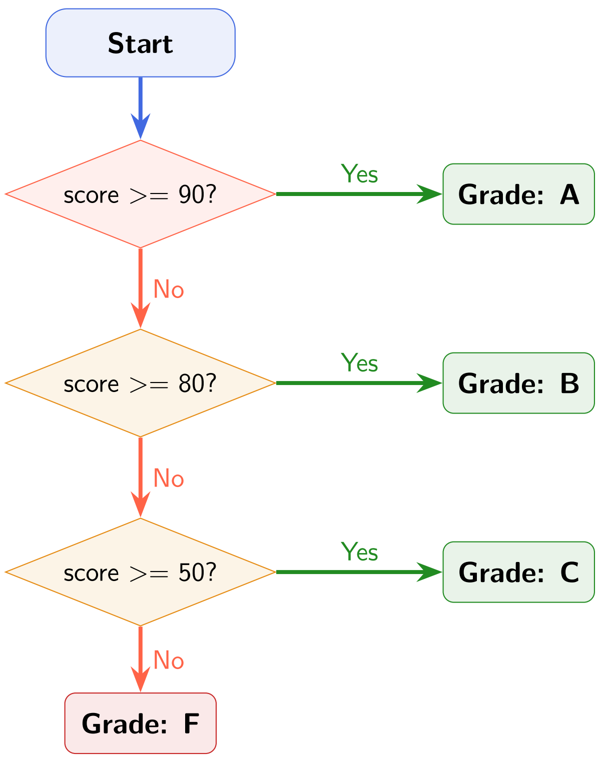
Programming Fundamentals
Lab 03

Conditions & Decision Making

if / elif / else in action



Comp 102 — Forman Christian University

Spring 2026

Estimated Time: ∼**3 hours** — Based on Lectures 5 & 6

How to Use This Lab

This lab is a **walkthrough tutorial**—it is designed to guide you through learning, not to test you. Work through each section at your own pace:

- **Predict** — Write down what you think will happen *before* touching the computer.
- **Type** — Enter the code in Thonny's **Code Editor** (top pane) and press **Run** (or **F5**).
- **Verify** — Compare your prediction with the actual result. If they differ, figure out *why*.
- Exercises are graded by difficulty:
  - ★ Easy — Immediate practice. Follow the pattern you just saw.
  - ★★ Medium — Requires some thinking. Combines concepts.
  - ★★★ Hard — Stretch goal. It's OK to need help!
- **Ask for help** whenever you're stuck for more than 5 minutes.

Prerequisites

This lab assumes you completed **Lab 02**. You should be comfortable with **booleans**, **comparison operators** (`==`, `!=`, `<`, `>`, `<=`, `>=`), **logical operators** (`and`, `or`, `not`), and basic `if`/`else` statements.

1 The `if` Statement (∼15 min)

Ref: Lecture 5 — The `if` Statement

In Lab 02, you got a quick preview of `if/else`. Now let's explore conditions more deeply. Remember: indentation defines what belongs to the `if` body.

1. ★ Easy Predict the output, then type it in the **Code Editor** and run:

```python
x = 10
if x > 5:
    print("x is large")
print("done")
```

Your prediction:

Since `x > 5` is `True`, does the body run? What about `"done"`?

2. ★ Easy Predict the output, then verify:

```python
x = 3
if x > 5:
    print("x is large")
print("done")
```

Your prediction:

The condition is `False`, so the indented body is skipped—but `"done"` still prints. Why?

3. ★★ Medium Predict the output carefully. Which lines are inside the `if` body?

```python
temperature = 38
if temperature > 35:
    print("It is hot!")
    print("Drink water!")
    print("Stay in shade!")
print("Weather report done.")
```

Your prediction:

Now change `temperature = 38` to `temperature = 30`. What prints now?

