# Session 4.1: Conditional Statements - Making Decisions

Programs That Think • Week 4, Day 1

## Today's Journey

- Part 1: If/Elif/Else - Decision Making
- Part 2: Workshop - Build Smart Programs
- Break
- Part 3: Boolean Logic & Complex Conditions
- Part 4: Workshop - Advanced Logic
- Part 5: Quiz & Wrap-up

## Part 1: Conditional Statements

Programs that make decisions!

## Welcome to Week 4!

Last week you learned:

- ✓ For loops (automation)
- ✓ Nested loops (2D data)
- ✓ While loops (conditions)
- ✓ break and continue

This week: DECISION MAKING!

Conditionals = programs that think

- Different actions for different data
- Filter and validate
- Respond intelligently

This makes programs truly smart! 🚀

## What Are Conditionals? (Simple Explanation)

Conditional = if this, then that

Real life examples:

- If raining, bring umbrella
- If grade >= 90, give A
- If age < 18, deny access
- If password correct, log in

Programs need to make choices!

Different inputs → different outputs

Conditionals = decision-making in code

## If Statement - Basic Syntax

Format:

- if CONDITION: `# code runs if True`

Parts:

- if = keyword
- CONDITION = expression (True or False)
- : = colon (required!)
- Indented code = runs when True

If condition is False:

- Code inside is skipped

## Your First If Statement

```
# Check if passing grade
grade = 85
if grade >= 60:
    print('You passed!')
    print('Congratulations!')
print('Grade check complete')
# If grade is 85:
# You passed!
# Congratulations!
# Grade check complete
# If grade was 50:
# Grade check complete
# (skips the indented code!)
```

## Comparison Operators (Memorize These!)

Operators that compare values:

- == equal to (note: TWO equals!)
- != not equal to
- < less than
- > greater than
- <= less than or equal to
- >= greater than or equal to
- Return: True or False (booleans)

Examples:

- 5 == 5 → True
- 5 != 3 → True
- 10 > 20 → False

# Comparison Operators in Action

```python
# Examples with variables
age = 25
score = 95
name = 'Alice'

# Comparisons return True/False
print(age >= 18)       # True
print(score == 100)    # False
print(name == 'Bob')   # False

# Use in if statements
if score >= 90:
    print('A grade!')   # Runs!
if age < 21:
    print('Under 21')   # Doesn't run

# Common mistake: = vs ==
# = assigns, == compares!
```

## If-Else - Two Choices

Format:

- if CONDITION: `# runs if True`
- else: `# runs if False`

One or the other ALWAYS runs

- Never both, never neither

Use when:

  - Binary choice (yes/no)
  - Pass/fail
  - Valid/invalid

Either/or decisions!

## If-Else Examples

```
# Grade pass/fail
grade = 75
if grade >= 60:
    print('PASSED')
    status = 'pass'
else:
    print('FAILED')
    status = 'fail'
print(f'Status: {status}')

# One of these ALWAYS runs
# If grade is 75: prints 'PASSED'
# If grade is 45: prints 'FAILED'
# else catches everything if doesn't match!
```

## If-Elif-Else - Multiple Choices

Format:

- if CONDITION1: # 1st condition
- elif CONDITION2: # 2nd condition
- elif CONDITION3: # 3rd condition
- else: # none matched

Checks conditions in order

First True condition wins, rest skipped!

Use for: A/B/C/D/F grades, categories, tiers

## Letter Grades with If-Elif-Else

```
# Convert number to letter grade
score = 87
if score >= 90:
    letter = 'A'
elif score >= 80:
    letter = 'B'
elif score >= 70:
    letter = 'C'
elif score >= 60:
    letter = 'D'
else:
    letter = 'F'
print(f'Score {score} = Grade {letter}')
# Output: Score 87 = Grade B
# Checks in order, stops at first True
# 87 >= 90? No. >= 80? Yes! → 'B', done!
```

## Order Matters in If-Elif-Else!

First matching condition wins!

❌ WRONG order:

```
if score >= 60: # This catches everything
    letter = 'D'
elif score >= 90:
    letter = 'A' # Never reached
```

✅ CORRECT order:

```
if score >= 90: # Check highest first
    letter = 'A'
elif score >= 60:
    letter = 'D'
```

Always start with most specific condition!

