Class

5

CSC 141 Computer Science I

elif, Nested Decisions & Program Design

Dr. Si Chen (schen@wcupa.edu)

Tue / Thu · 11:00 AM – 12:15 PM · Mitchell Hall 102
Fall 2026 · Week 3 · Thursday, September 10, 2026
PA1 now due Thu Sep 17 · Quiz 1 moved to Tue Sep 15 ·
wcu.ninja/csc141

Two schedule changes — write these down

- **Programming Assignment 1 is now due Thursday, September 17, 11:59 PM** — one extra week. Nothing else about it changed; the updated handout is on the course site.
- **Quiz 1 has moved to Tuesday, September 15**, first 15 minutes of class.
- Because the quiz is now *after* today, it covers **Classes 1–5** — including everything in today's class: `elif` ladders, nested decisions, flags, and program design.
- Use the extra week well: finish PA1 early, then re-read your own code and test the boundaries.

Recap from Tuesday

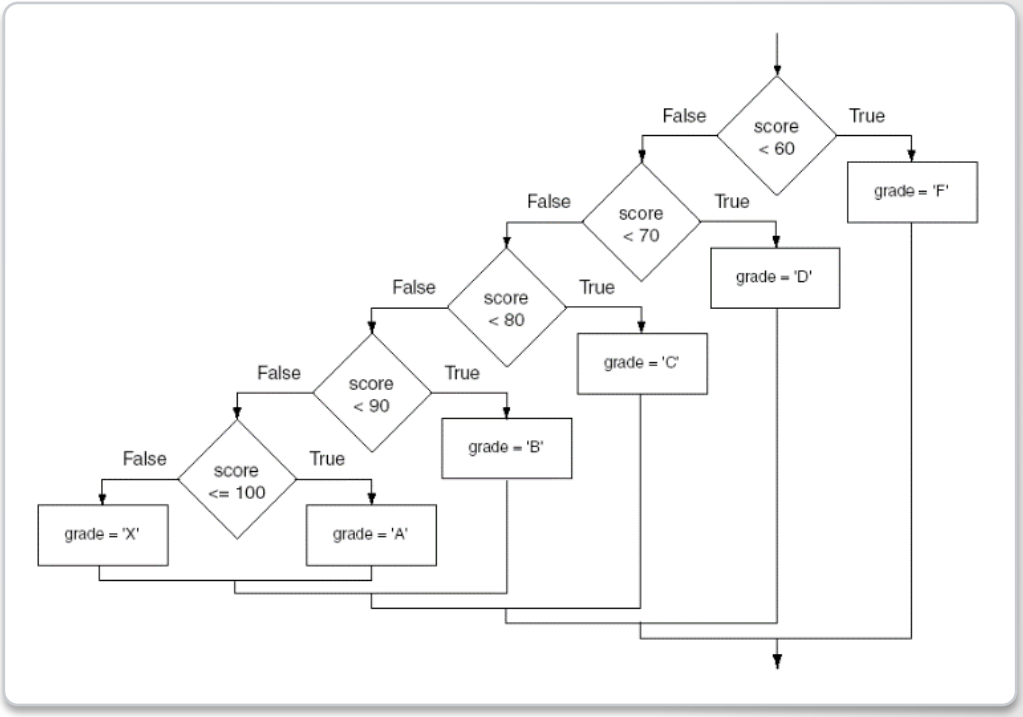
- Comparisons produce `bool`; `==` compares, `=` assigns.
- `if condition:` + indented block; `else:` for the other path; **exactly one** path runs.
- Strings compare by **character code**; use `.lower()` for case-insensitive checks.
- `and` / `or` / `not`; chained comparisons `1 <= x <= 100`; beware `x == "A" or "B"`.
- Today: **more than two paths**, **decisions inside decisions**, and **designing before coding**.

PART 1

if / elif / else

When there are more than two possibilities — one ladder, exactly one rung runs

The grade ladder as a flowchart



- Five possible grades → a **chain** of questions. Each question is asked **only if all previous ones were False**.
- Reading the chart: `score < 60` → F; otherwise `score < 70` → D; otherwise …
- The flowchart is 2018 Java; the Python is one `if` + several `elif` + a final `else`.
- Notice the last diamond, `score <= 100`: the original even handled **invalid input** ("X").