*All three indented `print` statements are inside the `if` body. The last line is **not** indented, so it runs always.*

4. ★★ Medium Predict the output for **both** values of `num`:

```python
num = 7
if num % 2 == 0:
    print("Even")
else:
    print("Odd")
print("Checked!")
```

With `num = 7`: ______________________
With `num = 4`: ______________________

*Exactly **one** branch runs—never both. `"Checked!"` runs regardless.*

2 The `elif` Chain (∼25 min)

Ref: Lecture 5 — The `elif` Chain

When you have more than two possibilities, use `elif` (short for "else if"). Python checks each condition from top to bottom. The **first** `True` condition wins—all others are skipped.

The Waterfall Model

```python
if condition_1:
    # runs if condition_1 is True
elif condition_2:
    # runs if condition_1 was False AND condition_2 is
      True
elif condition_3:
    # runs if all above were False AND condition_3 is True
else:
    # runs if ALL conditions were False
```

Think of it like a waterfall: the program "falls" from one condition to the next until it finds a True one.

5. ★ Easy Predict the output, then verify:

```python
marks = 72
if marks >= 90:
    print("Grade: A")
elif marks >= 80:
    print("Grade: B")
elif marks >= 70:
    print("Grade: C")
else:
    print("Grade: F")
```

Your prediction: ____________________

Which condition is the ***first*** *one that is True?*

6. ★★ Medium **Bug hunt!** This student used separate `if` statements instead of `elif`. Predict the output with `score = 95`:

```python
score = 95
if score >= 90:
    grade = "A"
if score >= 80:
    grade = "B"
if score >= 70:
    grade = "C"
print(grade)
```

Your prediction: ____________________

Why doesn't it print `"A"`?

With separate `ifs`*, every condition is checked independently—so later ones can overwrite earlier ones!*

7. ★★ Medium **Trace on paper first**—do not type this yet. Work through each condition with `hour = 14`:

```python
hour = 14
if hour < 6:
    print("Good night")
elif hour < 12:
    print("Good morning")
elif hour < 17:
    print("Good afternoon")
elif hour < 21:
    print("Good evening")
else:
    print("Good night")
```

Check `hour < 6`: $14 < 6$? __________
Check `hour < 12`: $14 < 12$? __________
Check `hour < 17`: $14 < 17$? __________ $\rightarrow$ prints: ____________________
Are the remaining conditions checked? __________

Now verify in Thonny. Then try `hour = 3`*—which branch runs?*

8. ★★★ Hard **Order matters!** Someone wrote the conditions in the wrong order. Predict the output with `marks = 95`:

```python
marks = 95
if marks >= 50:
    print("Pass")
elif marks >= 80:
    print("Merit")
elif marks >= 90:
    print("Distinction")
```

Your prediction: ____________________

Why is this wrong? What should the correct order be?

*Since `elif` stops at the first True, you must check the **most restrictive** condition first.*

9. ★★★ Hard **Write a program** that classifies temperature:

- Below 0: print `"Freezing"`
- 0 to 15 (inclusive): print `"Cold"`
- 16 to 30 (inclusive): print `"Pleasant"`
- Above 30: print `"Hot"`

Test with `temp = -5`, `temp = 10`, `temp = 25`, and `temp = 40`.

Write your code:

3 Nested `if` Statements (∼25 min)

Ref: Lecture 5 — Nested Conditions

You can put an `if` statement *inside* another `if` statement. The inner condition is only checked when the outer condition is `True`.

10. ★ Easy Predict the output, then verify:

```python
age = 20
has_ticket = True
if age >= 18:
    if has_ticket:
        print("Welcome to the movie!")
```

Your prediction: ____________________

Both conditions are True, so the inner body runs.

