Homework 4

Fill in your name

In [86]:
```
# Fill in your name

first_name = 'Scott'
last_name = 'Urista'

assert len(first_name) != 0, "First name is blank"
assert len(last_name) != 0, "Last name is blank"
```

This assignment uses an idiom called Filtering

Filtering - running through a sequence of items, and testing each item with a Boolean function.

We filter your luggage at the airport. We look at each item in your suitcase, and stop you if you have a gun, explosives, or a bottle of water.

We use filtering to decide if we should let you board

We filter our reciepts after a trip. We can charge the meals and the room to the company, but can't charge the tickets to see Buddy Guy.

We filtering to find the sublist of items to pass on

Problem 1

is_vowel()

Is this character a vowel?

This function takes a string and returns a Boolean.

```
def is_vowel(ch: str) -> bool:
```

Fill in your function definition in the cell below.

In [87]:
```

def is_vowel(ch):
    "Is the character ch a vowel?"
    vowels='aeiouAEIOU'
    return ch in vowels
```

Test cases for is_vowel()

Run the cell below to call your function.

In [88]:
```
import string

def validate_vowel():
    for ch in 'aeiouAEIOU':
        assert is_vowel(ch), f"'{ch}' should be a vowel"

    for ch in string.ascii_lowercase:
        if ch not in 'aeiou':
            assert not is_vowel(ch), f"'{ch}' is not a vowel"
            assert not is_vowel(ch.upper()), f"'{ch.upper()}' is not a vow

    print('Success!')

validate_vowel()
```

```
Success!
```

Our Unit Test uses filtering

In this unit test, we run through a list of vowels looking for a counterexample.

Problem 2: has_all_vowels()

Does the string contain all the vowels?

This function takes a string and returns a Boolean.

```
def has_all_vowels(word: str) -> bool:
```

Fill in your function definition in the cell below.

In [89]:
```
def has_all_vowels(word):
    "Does word have all the vowels?"
    word_lower = word.lower()
    vowels = "aeiou"
    for ch in vowels:
        if ch not in word_lower:
            return False

    return True
```

Test cases for has_all_vowels()

Run the cell below to call your function.

In [90]:
```
def validate_all_vowels():
    assert has_all_vowels('vexatious'), "'vexatious' has all the vowels"
    assert has_all_vowels('VEXATIOUS'), "'VEXATIOUS' has all the vowels"

    # Now remove one vowel at a time
    # Each test case should return False
    assert not has_all_vowels('vxatious'), "Missing 'e'"
    assert not has_all_vowels('vextious'), "Missing 'a'"
    assert not has_all_vowels('vexatous'), "Missing 'i'"
    assert not has_all_vowels('vexatius'), "Missing 'o'"
    assert not has_all_vowels('vexatios'), "Missing 'u'"

    print('Success!')

validate_all_vowels()
```

```
Success!
```

Problem 3: is_all_vowels()

Is every character in word a vowel?

This function takes a string and returns a Boolean.

```
def is_all_vowels(word: str) -> bool:
```

Fill in your function definition in the cell below.

In [91]:
```
def is_all_vowels(word):
    "Is every character in word a vowel?"
    vowels = "aeiouAEIOU"
    for ch in word:
        if ch not in vowels:
            return False
    return True
```

Test case for is_all_vowels()

Run the cell below to call your function.

In [92]:
```
def validate_is_all_vowels():
    assert is_all_vowels('aie'), "'aie' is all vowels"
    assert is_all_vowels('Aie'), "'Aie' is all vowels"
    assert is_all_vowels('AeIoU'), "'AeIoU' is all vowels"

    assert not is_all_vowels('kaie'), "has a k"
    assert not is_all_vowels('akie'), "has a k"
    assert not is_all_vowels('aike'), "has a k"
    assert not is_all_vowels('aiek'), "has a k"

    print('Success!')

validate_is_all_vowels()
```

```
Success!
```

Problem 4: is_palindrome()

Write a function that decides if a word is a palindrome, like Radar or madam.

Is the word the same, read forwards or backwards?

Your function is_palindrome() should take a string and return a Boolean

```
def is_palindrome(word: str) -> bool:
```

Fill in your function definition in the cell below.

In [93]:
```
def is_palindrome(word):
    "Is the word a palindrome?"
    a = word.lower()
    b = a[::-1]
    return a == b
```

Test cases for is_palindrome

Run the cell below to call your function.

