COMP110

CL18: Algorithmic Complexity + Quiz Review

Announcements

- EX04 (Dictionary Utils) due *today* (Oct 8)!
- Quiz 02 on Friday
    - Practice questions on the site
    - Review session on Thursday; look for an announcement on Canvas!
    - Please visit Office Hours and Tutoring for help!

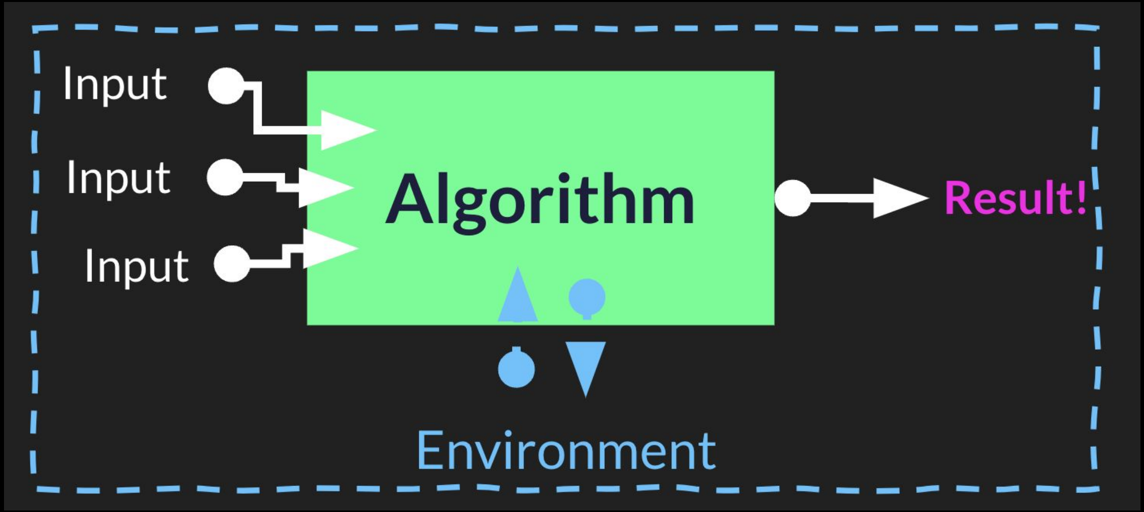
Review: Algorithms

Input is data given to an algorithm

An **algorithm** is a series of steps

An algorithm **returns** some **result**

An algorithm *may* be influenced by its **environment** and it *may* produce side-effects which influence its environment.



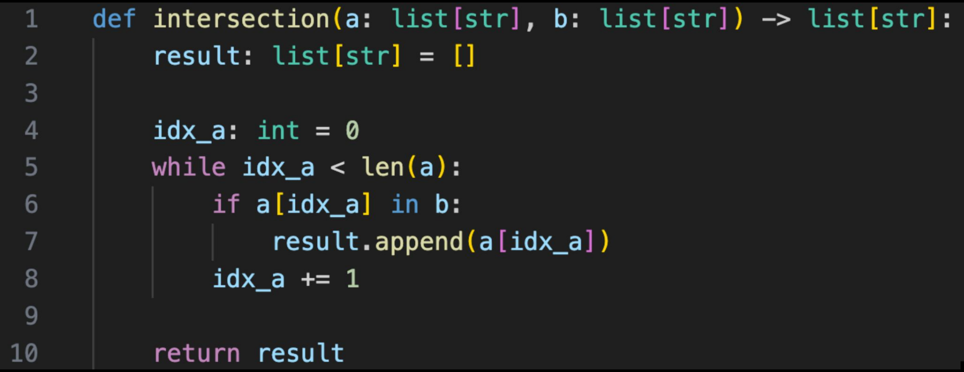
Warm-up:

With a partner, discuss which of the following factors you think is *most important* to consider when selecting or implementing an algorithm:

- Simplicity
- Ease of implementation
- Speed (how long does the code take to run?)
- Efficient use of memory
- Precision in answer

Is one factor *always* more important than the others, or would it vary by algorithm?