11. ★★ Medium Now change `age = 20` to `age = 15`. Predict:

```python
age = 15
has_ticket = True
if age >= 18:
    if has_ticket:
        print("Welcome to the movie!")
print("Goodbye")
```

Your prediction:

Is `has_ticket` ever checked? __________

*When the outer condition is False, Python skips **everything** inside it—including the inner `if`.*

12. ★★ Medium **Trace on paper first.** Work through this login checker for each case below, then verify in Thonny:

```python
username = "admin"
password = "1234"
if username == "admin":
    if password == "secret":
        print("Login successful")
    else:
        print("Wrong password")
else:
    print("Unknown user")
```

Case 1 (`username="admin"`, `password="1234"`): ____________________
Case 2 (`username="admin"`, `password="secret"`): ____________________
Case 3 (`username="guest"`, `password="secret"`): ____________________

*Nesting lets you give **specific** error messages.*

13. ★★★ Hard The nested code below can be rewritten using `and`. Fill in the equivalent flat version:

```python
# Nested version:
if age >= 18:
    if has_id:
        print("You may enter")
```

```python
# Flat version (fill in the blank):
if ____________:
    print("You may enter")
```

Your answer: ____________________

When would you prefer nesting over using `and`?

Hint: Nesting is better when you need different messages for different failure reasons.

14. ★★★ Hard **Write a program** for a theme park ride with these rules:

- You must be at least 120 cm tall
- If you are under 120 cm, print `"Too short for this ride"`
- If you are tall enough but under 12 years old, print `"You need an adult companion"`
- If you are tall enough and 12 or older, print `"Enjoy the ride!"`

The program should ask for height and age using `input()`.

Write your code:

4 Combining Conditions in Practice (∼20 min)

Ref: Lecture 5 — Conditions with Logical Operators

You already know `and`, `or`, and `not` from Lab 02. Now let's use them inside `if` statements to make real decisions.

15. ★ Easy Predict the output, then verify:

```python
age = 20
gpa = 3.5
if age >= 18 and gpa >= 3.0:
    print("Eligible for scholarship")
else:
    print("Not eligible")
```

Your prediction: ____________________

Now try `age = 20, gpa = 2.5`. What changes? ____________________

16. ★ Easy Predict the output, then verify:

```python
day = "Saturday"
if day == "Saturday" or day == "Sunday":
    print("Weekend!")
else:
    print("Weekday")
```

Your prediction: ____________________

Now try `day = "Monday"`: ____________________

17. ★★ Medium Python supports **chained comparisons**. Predict the output:

```python
x = 7
if 1 <= x <= 10:
    print("In range")
else:
    print("Out of range")
```

Your prediction: ____________________

This is equivalent to `if x >= 1 and x <= 10`. Try `x = 15`: ____________________

18. ★★ Medium A cinema gives discounts to children (under 12), seniors (65 and over), or students. Predict the output:

```python
age = 70
is_student = False
if age < 12 or age >= 65 or is_student:
    print("Discount applies!")
else:
    print("Full price")
```

Your prediction: ____________________

Which of the three `or` conditions is True? ____________________

19. ★★★ Hard **Write a program** that checks password strength:

- Ask the user for a password
- If the password is less than 8 characters: print `"Too short"`
- If 8 characters or more **and** contains no spaces: print `"Strong password"`
- If 8 characters or more **but** contains a space: print `"No spaces allowed"`

Hint: Use `" " in password` to check for spaces.

Write your code:

5 Common Pitfalls & Debugging (~15 min)

Ref: Lecture 5 — Common Mistakes

20. ★ Easy **Spot the bug!** This code has a `SyntaxError`. What's wrong?

```python
x = 10
if x > 5
    print("big")
```

The bug: ____________________

*Every `if`, `elif`, and `else` line must end with a **colon** (`:`).*

21. ★ Easy **Spot the bug!** This code runs but gives the wrong result:

```python
password = "hello"
if password = "hello":
    print("Correct!")
```

The bug: ____________________

Remember: `=` is assignment, `==` is comparison.

22. ★★ Medium **Spot the bug!** This code has an `IndentationError`. What's wrong?

```python
age = 20
if age >= 18:
print("Adult")
    print("Can vote")
```

The bug: ____________________
Fix: ____________________

23. ★★ Medium What does the `pass` keyword do? Predict the output:

```python
x = 5
if x > 10:
    pass
else:
    print("x is small")
print("done")
```

Your prediction:

`pass` is a placeholder that does nothing. It's useful when you need an `if` body but haven't written the code yet.

