Sequence, Selection, and Iteration

Table of contents

See csse1001[1] for course logistics — this note covers Lecture 3B's technical content.

Today's outline

Despite the title, this lecture is entirely about **selection** (controlling program flow) — sequencing was already covered in 2025-08-04-python-memory-model[2], and iteration is the topic of the next lecture.

Learning objectives

1. Predicates and the logical operators `and`, `or`, `not` let us build up complex conditions.
2. The `if` statement (and its `elif`/`else` variants) lets us skip or select blocks of code based on a condition.
3. If-statements can often be refactored into simpler, equivalent forms.

[1] courses/csse1001/csse1001.pdf
[2] courses/csse1001/2025-08-04-python-memory-model.pdf

Selection

See python-boolean-logic[3] for the full reference on booleans, predicates, `and`/`or`/`not`, precedence, short-circuiting, and truthiness, and python-if-statements[4] for the full reference on `if`/`elif`/`else`, the common elif-ordering bug, and simplifying if-statements.

Exercises

Bracket for `False`. Bracket the expression below so that it evaluates to `False`. How many different bracketings can you find?

```
False and False or True and False or False or True
```

Add a single `not`. Add a single `not` to the same expression so that it evaluates to `False`.

> Both exercises are posed but left unsolved in the source material — flagged here rather than guessed at.

Refactoring exercise. Refactor the following code:

```python
def foo(x, y):
    if x > 100 and y > 0:
        if y > 100 and x > 0:
            return "A"
        elif y > 100 or x > 0:
            return "A"
        else:
            return "B"
    elif y <= 0 or x <= 0:
        if x == y:
            return "A"
        if x <= y and x >= y:
            return "B"
        if y < x and x < y:
            return "C"
        else:
            return "A"
    else:
        if x <= 100 and y > 0:
            return "A"
        if x > 100 or y <= 0:
            return "B"
```

[3] courses/csse1001/python-boolean-logic.pdf
[4] courses/csse1001/python-if-statements.pdf

```
        else:
            return "D"
```

No worked solution is given in the source for this exercise either — left as an exercise rather than invented here.

Summary

Blocks of code can be skipped using `if` statements. This control flow depends on the evaluation of predicate (boolean-valued) statements.

Next lecture

2025-08-18-while-loops[5] — iteration.

[5] courses/csse1001/2025-08-18-while-loops.pdf