Homework 6: Fall 2020

Fill in your name

In [132]:
```
first_name = "Scott"
last_name = "Urista"

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

Problem 1: Mailman

Turn an e-mail address into a list of components

We address letters and and e-mail backwards. When the post office gets a letter, they need to read from the bottom up to decide where to send it next

```
Stephen Dedalus
Class of Elements
Clongowes Wood College
Sallins
Country Kildare
Ireland
```

Internet addresses such as 'jparker@word.std.com (mailto:jparker@word.std.com)' work the same way.

Write a function that takes a string holding an e-mail address and returns a list with two items: the username, followed by a list of the steps we will need to take to route the mail. In the case above, you would return

```
['jparker', ['com', 'std', 'world']]
```

Hint: Use the string method split() twice.

In [133]:
```
# Takes a string and returns a list
def parse_email_address(s):
    "split a mail address into recipient and list of hops"
    split_email = s.split("@")
    steps_list = split_email[1].split(".")
    split_email[1] = steps_list[::-1]  # need to reverse order!

    return split_email
```

Test cases for Mailman

In [134]:
```
def mailman_test():
    assert(parse_email_address('jdp@world.std.com') == ['jdp', ['com', 's

    return('Pass')

mailman_test()
```

Out[134]:
```
'Pass'
```

Problem 2: Parentheses

Decide if a string contains valid nested parentheses

You are given a string consisting only of parentheses - (,), {, }, [, and]. Write a Boolean function is_valid_parens() that takes a string and decides if it consists of valid nested parenthesis.

Hint: Your function should take open parentheses, such as '(', and 'push it on a stack' and should take closing parentheses, and pop the stack and compare. If the close parenthesis doesn't match the open parenthesis on top of the stack, the string is invalid. If the stack is empty too soon, or is not empty when you finish the string, the string is invalid.

You can read about stacks here:

https://en.wikipedia.org/wiki/Stack_(abstract_data_type) (https://en.wikipedia.org/wiki/Stack_(abstract_data_type))

Implement your stack with a list, pushing and poping the final element.

In [135]:
```
# Takes a string, and returns a Boolean
#  '{()[{}]}' is valid:    return True
#  '{()[{}}' is not:       return False
def is_valid_parens(s):
    "Is this a well-nested set of parentheses?"
    stack = []
    parens_dict = {"(":")", "[":"]", "{":"}"}

    for parens in s:
        if parens in parens_dict:
            stack.append(parens)
        elif len(stack) == 0 or parens_dict[stack.pop()] != parens:
            return False

    return len(stack) == 0
```

Test case for is_valid_parens()

In [136]:
```
def test_parens():
    assert(is_valid_parens(""))
    assert(is_valid_parens("[]"))
    assert(is_valid_parens('{()[{}]}'))
    assert(is_valid_parens("{}"))
    assert(is_valid_parens("{[]}"))
    assert(is_valid_parens("{}[]"))
    assert(is_valid_parens("([{}({}[])])"))

    assert not is_valid_parens('{()[{}}]'), 'Interlaced parentheses'
    assert not is_valid_parens("[["), "Unmatched opens"
    assert not is_valid_parens("}{"), "Unmatched close"
    assert not is_valid_parens("{]"), "Missmatched parentheses"
    assert not is_valid_parens("{[])"), "Missmatched parentheses"
    assert not is_valid_parens("{[)][]}"), "Missmatched parentheses"
    assert not is_valid_parens("([{])"), "Missmatched parentheses"
    assert not is_valid_parens("[({]})"), "Missmatched parentheses"

    return 'Pass'

test_parens()
```

Out[136]:
```
'Pass'
```

Problem 3: Solitary Words

A word is *solitary* if no letter appears more than once.

Thus 'once' is solitary. 'Solitary' is solitary. 'Pop' is not solitary, as there are two p's.

Write a function that takes a string and returns a Boolean telling us if the string is solitary.

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

Hint:

Review Downey's discussion of Dictionary as a Collection of Counters.

Hint:

You can also use a Set to keep track of the letters in a word

Hint:

You can solve this without Sets or Dictionaries. Please don't. This is the only problem on this set that uses these new ideas.

In [138]:
```python
def solitary(word):
    """Take a string, and decide if any letter appears twice"""
    solitary_dict = {}
    for char in word:
        if char.lower() in solitary_dict:
            return False
        else:
            solitary_dict[char.lower()] = None

    return True
```

Unit Test cases for solitary()

In [139]:
```python
def test_solitary():
    assert solitary('abcd')
    assert not solitary('aa'), "Two a's"
    assert solitary('solitary')
    assert not solitary('Pop'), "Two p's"
    assert not solitary("eleven"), "Three e's"
    assert solitary("subdermatoglyphic")

    print('Success!')

test_solitary()
```

```
Success!
```

Problem 4: Find Large Files

Write a function that takes a directory and a size in bytes, and returns a list of files in the directory or below that are larger than the size.

For example, you can use this function to look for files larger than 1 Meg below your Home directory.

You will find a Python function that gives you the size of a file in the os.path library:

[https://pymotw.com/3/os.path/ (https://pymotw.com/3/os.path/)](https://pymotw.com/3/os.path/)

In [140]:
```
import os

def find_large_files(dirname, filesize):
    "Return a list of large files below this point"
    large_files_list = []
    files = next(os.walk(dirname))[2]
    for file in files:
        if os.stat(dirname+"/"+file).st_size > filesize:
            large_files_list.append(file)