24. ★★ Medium Predict the output for **both** cases:

```python
name = "Ali"
if name:
    print(f"Hello, {name}!")
else:
    print("No name provided")
```

With `name = "Ali"`: ____________________
With `name = ""`: ____________________

*A non-empty string is **Truthy**; an empty string is **Falsy**.*

6 Tracing Practice (∼20 min)

Ref: Lecture 5 — Putting It All Together

These exercises require you to carefully trace through longer programs. **Use pencil and paper**—work through each line step by step *before* touching the computer. Only verify in Thonny after you have written down your answer.

25. ★★ Medium **Trace on paper first.** Work through each condition with pencil before verifying in Thonny:

```python
x = 15
y = 8
z = True
if x > 10:
    if y > 10:
        print("Both large")
    elif z:
        print("x large, z is True")
    else:
        print("x large only")
else:
    print("x is small")
```

Is `x > 10`? __________ Is `y > 10`? __________ Is `z` True? __________
Output: ____________________

26. ★★★ Hard **FizzBuzz!** Trace on paper for `n = 15`, then `n = 9`, then `n = 7`:

```python
n = 15
if n % 15 == 0:
    print("FizzBuzz")
elif n % 3 == 0:
    print("Fizz")
elif n % 5 == 0:
    print("Buzz")
else:
    print(n)
```

`n = 15`: ____________________ `n = 9`: ____________________ `n = 7`: ____________________

Why must we check `n % 15 == 0` **before** `n % 3` and `n % 5`?

27. ★★★ Hard **Trace on paper** the leap year checker for `year = 2000`, `year = 1900`, and `year = 2024`:

```python
year = 2000
if year % 4 != 0:
    print("Not a leap year")
elif year % 100 != 0:
    print("Leap year")
elif year % 400 == 0:
    print("Leap year")
else:
    print("Not a leap year")
```

year = 2000: ______________________
year = 1900: ______________________
year = 2024: ______________________

Trace each condition step by step for every year.

7 Capstone Challenges (∼20 min)

These problems combine everything from Part 1. Write your solutions in Thonny's code editor.

28. ★★ Medium **Even/Odd with a Twist.** Write a program that:

1. Asks the user for an integer
2. If the number is 0, prints `"Zero is special --- it is even but neither positive nor negative"`
3. If the number is even (and not 0), prints `"Even"`
4. If the number is odd, prints `"Odd"`

Test with `0`, `7`, and `-4`.

Write your code:

29. ★★★ Hard **Simple Calculator.** Write a program that:

1. Asks the user for two numbers
2. Asks for an operator (`+`, `-`, `*`, `/`)
3. Uses an `if`/`elif`/`else` chain to compute and print the result
4. If the operator is `/` and the second number is 0, prints `"Cannot divide by zero"`
5. If the operator is invalid, prints `"Unknown operator"`

Example:

```
Enter first number: 10
Enter second number: 3
Enter operator (+, -, *, /): *
Result: 30.0
```

Write your code:

30. ★★★ Hard **Letter Grade with Modifiers.** Write a program that:

1. Asks the user for marks (0–100)
2. Assigns a letter grade: A (90–100), B (80–89), C (70–79), D (60–69), F (below 60)
3. Adds a `+` if the ones digit is 7, 8, or 9 (e.g., 87 → B+)
4. Adds a `-` if the ones digit is 0, 1, or 2 (e.g., 80 → B−)
5. Exception: there is no A+ (cap at A) and no F+ or F−

Example: input `87` → `"B+"`   input `91` → `"A-"`   input `72` → `"C-"`

Write your code:

8 CCC J1 Practice Problems (~30 min)

Ref: Lecture 6 — CCC Problem Solving

The **Canadian Computing Competition (CCC)** Junior Division Problem 1 (J1) tests your ability to read a problem, identify the conditions, and write a correct program.