Grade-letter ladder, my 2018 Class 9 deck. Java if-else-if → Python if / elif / else.

if / elif / else in Python

```
score = float(input("Score (0-100)? "))

if score >= 90:
    letter = "A"
elif score >= 80:    # only if score < 90
    letter = "B"
elif score >= 70:
    letter = "C"
elif score >= 60:
    letter = "D"
else:
    letter = "F"

print(f"Score {score:.1f} -> {letter}")
```

- `elif` = "else if". Python checks conditions **top to bottom** and runs the **first** one that is `True`.
- After that block, it **skips the rest** of the ladder — so `elif score >= 80` never needs `and score < 90`.
- `else` catches everything that fell through. It is optional but usually wise.
- **Order matters.** Put `>= 60` first and every passing score becomes "D". Try it.

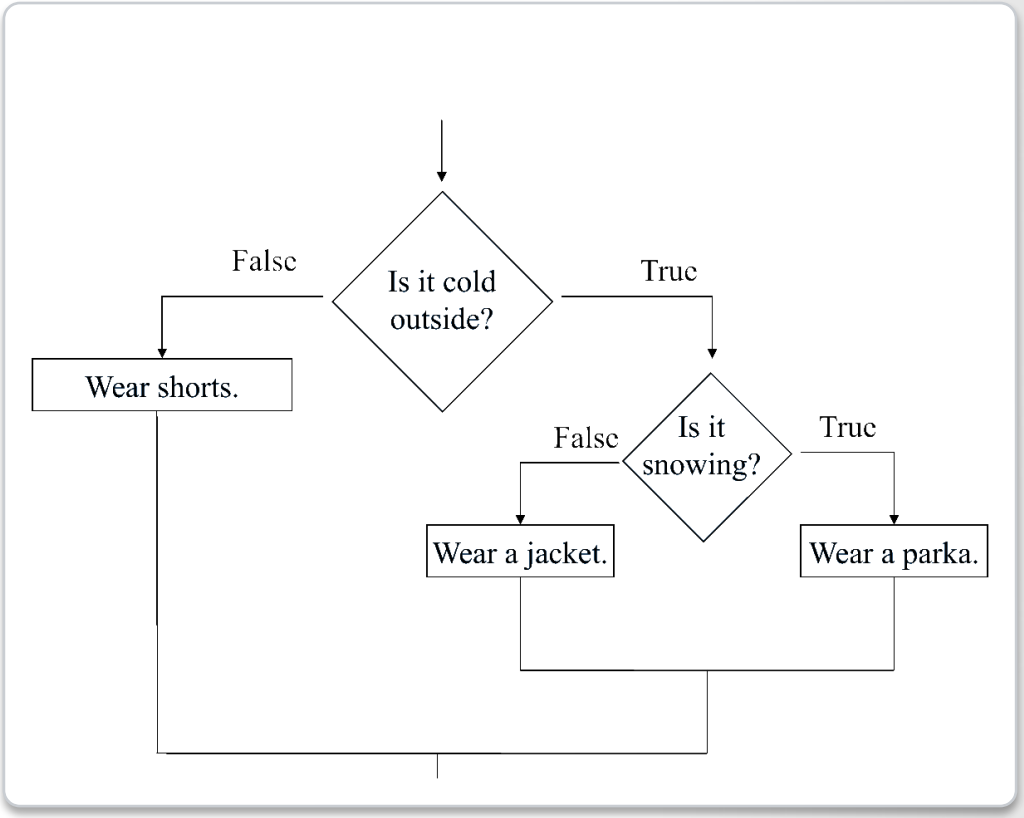
grade_letter.py in the Class 5 bundle — break it on purpose by reordering the rungs.

PART 2

Nested decisions

A decision inside a decision — and when to flatten it with and

Cold outside? Then: is it snowing?



