Homework 6: Fall 2020

Fill in your name

In [ ]:
```
first_name = ""
last_name = ""

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 [ ]:
```
# Takes a string and returns a list
def parse_email_address(s):
    "split a mail address into recipient and list of hops"
    pass
```

In [ ]:
```
# Takes a string and returns a list
def parse_email_address(s):
    "split a mail address into recipient and list of hops"
    res = s.split('@')
    location = res[1].split('.')
    res[1] = location[::-1]
    return res

parse_email_address('jdp@world.std.com')
```

Test cases for Mailman

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

    return('Pass')

mailman_test()
```

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 [ ]:
```
# 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?"
    pass
```

In [ ]:
```
def is_valid_parens(s):
    "Is this a well-nested set of parentheses?"
    stack = []
    pairs = {'[':']', '{':'}', '(':')'}   # a dictionary for things th

    # loop through each character in the string
    for ch in s:
        # Is this an open paren?
        if ch in pairs:
            stack.append(ch)
        else:
            # Close paren
            try:
                cur = stack.pop()
            except IndexError:
                return False

            # Does the pair match?
            if pairs[cur] != ch:
                return False

    # If the list is empty, we have a valid expression
    return len(stack) == 0
```

Test case for is_valid_parens()

In [ ]:
```
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()
```

Problem 3: Solitary

While this is a classic program that has several elegant solutions, the solutions are too easy to find. I have decided to replace the problem with the following:

Problem 3: Secret code

We can devise a secret code that maps 'one' to 'two'. We send 'o' to 't', 'n' to 'w' and 'e' to 'o'.

We cannot find any code that sends 'two' to 'three', as the words have different length.

We cannot find a code that sends 'foo' to 'bar', as we would need 'o' to represent 'a' and 'r'.

Likewise we cannot send 'four' to 'aaaa', as there would be no way to map the letters back.

Write a Boolean function secret_code() that decides if we can find a code that sends one word to another and back again.

```
def secret_code(word1: str, word2: str) -> bool:
```

In [ ]:
```
def secret_code(word1, word2):
    """Can we map word1` to word2?"""
    pass
```

In [8]:
```
def secret_code(s1, s2):
    """Can we map s1` to s2?"""
    if len(s1) != len(s2):
        return False

    mapping = {}   # Mapping
    seen = set()   # charaacters we have seen
    for pos in range(len(s1)):
        ch1 = s1[pos]
        ch2 = s2[pos]
        if ch1 in mapping:
            if mapping[ch1] != ch2:
                return False
        elif ch2 in seen:
            return False
        else:
            mapping[ch1] = ch2
            seen.add(ch2)

    return True
```

Unit Test cases for secret code

In [9]:
```
def test_secret():
    assert secret_code('one', 'two'), "Can map"
    assert not secret_code('one', 'four'), "Different lengths"
    assert not secret_code('one', 'aaa'), "No way back"
    assert not secret_code('sheep', 'black'), "Can't decide where 'e'

    print('Success!')

test_secret()
```

```
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/)

In [ ]:
```
import os

def find_large_files(dirname, filesize):
    "Return a list of large files below this point"
    pass
```

In [11]:
```
import os

def find_large_files(dirname, filesize):
    "Return a list of large files below this point"

    res = []

    # Walk over the files in this directory
    for name in os.listdir(dirname):

        # Construct a full path
        path = os.path.join(dirname, name)

        # print filenames, and traverse directories
        if os.path.isfile(path):
            if os.path.getsize(path) > filesize:          # Check file
                res.append(path)
        else:
            res = res + find_large_files(path, filesize)

    return res
```

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 [12]:
```
lst = find_large_files('..', 1048576)
print(len(lst))

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

```
1
../Assignment2.webarchive
```

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.

For example, here is the copyright notice for the New York Times, https://www.nytimes.com (https://www.nytimes.com). I have introduced whitespace to help visualize the element.

```
<li data-testid="copyright">
    <a class="css-jq1cx6" href="https://help.nytimes.com/hc/en
-us/articles/115014792127-Copyright-notice">©
        <span>2020</span>
        <span>The New York Times Company</span>
    </a>
</li>
```

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 [ ]:
```
# 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)
```

Include your program below

In [ ]:
```
# save_url.py
#
# Usage:
#     python save_url.py <website> <textfile>

import urllib.request
import sys
```

In [ ]:
```
# save_url.py
#
# Jeff Parker
#
# Usage:
#     python save_utl.py <website> <output file>

import urllib.request
import sys

def fetch_contents(website):
    try:
        res = []

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

            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) < 3):
    print(f"Usage: python save_url <website> <output file>")
else:
    lst = fetch_contents(sys.argv[1])
    dest = sys.argv[2]

    try:
        with open(dest, 'w') as fout:
            for line in lst:
                fout.write(line)
    except:
        print(f"Could not open {dest}")
```

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>
         <span class="pre"><a href="/psf-landing/">Python Softwar
         <span class="pre"><a href="/about/legal/">Legal Statemer
         <span class="pre"><a href="/privacy/">Privacy Policy</a>
         <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</
</div>
```

In [ ]:
```
### Copyright notice for 'http://en.wikipedia.org/wiki/Python'

<li id="footer-info-copyright">Text is available under the <a rel="lic
additional terms may apply.  By using this site, you agree to the <a h


<li id="footer-copyrightico"><a href="https://wikimediafoundation.org/
```

Post Mortem

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

Did anything confuse you or cause difficulty?

In [ ]:
```
# Enter your thoughts
```