CL22: Practice with Unit Tests

Announcements / Reminders

- **EX05** due Sunday (Oct 26) at 11:59pm
  - Writing functions to test the correctness of the functions you wrote for EX04!
- **Quiz 02** regrade requests are open till Tuesday (Oct 28) at 11:59pm
  - Questions about a particular question? Please visit us in Office Hours/Tutoring!
- **Quiz 03** is next Friday (Oct 31)
  - If you take your quizzes through ARS, double-check that you have scheduled to take it with them
  - If you will have a university-approved absence on this date and you'd like to take the quiz, email me!
  - Practice questions will be added to the site by Saturday
  - Hybrid Review Session on Thursday (Oct 30)

On Monday, we built this function in **`exercises/ex04/dictionary.py`**:

```
def bin_len(words: list[str]) -> dict[int, set[str]]:
    """Sort the elements of a list into a dict based on their lengths."""
    result: dict[int, set[str]] = {}
    for w in words:
        word_len: int = len(w)
        if word_len in result:
            result[word_len].add(w)
        else:
            result[word_len] = {w}
    return result
```

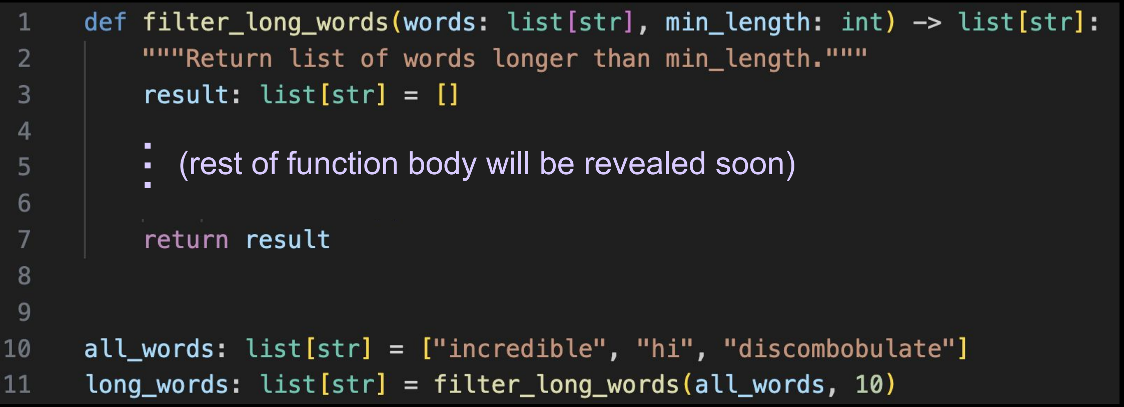
… and wrote some unit tests (accounting for use cases and edge cases) in **`exercises/ex04/dictionary_test.py`**

Testing For Desired Behavior

- We can also write unit tests that check that your function does what you want it to, rather than just checking that the return value is correct for a given function call
- Common example: checking whether your function *mutates* its input

Example in VSCode…

Consider the following skeleton for this function definition:



```python
def filter_long_words(words: list[str], min_length: int) -> list[str]:
    """Return list of words longer than min_length."""
    result: list[str] = []

    ⋮  (rest of function body will be revealed soon)

    return result


all_words: list[str] = ["incredible", "hi", "discombobulate"]
long_words: list[str] = filter_long_words(all_words, 10)
```