Each problem is on the following pages. Read the problem statement carefully, then write your Python solution in Thonny. **Then submit your solution to the DMOJ online judge to see if it passes all test cases.**

CCC '04 J1 - Squares

Time limit: 2.0s **Memory limit:** 256M

Canadian Computing Competition: 2004 Stage 1, Junior #1

Gigi likes to play with squares. She has a collection of equal-sized square tiles. Gigi wants to arrange some or all of her tiles on a table to form a solid square. What is the side length of the largest possible square that Gigi can build?

For example, when Gigi has 9 tiles she can use them all to build a square whose side length is 3. But when she has only 8 tiles, the largest square that she can build has side length 2.

Write a program that inputs the number of tiles and then prints out the maximum side length. You may assume that the number of tiles is less than ten thousand.

Sample Input 1

```
9
```

Sample Output 1

```
The largest square has side length 3.
```

Sample Input 2

```
8
```

Sample Output 2

```
The largest square has side length 2.
```

Sample Input 3

```
7535
```

Sample Output 3

```
The largest square has side length 86.
```

CCC '02 J1 - 0123456789

Time limit: 0.5s **Memory limit:** 256M

Canadian Computing Competition: 2002 Stage 1, Junior #1

```
 _     _  _     _  _  _  _  _
| |  | _| _||_||_ |_   ||_||_|
|_|  ||_  _|  | _||_|  ||_| _|
```

Most digital devices show numbers using a seven segment display. The seven segments are arranged as shown:

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

For this problem, each segment is represented by three asterisks in a line as shown above.

Any digit from 0 - 9 can be shown by illuminating the appropriate segments. For example, the digit 1 may be displayed using the two segments on the right side:

```
      *
      *
      *

      *
      *
      *
```

The digit 4 needs four segments to display it properly:

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

Write a program that will accept a single digit input from the user, and then display that digit using a seven segment display. You may assume that each segment is composed of three asterisks.

DMOJ-specific note: None of your lines should contain any trailing whitespace. The last line must end with a newline.

Sample Input

```
9
```

Sample Output

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

CCC '07 J1 - Who is in the Middle?

Time limit: 2.0s **Memory limit:** 256M

Canadian Computing Competition: 2007 Stage 1, Junior #1

In the story Goldilocks and the Three Bears, each bear had a bowl of porridge to eat while sitting at his/her favourite chair. What the story didn't tell us is that Goldilocks moved the bowls around on the table, so the bowls were not at the right seats anymore. The bowls can be sorted by weight with the lightest bowl being the Baby Bear's bowl, the medium bowl being the Mama Bear's bowl and the heaviest bowl being the Papa Bear's bowl.

Write a program that reads in three weights and prints out the weight of Mama Bear's bowl. You may assume all weights are positive integers less than 100.

Sample Input

```
10
5
8
```

Sample Output

```
8
```

CCC '05 J1 - The Cell Sell

Time limit: 2.0s **Memory limit:** 256M

Canadian Computing Competition: 2005 Stage 1, Junior #1

Moe Bull has a cell phone and after a month of use is trying to decide which price plan is the best for his usage pattern. He has two options, each plan has different costs for daytime minutes, evening minutes and weekend minutes.

Plan	Costs		
	daytime	evening	weekend
A	100 free minutes then 25 cents per minute	15 cents per minute	20 cents per minute
B	250 free minutes then 45 cents per minute	35 cents per minute	25 cents per minute

Write a program that will input the number of each type of minutes and output the cheapest plan for this usage pattern, using the format shown below. The input will be in the order of daytime minutes, evening minutes and weekend minutes. In the case that the two plans are the same price, output both plans.