In [94]:
```
def validate_palindrome():
    lst = ['A', 'radar', 'Radar', 'Ada', 'madaM']
    for word in lst:
        assert is_palindrome(word), f"'{word}' is a palindrome"

    lst = ['adar', 'loop', 'leal']
    for word in lst:
        assert not is_palindrome(word), f"'{word}' is not a palindrome"


    print('Success!')

validate_palindrome()
```

```
Success!
```

Problem 5: Crossword Puzzle

Write a function that detect five letter words that start with 'a' and end in 't'.

Your function is_match() should take a string and return a Boolean

```
def is_match(word: str) -> bool:
```

Fill in your function definition in the cell below.

In [102]:
```
def is_match(word):
    "Is word a five letter string starting with 'a' and ending in 't'?"
    return len(word) == 5 and word[0] == 'a' and word[4] == 't'
```

Run the cell below to call your function.

In [103]:
```
def validate_match():
    for word in ['A', 'Radar', 'at', 'splat', 'adotp']:
        assert not is_match(word), f"'{word}' is not a match"

    for word in ['adopt', 'agent', 'argot', 'avast', 'adult']:
        assert is_match(word), f"'{word}' should be a match"

    print('Success!')

validate_match()
```

```
Success!
```

Problem 6: Find all matches in the Dictionary

Write a function that takes a Boolean function and a filename and prints the matches

Open the file words.txt, and print all five letter words that start with 'a' and end in 't'.

Your function find_match() should take a function and a filename and return a list of strings.

The function you pass in to find_match should take a string and return a Boolean.

We can use Python's Type Hints to describe the interface. This looks complicated, but this is a complicated idea.

```
f: Callable[[str], bool]
```

says: the parameter f is a function (Callable) that takes a string and returns a Boolean.

The return value is described by this:

```
-> List[str]:
```

which means "The function returns a list of strings".

The full description is:

```
from typing import List, Callable

def find_match(f: Callable[[str], bool], filename: str) -> Li
st[str]:
```

Fill in your function definition in the cell below.

In [104]:
```
# Write a function that uses your is_match() to filter the contents of wo

# Your function find_match takes a function: pass in your function is_matc
# find_match should return a list holding the words in the file words.txt

from typing import List, Callable

def find_match(f: Callable[[str], bool], filename: str) -> List[str]:
    "Returns a list of all lines matching the condition that function func
    filtered_words = []
    try:
        with open(filename) as words_file:
            for line in words_file:
                if f(line.rstrip()):
                    filtered_words.append(line.rstrip())

        return filtered_words

    except OSError:
        print("Houston we have a problem - no such file exists")
```

Run the cell below to call your function.
Be sure you have Downey's **words.txt** in the same directory as your notebook

In [105]:
```python
# Call your function to find the matches

lst = find_match(is_match, 'words.txt')

print(lst)
```

```
['abaft', 'abbot', 'abort', 'about', 'adapt', 'adept', 'admit', 'adopt',
'adult', 'adust', 'afoot', 'afrit', 'agent', 'agist', 'aglet', 'alant',
'alert', 'alist', 'allot', 'aloft', 'ambit', 'ament', 'amort', 'anent',
'angst', 'apart', 'aport', 'argot', 'arhat', 'armet', 'ascot', 'asset',
'atilt', 'audit', 'aught', 'avast', 'avert', 'await']
```

Problem 7: Find the palindromes in words.txt.

Use your function from Problem 6, but use is_palindrome(), the Boolean function you defined in problem 4, as the filter.

You should be able to use your work from problems 4 and 6 without changing them

Be sure you have Downey's **words.txt** in the same directory as your notebook.
Print the number of palindromes in the file.

Print the first 10 palindromes in the file.

In [106]:
```python
# Write a Python fragment that uses your work from problems 2 and 4

print(len(find_match(is_palindrome, 'words.txt')))
print(find_match(is_palindrome, 'words.txt')[0:10])
```

```
91
['aa', 'aba', 'aga', 'aha', 'ala', 'alula', 'ama', 'ana', 'anna', 'ava']
```

Post Mortem

How long did it take you to solve this problem set?

Did anything confuse you or cause difficulty?

```
Enter your thoughts
OK, the problems this week felt a *lot* easier than last week's turtles,
to the point where I feel like I must be missing something. I spent a
bit of time reading up on try/except and how to handle the error message
when the file doesn't exist. If I understand correctly, we don't need to
explicitly close the file when we use with open(filename)?
```

In [ ]:
```

```