- The second question is asked **only on the True path** of the first — it is **nested** inside it.
- Three outcomes: shorts / jacket / parka.
- In code, nesting = **indenting one if inside another**. Each level adds 4 spaces.
- The `else` that matches an `if` is the one at the **same indentation** — Python makes the pairing visible.

Nested if flowchart, 2018 Class 9 deck.

Nested vs. flat: two ways to write the same rule

nested_if.py

```
if cold_outside:
    if snowing:
        print("Wear a parka.")
    else:
        print("Wear a jacket.")
else:
    print("Wear shorts.")

# loan: nested gives SPECIFIC messages
if salary >= 30000:
    if years >= 2:
        print("Qualified.")
    else:
        print("Need 2 years on the job.")
else:
    print("Need $30,000 salary.")
```

Which one to choose?

- **Nest** when the inner question only makes sense on one path, or when each failure needs its **own message**.
- **Flatten** with `and` when you only need yes/no:
- `if salary >= 30000 and years >= 2:`

- Two levels is normal. **Three or more** is a smell — usually a sign to use `elif`, `and`, or (Week 5) a **function**.

- Readability rule: a reader should know which `else` belongs to which `if` **at a glance**. Indentation does that for you — keep it consistent.

PART 3

Flags, truthiness & one-line decisions

Small tools that make decision code shorter and clearer

Flags: a bool that remembers

```
score = int(input("Score? "))
high_score = False       # flag starts "down"

if score > 95:
    high_score = True    # raise the flag

# ... later in the program ...
if high_score:           # NOT: == True
    print("New high score!")

# shorter: store the comparison directly
high_score = score > 95
```

- A **flag** is a `bool` variable that records whether something happened (2018: `isHighScore`).
- Test a flag with `if high_score:` — it already **is** a bool; `== True` is redundant.
- Flags shine when the check happens in one place and the reaction somewhere else (loops, Week 7).
- Naming: `is_…`, `has_…`, `…_found` read as yes/no questions.

Truthiness: Python treats some values as False

The rule

- `if` does not require a bool — **any** value works.
- **Falsy** values: `0`, `0.0`, `""` (empty string), `None`, (later: empty lists).
- **Everything else is truthy**: `42`, `-1`, `"no"`, `" "`.
- So `if name:` means "if the user typed something".
- This is why `grade == "A" or "B"` misbehaves: `"B"` is truthy, so the whole `or` is `True`.

flags_and_truthiness.py

```
name = input("Name (or Enter)? ")
if name:                  # same as
name != ""
    print(f"Hello, {name}!")
else:
    print("No name entered.")

if not name:
    print("(also works)")

count = 0
if count:                 # 0 is falsy
    print("never printed")
```

One-line decisions: the conditional expression

```
# The 2018 ConsultantCharges rule: bill at least 5
hours
hours = float(input("Hours worked? "))

# long form:
if hours < 5:
    billable = 5
else:
    billable = hours

# same thing, one line:
billable = 5 if hours < 5 else hours

# and often clearest of all:
billable = max(5, hours)
label = "even" if n % 2 == 0 else "odd"
```

- `value_if_true if condition else value_if_false` — Python's version of Java's `? :`.
- Use it when you are **choosing a value**, not running different actions.
- If it does not fit on one readable line, use the long form.
- Sometimes a built-in (`max`, `min`, `abs`) says it better than any `if`.

Coming from Java? Three differences

No switch fall-through

Java's `switch` needed `break` on every case or execution **fell through** (the 2018 NoBreaks demo).

Python has no such trap: use `if / elif / else`, or `match / case` (Python 3.10+) — no `break` needed.

Blocks by indentation

No `{ }`. The block is the indented lines.

A misplaced brace becomes a misplaced indent — same bug, different look.

Strings compare with ==

No `.equals()`. `==` compares text; `.lower()` for case-insensitive.

`is` exists but means "same object" — do not use it for text.

PART 4

Design before you code

Problem → pseudocode → test cases → code → test. Decisions are where this pays off

The five steps (do them in this order)

1 · Understand

Restate the problem in one sentence.