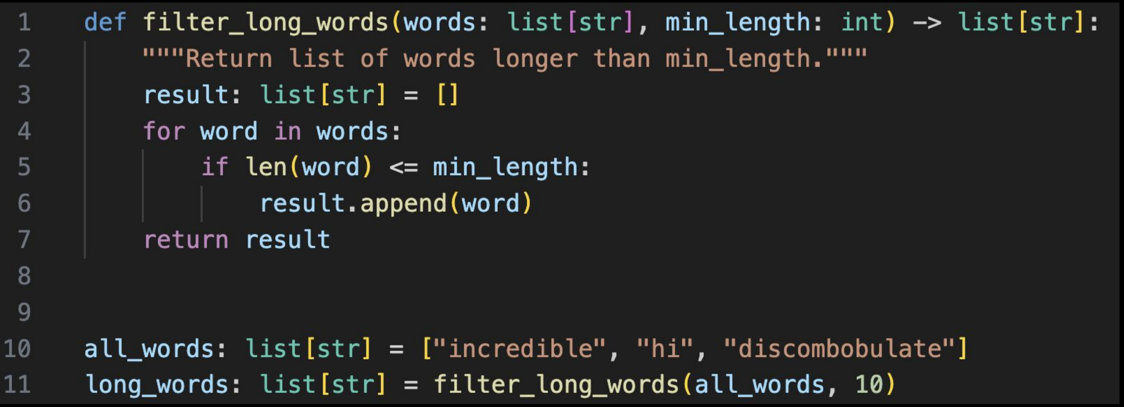
The function should:

- Take as input a **`list[str]`** named **`words`** and an **`int`** named **`min_length`**
- Return a **`list[str]`** of only the strings in **`words`** that are *longer* than the **`min_length`**
- Not mutate the **`words`** list

Write 3 function calls to filter_long_words:

- 2 that demonstrate typical use cases:
- 1 that demonstrates an edge case:

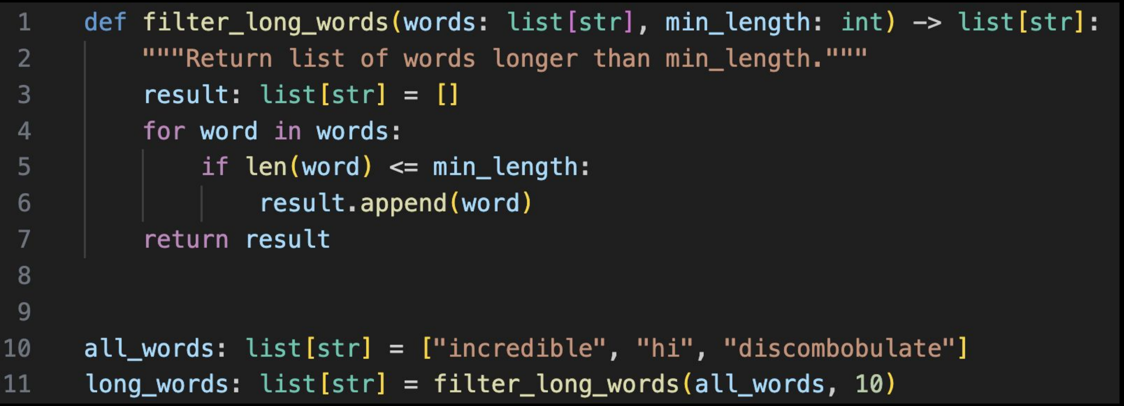
Complete a memory diagram for this code listing. If you wrote unit tests for your use and edge cases, would they all pass? Why or why not?



```python
def filter_long_words(words: list[str], min_length: int) -> list[str]:
    """Return list of words longer than min_length."""
    result: list[str] = []
    for word in words:
        if len(word) <= min_length:
            result.append(word)
    return result


all_words: list[str] = ["incredible", "hi", "discombobulate"]
long_words: list[str] = filter_long_words(all_words, 10)
```

Complete a memory diagram for this code listing. If you wrote unit tests for your use and edge cases, would they all pass? Why or why not?



```python
def filter_long_words(words: list[str], min_length: int) -> list[str]:
    """Return list of words longer than min_length."""
    result: list[str] = []
    for word in words:
        if len(word) <= min_length:
            result.append(word)
    return result


all_words: list[str] = ["incredible", "hi", "discombobulate"]
long_words: list[str] = filter_long_words(all_words, 10)
```

How could we alter the function body to fix the logic and get our unit tests to pass?