_authenticated = False
```

The `if-elif-else` chain

Used when you have multiple conditions to check sequentially. The conditions are evaluated from top to bottom. As soon as a condition is `True`, its corresponding block is executed, and the entire chain is exited. The `else` block (optional) executes if none of the preceding conditions are true.

Real-Time Example: Grading system calculation.

Python
```
score = 85

if score >= 90:
    grade = 'A'
elif score >= 80: # This runs only if score < 90
    grade = 'B'
elif score >= 70: # This runs only if score < 80
    grade = 'C'
else:
    grade = 'F'

print(f"Student score: {score}, Grade: {grade}") # Output: Grade: B
```

Nested if statements

Placing one if or if-else structure inside another if or else block. This allows for complex, multi-level decision-making.

Real-Time Example: E-commerce shipping qualification.

Python
```
is_prime_member = True
order_total = 75.50

if is_prime_member:
    if order_total >= 50:
        shipping_cost = 0.00
        print("🎉 Prime member with order over $50: Free shipping!")
    else:
        shipping_cost = 5.00
        print("Prime member, but order under $50. Shipping cost: $5.00")
else:
    # Non-Prime Member logic
    shipping_cost = 10.00
    print("Not a Prime member. Shipping cost: $10.00")
```

3.2 Looping Statements (Iteration)

Looping statements allow a block of code to be executed repeatedly.

The while loop

Executes a block of code **as long as** its condition remains True. You must ensure the condition eventually becomes False to avoid an infinite loop.

Real-Time Example: Countdown timer.

Python
```
timer = 5
```