List **inputs**, **outputs**, and the **rules**.

2 · Pseudocode

Plain-English steps, in order.

Every decision becomes an "if … otherwise …" line.

3 · Test cases FIRST

Pick inputs **and** expected outputs before coding.

Always include the **boundaries**: exactly 18.5, exactly 25, exactly 100.

4 · Code

Translate the pseudocode line by line.

Constants for the magic numbers.

5 · Test & fix

Run every test case. Compare, don't eyeball.

A wrong boundary is a **one-character fix** (< vs <=) — if you tested for it.

HANDS-ON ⏱ 12 min

Hands-on: BMI advisor — design first, then code

- **Problem:** read weight (lb) and height (in); `bmi = 703 × weight / height²`; print BMI (1 decimal) **and** a category: < 18.5 underweight · < 25 normal · < 30 overweight · else obese.
- **Steps 1–2 (3 min, paper):** inputs, outputs, pseudocode ladder.
- **Step 3 (2 min):** 4 test cases incl. two **boundaries** — exactly 25.0? exactly 18.5?
- **Steps 4–5 (7 min):** code at **wcu.ninja/csc141/python?class=5**; run your tests.
- **Stretch:** ladder in the *opposite* order (start `>= 30`). Still correct? Why?

Test cases:

150 lb, 68 in → 22.8 normal

100 lb, 66 in → 16.1 under

bmi 25.0 → ? bmi 18.5 → ?

Ladder:

```
if bmi < 18.5: ...
elif bmi < 25: ...
elif bmi < 30: ...
else: ...
```

Answers: 25.0 → overweight; 18.5 → normal (< is strict).

Reading code: three silent bugs (AI-written)

```
total = float(input("Order total? "))
member = input("Member? (yes/no) ")

if member = "yes":              # BUG 1
    discount = 0.10
else:
    discount = 0.0

if total >= 50:                 # BUG 2
    discount += 0.05
elif total >= 100:
    discount += 0.20            # never reached!

if member == "Yes":             # BUG 3
    print("Thanks for being a member!")
```

- **Bug 1:** `=` instead of `==`. Python refuses to run it — **read the SyntaxError**, it points at the line.
- **Bug 2:** ladder order. `>= 50` catches 120 first; the 20% rung is dead code. Silent — no error.
- **Bug 3:** case. The user typed `yes`; the check wants `Yes`. Silent. Fix: `member.lower() == "yes"`.
- Two of three bugs produce **no error message**. Only reading + boundary tests catch them.

debug_decisions.py (Class 5 bundle) has the fixed version with the bugs explained in comments.

Before next class (Tue, Sep 15)

- **Quiz 1 is Tuesday**, first 15 minutes: Classes 1–5. Study the code bundles and your own PA1 code; practise *reading* code, not just writing it.
- **PA1 due Thursday, Sep 17, 11:59 PM** (D2L). You have an extra week — do not spend all of it on the last program.
- Practice: write a **season finder** — month number 1–12 → Winter/Spring/Summer/Fall, with an `else` for invalid months. Design the test cases first.
- Read the Class 5 code bundle: `grade_letter.py`, `nested_if.py`, `leap_year.py` — predict each output before running.
- Next week (Week 4): **nested decisions in bigger programs, testing, debugging, tracing**. **PA2 assigned Tuesday.**

Sources

- Grade-ladder and nested-if flowcharts, Flags, ConsultantCharges, loan and NoBreaks/switch examples: Dr. Si Chen, CSC 141 Spring 2018 Class 9–12 decks (Java), translated to Python.
- Python Language Reference §6.13 (conditional expressions), §4.1 (truth value testing / falsy values).
- Python Tutorial §4.1 (if / elif / else); PEP 636 (structural pattern matching, match/case, Python 3.10).
- CDC, "About Adult BMI" (BMI = 703 × lb / in²; categories 18.5 / 25 / 30).
- Gaddis, Starting Out with Java, Ch. 3 (Decision Structures) — original source of several 2018 examples.
- Stack Overflow, "2025 Developer Survey — AI" (context for the "almost right" framing from Class 4).