Homework 7

Fill in your name

In [1]:
```
first_name = "Shariq"
last_name = "Jamil"

assert(len(first_name) != 0)
assert(len(last_name)  != 0)
```

Problem 1: List of Squares

The List Comprehension below takes an integer N and returns the squares of the first N integers.

Rewrite the function without using List Comprehensions

```python
   def squares(N):
       return [i * i for i in range(N+1)]
```

In [14]:
```
# Takes an integer and returns a list of the squares up to N*N
# squares(4) would return [0, 1, 4, 9, 16]
def squares(N):
    list_to_return = []
    for i in range(N+1):
        # add square of i to return list
        list_to_return.append(i * i)
    return list_to_return
```

Unit Tests for Squares

In [15]:
```
def test_squares():
    assert(squares(1) == [0, 1])
    assert(squares(4) == [0, 1, 4, 9, 16])
    assert(squares(9) == [0, 1, 4, 9, 16, 25, 36, 49, 64, 81])

    return 'Pass'

test_squares()
```

Out[15]:
```
'Pass'
```

Problem 2: Whozit

The List Comprehension below takes an integer and returns a list.

Rewrite Whozit without using a List Comprehension

```
def whozit(n):
        return [i for i in range(n) if '3' in str(i)]
```

In [20]:
```
'''
This function takes in a number and returns
a list containing any numbers between 0 and that number
that contain the number '3'
'''
def whozit(n):
    list_to_return = []
    # iterate until n
    for i in range(n):
        # if substring '3' is found in i
        if '3' in str(i):
            # add number to return list
            list_to_return.append(i)
    return list_to_return
```

Unit tests for Whozit

In [21]:
```
def test_whozit():
    assert whozit(20) == [3, 13]
    assert whozit(30) == [3, 13, 23]
    assert whozit(40) == [3, 13, 23, 30, 31, 32, 33, 34, 35, 36, 37, 3
8, 39]

    print("Success")

test_whozit()
```

```
Success
```

Problem 3: Loop Garou

Rewrite loop_garou as a List Comprehension, and describe what it does

```
def loop_garou(s):
        res = []
        for ch in s:
            if ch in 'aeiou':
                res.append(ch)
        return '-'.join(res)
```

In [48]:
```
'''
This function takes in a string and returns a string of
vowels found in the string concatenated by '-'
'''
def loop_garou(s):
    return '-'.join([ch for ch in s if ch in 'aeiou'])
```

Out[48]:
```
'a-e-i-o-u'
```

Unit tests for Loop Garou

In [50]:
```
def test_loop_garou():
    assert loop_garou('We are his sisters') == 'e-a-e-i-i-e'
    assert loop_garou('his cousins and his aunts') == 'i-o-u-i-a-i-a-u'

    print('Success')

test_loop_garou()
```

```
Success
```

Problem 4: Common Elements

Write a function common that takes two lists, and return a list of the items that appear in both lists, in the order they appear in the first list.

Given [1, 3, 2, 5, 6, 0, 7] and [7, 4, 3, 2, 1], you should return [1, 3, 2, 7]

In [58]:
```
def common(lst1, lst2):
    # returns an array of items that exist in
    # lst1 and lst2 in the order of lst1
    return [i for i in lst1 if i in lst2]
```

Unit Tests for Common

In [59]:
```
def test_common():
    assert common([1, 3, 2, 5, 6, 0, 7], [7, 4, 3, 2, 1]) == [1, 3, 2,
7]
    assert common([1, 2, 3, 4, 5], [7, 5, 3, 1]) == [1, 3, 5]
    assert common([7, 5, 3, 1], [1, 2, 3, 4, 5]) == [5, 3, 1]
    assert common([3*i for i in range(7)], [2*i for i in range(10)]) ==
[0, 6, 12, 18]

    print('Success')

test_common()
```

```
Success
```

Problem 5: Missouri Wine and Beer

Find the top beer and wine suppliers to the state of Missouri.

The CVS file Missouri_Beer_Wine.cvs lists suppliers to the state of Missouri, which runs package stores. Each line records a different supplier, and includes a 5 or 9 digit zip code. Find the 5-digit zip codes that hold the most suppliers to the state. When you see a 9-digit zip code, truncate the last 4 digits.

Your function should return a list of lists, with the frequency and the zip code. Organize the list in decreasing order of frequency.

Here are three items from my list of 720 zip codes

[... [9, '65616'], [8, '94573'], [8, '63103'] ...]

This tells me that Branson, Missouri, has 9 beer or wine suppliers to the state, and Rutherford, California, has 8.

Print the number of suppliers and the zip code for the 10 most common zip codes in the file.

Use the csv library to read the textfile.
Use the idiom Dictionary as a Counter (section 11.1 of Downey) to count the number of times you see each zip code. Traverse the Dictionary and build a list of lists, and use sort() or sorted() to organize your list.

To validate your results, you can check the three zip codes above, and you can use Google to map the zip code to a location and check that that is a likely source for wine or beer.

In [99]:
```
# Put your implementation there
import csv

def list_cities(filename):
    with open('Missouri_Beer_wine.csv') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        suppliers = dict()
        first_line = True

        for supplier in csv_reader:
            #skip first line
            if first_line:
                first_line = False
                continue
            # store ref to zip code
            zip = supplier[1]
            # if 9 digit zip, truncate to digits
            if len(zip) > 5:
                zip = zip[0:5]
            # if already added to dict, increment frequency
            if zip in suppliers:
                suppliers[zip] += 1
            # if not in dict, add with frequency of 1
            else:
                suppliers[zip] = 1

        # traverse dictionary add create list in prescribed format of
[[freq, zip],...]
        supplier_list = [[suppliers[i],i] for i in suppliers]
        # sort in descending order
        supplier_list.sort(reverse = True)
        return supplier_list
```

In [100]:
```
lst = list_cities("Missouri_Beer__Wine.csv")

print(len(lst))
print(lst[:10])
```

```
720
[[54, '94558'], [41, '94574'], [27, '95448'], [18, '95476'], [18, '94
515'], [16, '95472'], [16, '93446'], [12, '95403'], [11, '99362'], [1
1, '94562']]
```

Post Mortem

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

Did anything confuse you or cause difficulty?

In [ ]:
```
# Enter your thoughts
# It took me about five hours to finish this homework.
# Nothing confusing but I have a decent understanding of list comprehen
sion now. Very cool feature.
```