# Conditionals in ML

Machine Learning uses conditionals constantly:

Classification:

- if prediction > 0.5:
-     label = 'positive'
- else:
-     label = 'negative'

Early stopping:

- if accuracy > 0.95:
-     print('Target reached!')
-     break

Data validation:

- if data is None or len(data) == 0:
-     raise ValueError('Empty data!')

You'll write these daily!

## Part 2: Workshop

Build Smart Programs!

## 💻 Exercise 1: Grade Calculator

Create file: grade_calculator.py

Build a grade conversion system:

1. Get a numeric score (use score = 87 for testing)

2. Convert to letter grade (A/B/C/D/F):

   A: >= 90

   B: >= 80

   C: >= 70

   D: >= 60

   F: < 60

3. Print: 'Score 87 = Grade B'

4. Add message for each grade:

   A: 'Excellent!'

   B: 'Good job!'

   C: 'Satisfactory'

   D: 'Needs improvement'

   F: 'Please see instructor'

Test with different scores!

## 💻 Exercise 2: Age Validator

Create file: age_validator.py

Build age validation system:

1. Get age (use age = 25 for testing)

2. Validate:

- If age < 0: 'Invalid age'
- If age < 13: 'Child'
- If age < 18: 'Teen'
- If age < 65: 'Adult'
- If age >= 65: 'Senior'
- If age > 120: 'Invalid age'

3. Print category with message

Test edge cases: -1, 0, 13, 18, 65, 121

This is input validation!

## 💻 Exercise 3: Simple Calculator

Create file: simple_calculator.py

Build calculator with operators:

1. Set: num1 = 10, num2 = 5, operator = '+'

2. Use if-elif-else for operator:

   '+': add

   '-': subtract

   '*': multiply

   '/': divide (check for division by zero!)

   else: 'Invalid operator'

3. Print result: '10 + 5 = 15'

Test all operators!

Bonus: Handle division by zero error

## ☕ Break

Smart programs!

## Part 3: Boolean Logic

Complex conditions

## Boolean Operators - Combining Conditions

Combine multiple conditions:

and - both must be True

- True and True → True
- True and False → False

or - at least one must be True

- True or False → True
- False or False → False

not - reverse/negate

- not True → False
- not False → True

Build complex logic!

## Using 'and' Operator

```
# Both conditions must be True
age = 25
has_license = True

# Can rent car if age >= 21 AND has license
if age >= 21 and has_license:
    print('You can rent a car')
else:
    print('Cannot rent car')

# Both must be True:
# age=25, license=True → Can rent
# age=18, license=True → Cannot (age fails)
# age=25, license=False → Cannot (license fails)
# and requires ALL conditions True!
```

## Using 'or' Operator

```python
# At least one condition must be True
is_weekend = True
is_holiday = False

# Can sleep in if weekend OR holiday
if is_weekend or is_holiday:
    print('Sleep in!')
else:
    print('Set alarm')

# Only one needs to be True:
# weekend=True, holiday=False → Sleep in!
# weekend=False, holiday=True → Sleep in!
# weekend=True, holiday=True → Sleep in!
# weekend=False, holiday=False → Set alarm
# or requires AT LEAST ONE True!
```

## Using 'not' Operator

```
# Reverse a boolean value
is_raining = False

# If NOT raining, go outside
if not is_raining:
    print('Go for a walk!')
else:
    print('Stay inside')

# not reverses the value:
# not True → False
# not False → True

# Another example
is_member = True
if not is_member:
    print('Please sign up')
# Useful for checking False conditions!
```

## Combining Multiple Conditions

Can combine and, or, not together!

Use parentheses for clarity:

- if (condition1 and condition2) or condition3:

Examples:

- (age >= 18 and has_id) or is_member
- score >= 90 and (effort == 'high' or extra_credit)
- not (is_closed or is_full)

Parentheses show what's evaluated first

Like math: (2 + 3) × 4

Build complex logic!