    return large_files_list



def find_large_files_dict(dirname, filesize):
    "Same program as above, but using dictionary for storing file name and
    large_files_dict = {}
    files = next(os.walk(dirname))[2]
    for file in files:
        if os.stat(dirname+"/"+file).st_size > filesize:
            large_files_dict[file]=os.stat(dirname+"/"+file).st_size

    return large_files_dict

find_large_files_dict("..", 0)
```

Out[140]:
```
{'homework5_SUrista.ipynb': 20443,
 'Lecture05.pdf': 11347127,
 'homework5_SUrista.pdf': 455848,
 'Lecture04.pdf': 14939568,
 'walk.py': 442,
 'words.txt': 1130523,
 'Lecture06.pdf': 16452709}
```

Show your program in action

Give the parameters and show the results for your program

I looked for files larger than a Megabyte found below the directory one step up.

In [141]:
```
lst = find_large_files("..", 1048576)
print(len(lst))

for path in lst:
    print(path)
```

```
4
Lecture05.pdf
Lecture04.pdf
words.txt
Lecture06.pdf
```

Problem 5:

The following stand-alone program takes a url from the command line, reads the contents of a webpage, and prints it.

Modify the program to take a filename as a second parameter and save the contents of the webpage in a text file.

```
python save_url.py 'http://www.python.org/' pythonpage.txt
```

would save the contents of the webpage in the text file pythonpage.txt.

You may want to review the mycopy.py program from day 4 which takes two parameters and copies the contents of the first file to the second.

Use this and an editor to find the copyright notice on the following websites.

```
website = 'http://www.python.org/'
website = 'https://www.extension.harvard.edu'
website = 'http://en.wikipedia.org/wiki/Python'

website = Your piazza link: mine looks something like this:
        https://piazza.com/class/myxlplyxmyxlplyx?cid=194
```

You will need to remove the last bit from your piazza link that specifies the cid: '?cid=194'. In my case, this would leave https://piazza.com/class/myxlplyxmyxlplyx (https://piazza.com/class/myxlplyxmyxlplyx)

This problem gives you a chance to examine webpages, and shows how different website creators deal with a common problem, presenting a copyright. You will see that writing a program to extract the copyright from different websites would be difficult.

One alternative way to view the source for a website is through your browser. For example, in Chrome you can use View/Developer/View Source

In [142]:
```python
# read_url.py
#
# Jeff Parker
#
# Usage:
#     python read_url.py <website>

import urllib.request
import sys


def fetch_contents(website):
    "Return the contents of this webpage as a list of lines"
    try:
        res = []

        with urllib.request.urlopen(website) as f:
            text = f.read().decode('utf-8')

            # Break the page into lines
            text = text.split('\n')
            for line in text:
                res.append(line)

        return res

    except urllib.error.URLError as e:
        print(e.reason)
        return []

if (len(sys.argv) != 2):
    print(f"Usage: python read_url.py <website>")
else:
    lst = fetch_contents(sys.argv[1])

    # Now display the contents
    for line in lst:
        print(line)
```

```
Usage: python read_url.py <website>
```

Include your program below

In [ ]:

```
# save_url.py
# S. Urista / 4 Oct 2020
# Usage:
#     python save_url.py <website> <textfile>

import urllib.request
import sys

def fetch_contents(website, filename):
    "Saves the contents of a webpage to a file"
    try:
        res = []

        with urllib.request.urlopen(website) as f:
            text = f.read().decode('utf-8')

            # Break the page into lines
            text = text.split('\n')
            for line in text:
                res.append(line)

        # print contents to file
        with open(filename, 'w') as f:
            for line in text:
                print(line, file=f)

        return res

    except urllib.error.URLError as e:
        print(e.reason)
        return []


if (len(sys.argv) != 3):
    print(f"Usage: python read_url.py <website> <filename>")
else:
    fetch_contents(sys.argv[1], sys.argv[2])
```

Show the webpage elements holding the copyright information for each website

In [ ]:

```
### Copyright notice for 'http://www.python.org/'
<div class="copyright">
    <p><small>
        <span class="pre">Copyright &copy;2001-2020.</span>
            &nbsp;<span class="pre"><a href="/psf-landing/">Python Softwa
            &nbsp;<span class="pre"><a href="/about/legal/">Legal Statemen
            &nbsp;<span class="pre"><a href="/privacy/">Privacy Policy</a
            &nbsp;<span class="pre"><a href="/psf/sponsorship/sponsors/#he
    </small></p>
</div>
```

In [ ]:

```
### Copyright notice for 'https://www.extension.harvard.edu'
  <div class="pane-content longform">
    <p>Copyright &copy;2020 President and Fellows of Harvard College</p>
  </div>
```

In [ ]:

```
### Copyright notice for 'http://en.wikipedia.org/wiki/Python'
<li id="footer-copyrightico">
    <a href="https://wikimediafoundation.org/">
    <img src="/static/images/footer/wikimedia-button.png"
    srcset="/static/images/footer/wikimedia-button-1.5x.png 1.5x,
    /static/images/footer/wikimedia-button-2x.png 2x" width="88" height="3
    loading="lazy" /> </a>
</li>
```

In [ ]:

```
### Copyright notice for Piazza
<li><a class="log_click" data-log-click="footer_legal_copyright" href="/le
```

Post Mortem

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

Did anything confuse you or cause difficulty?

In [131]:

```
# Enter your thoughts
# Took about 2 hours, of which 1hr45mn spent on os.stat.
# While I see the utility in using Notebooks for the homework exercises, I
# Find Large Files has us return a list; could have done this by just retu
# I added a second function that returns a dictionary instead of a list, a
# (key) and file size (value), just so it's clear that the find_large_file
# showing files larger than filesize
```