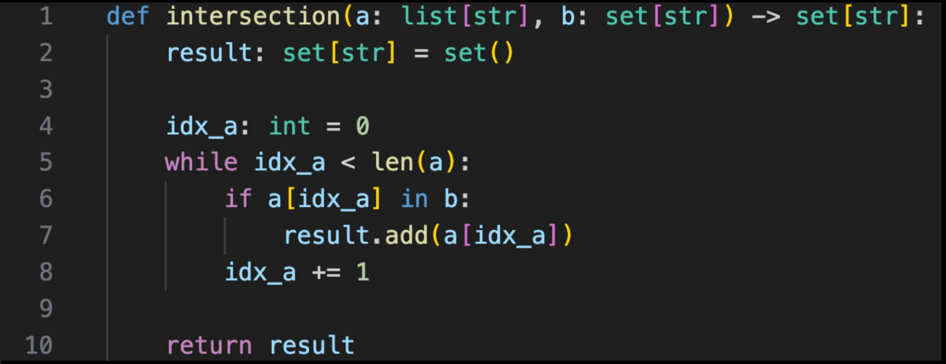
Recall: comparing lists and sets



```python
def intersection(a: list[str], b: list[str]) -> list[str]:
    result: list[str] = []

    idx_a: int = 0
    while idx_a < len(a):
        if a[idx_a] in b:
            result.append(a[idx_a])
        idx_a += 1

    return result
```



```python
def intersection(a: list[str], b: set[str]) -> set[str]:
    result: set[str] = set()

    idx_a: int = 0
    while idx_a < len(a):
        if a[idx_a] in b:
            result.add(a[idx_a])
        idx_a += 1

    return result
```

Suppose `a` and `b` each had 1,000,000 elements. The worst case difference here is approximately 1,000,000 operations, versus 1,000,000**2 or 1 trillion (1,000,000,000,000) operations.

If your device can perform 100,000,000 operations per second, then...

A call to `a` will complete in 2.78 hours and `b` will complete in 1/100th of a second.

Running time: how long does an algorithm take to run?

- Empirical analysis: write the code and test how long it takes to run!
    - Weaknesses:
        - You have to write the code for the whole algorithm and run it to see how long it will take
        - Different computers with different specs will have different runtimes
- Rather than using empirical analysis, computer scientists commonly consider the **number of operations (steps)** an algorithm requires
    - 1 operation == 1 step

Runtime analysis: Best, average, and worst case

Best case (lower bound):

- Minimum number of operations (running time) required for the algorithm to execute

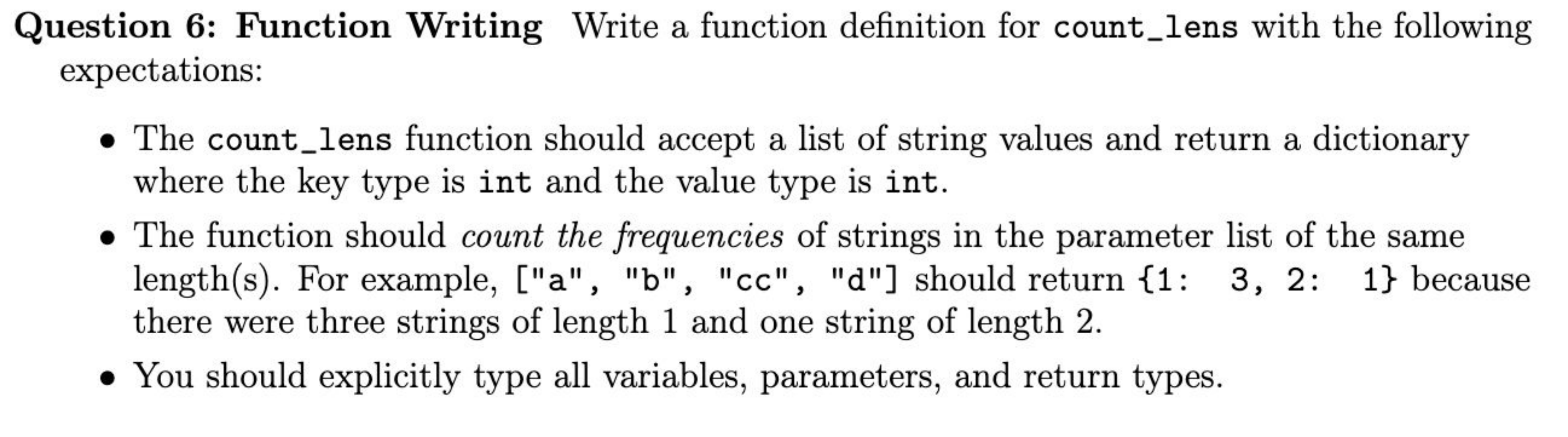
Average case:

- Average running time among several different inputs

Worst case (upper bound) ✨:

- The maximum running time given an input
  - How does the number of operations grow as an input grows?
- Important to understand how our algorithm will perform in the *worst* case
  - Prepare for the worst case. If an input ends up requiring fewer operations, great!!

Runtime analysis: order of growth



Question 6: Function Writing Write a function definition for `count_lens` with the following expectations:

- The `count_lens` function should accept a list of string values and return a dictionary where the key type is `int` and the value type is `int`.
- The function should *count the frequencies* of strings in the parameter list of the same length(s). For example, `["a", "b", "cc", "d"]` should return `{1: 3, 2: 1}` because there were three strings of length 1 and one string of length 2.
- You should explicitly type all variables, parameters, and return types.