## Complex Condition Example

```
# College admission logic
gpa = 3.8
test_score = 1350
has_recommendation = True

# Accept if:
# (High GPA AND good test) OR has recommendation
if (gpa >= 3.5 and test_score >= 1300) or has_recommendation:
    print('ACCEPTED!')
else:
    print('Not accepted')
# This student: Accepted!
# gpa=3.8 and test=1350 → both True → AND is True
# True OR anything → True
# Multiple paths to acceptance!
```

# Truthy and Falsy Values

Python treats many values as True/False:

Falsy (considered False):

- False (the boolean)
- 0 (zero)
- " “ (empty string)
- [] (empty list)
- None

Truthy (considered True):

- True (the boolean)
- Any non-zero number
- Any non-empty string/list

Shortcuts for checking empty/None!

## Using Truthy/Falsy Values

```
# Check if list has items
data = [1, 2, 3]

# Long way
if len(data) > 0:
    print('Has data')

# Short way (Pythonic!)
if data:
    print('Has data')

# Empty list is falsy
empty = []
if not empty:
    print('List is empty')  # Runs!

# Check if string exists
name = 'Alice'
if name:
    print(f'Hello, {name}')
# Professional Python style!
```

## Conditional Expressions (Ternary Operator)

One-line if-else:

Format:

- value = TRUE_RESULT if CONDITION else FALSE_RESULT

Example:

- status = 'pass' if score >= 60 else 'fail'

Equivalent to:

- if score >= 60:
-     status = 'pass'
- else:
-     status = 'fail'

Compact, professional, clean!

## Ternary Operator Examples

```
# 1. Grade status in one line
score = 85
status = 'PASS' if score >= 60 else 'FAIL'
print(status)  # PASS

# 2. Age category
age = 25
category = 'adult' if age >= 18 else 'minor'

# 3. Even/odd
num = 7
parity = 'even' if num % 2 == 0 else 'odd'

# 4. Max of two numbers
a, b = 10, 15
max_value = a if a > b else b
# Clean and readable!
# Use for simple if-else assignments
```

## Part 4: Workshop

Complex Logic!

## 💻 Exercise 4: Smart Admission System

Create file: admission_system.py

Build college admission logic:

Given:

```
  gpa = 3.7
  test_score = 1280
  extracurriculars = True
  essay_quality = 'excellent'
```

Admission rules (use and/or/not):

1. Auto-accept: GPA >= 3.8 AND test >= 1400
2. Consider: (GPA >= 3.5 AND test >= 1200) OR extracurriculars
3. Interview: essay == 'excellent' AND GPA >= 3.0
4. Reject: all others

Print decision with explanation

Test with different combinations

## Part 5: Quiz & Wrap-up

Session Complete!

## 📝 Session 4.1 Quiz

## Session 4.1 Complete! Smart Programs!

What you mastered today:

- ✓ If/elif/else statements
- ✓ Comparison operators (==, !=, <, >, <=, >=)
- ✓ Boolean operators (and, or, not)
- ✓ Complex conditions
- ✓ Truthy/falsy values
- ✓ Ternary operator (one-line if-else)

You can now:

- Make decisions in code
- Validate data
- Build intelligent programs

Programs that think!

# Homework - Decision Making

1. Complete the quiz (10 questions)

2. Project: Smart Grading System

    - Build: smart_grader.py
    - Input: score, attendance %, participation
    - Calculate final grade with rules:
    - Score >= 90 and attendance >= 80: A
    - Score >= 80 and (attendance >= 70 or participation): B
    - Else: based on score only
    - Print detailed report
    - Use complex boolean logic!

3. Git Practice

    - Feature branch for homework
    - Incremental commits
    - Merge and push

4. Prepare: Read about loops + conditionals (next topic)

## Looking Ahead: Session 4.2

Next session: Combining Conditionals & Loops

You'll learn:

- Filtering data with if in loops
- Data validation
- Smart data processing
- Building complex programs

Why it matters:

- Real data processing
- Quality control
- Intelligent filtering
- Production-ready code

Combining all your skills!

## Week 4 Progress!

Your growth is incredible:

Three weeks ago:

- 'What's Python?'

Today:

- 'I write programs with logic and decisions!'

You've learned:

- Programming fundamentals
- Automation with loops
- Decision-making with conditionals
- Version control with Git

This week: Combining everything

You're over halfway to ML-ready!

See you next session!