Sample Input 1

```
251
10
60
```

Sample Output 1

```
Plan A costs 51.25
Plan B costs 18.95
Plan B is cheapest.
```

Sample Input 2

```
162
61
66
```

Sample Output 2

```
Plan A costs 37.85
Plan B costs 37.85
Plan A and B are the same price.
```

CCC '06 J1 - Canadian Calorie Counting

Time limit: 2.0s **Memory limit:** 256M

Canadian Computing Competition: 2006 Stage 1, Junior #1

At Chip's Fast Food emporium there is a very simple menu. Each food item is selected by entering a digit choice.

Here are the three burger choices: 1 – Cheeseburger (461 Calories) 2 – Fish Burger (431 Calories) 3 – Veggie Burger (420 Calories) 4 – no burger	Here are the three drink choices: 1 – Soft Drink (130 Calories) 2 – Orange Juice (160 Calories) 3 – Milk (118 Calories) 4 – no drink
Here are the three side order choices: 1 – Fries (100 Calories) 2 – Baked Potato (57 Calories) 3 – Chef Salad (70 Calories) 4 – no side order	Here are the three dessert choices: 1 – Apple Pie (167 Calories) 2 – Sundae (266 Calories) 3 – Fruit Cup (75 Calories) 4 – no dessert

Write a program that will compute the total Calories of a meal.

Input Specification

The program should input a number for each type of item then calculate and display the Calorie total. The first line will be the customer's choice of burger, the second will be the choice of side, then drink, then dessert. You may assume that each input will be a number from 1 to 4. That is, each customer has to pick exactly one number from each of the four options out of each of the four categories.

Output Specification

The program prints out the total Calories of the selected meal, and stops executing after this output.

Sample Input

```
2
1
3
4
```

Sample Output

```
Your total Calorie count is 649.
```

Explanation

The customer chose Burger #2, Side #1, Drink #3 and Dessert #4.

CCC '08 J1 - Body Mass Index

Time limit: 2.0s **Memory limit:** 256M

Canadian Computing Competition: 2008 Stage 1, Junior #1

The Body Mass Index (BMI) is one of the calculations used by doctors to assess an adult's health. The doctor measures the patient's height (in metres) and weight (in kilograms), then calculates the BMI using the formula:

$$BMI = \frac{weight}{height \times height}$$

Write a program which takes the patient's weight and height as input, calculates the BMI, and displays the corresponding message from the table below.

BMI Category	Message
More than 25	Overweight
Between 18.5 and 25.0 (inclusive)	Normal weight
Less than 18.5	Underweight

Sample Input 1

```
69
1.73
```

Output for Sample Input 1

```
Normal weight
```

Explanation of Output for Sample Input 1

The BMI is $\frac{69}{1.73 \times 1.73}$ which is approximately 23.0545. According to the table, this is a "Normal weight".

Sample Input 2

```
84.5
1.8
```

Output for Sample Input 2

```
Overweight
```

Explanation of Output for Sample Input 2

The BMI is $\frac{84.5}{1.8 \times 1.8}$, which is approximately 26.0802. According to the table, this is "Overweight".

Self-Assessment Checklist

Before you leave, check off what you can do:

- ☐ I can write an `if` statement and I know that **indentation** defines what's inside the body
- ☐ I can write an `if/else` and I know that exactly **one** branch runs
- ☐ I can write an `if/elif/else` chain and I know the **first** True condition wins
- ☐ I understand why the order of `elif` conditions matters (most restrictive first)
- ☐ I can write **nested** `if` statements and I know the inner condition is only checked when the outer is True
- ☐ I know when to use nesting vs. a flat `and` condition
- ☐ I can combine `and`, `or`, and `not` in `if` conditions to make complex decisions
- ☐ I can use Python's chained comparisons (e.g., `1 <= x <= 10`)
- ☐ I know the three common pitfalls: missing colon, `=` vs `==`, and indentation errors
- ☐ I know what `pass` does and when to use it
- ☐ I can trace the execution path through a multi-branch `if/elif/else` program
- ☐ I can translate a word problem into conditions and write a working program
- ☐ I can identify and fix bugs in conditional code