PythonExercises

January 15, 2024

1 Python Exercises

In this assignment, you will implement simple functions that work with lists, strings, and text files. My solutions are between one and seven lines of code. The problem marked as 5111 is mandatory for MS students, UG students who solve it correctly get bonus points.

1.1 Write Your Name Here:

2 Submission instructions

1. Click the Save button at the top of the Jupyter Notebook.
2. Please make sure to have entered your name above.
3. Select Cell -> All Output -> Clear. This will clear all the outputs from all cells (but will keep the content of ll cells).
4. Select Cell -> Run All. This will run all the cells in order, and will take several minutes.
5. Once you've rerun everything, select File -> Download as -> PDF via LaTeX and download a PDF version showing the code and the output of all cells, and save it in the same folder that contains the notebook file.
6. Look at the PDF file and make sure all your solutions are there, displayed correctly.
7. Submit **both** your PDF and the notebook file .ipynb on Canvas.
8. Make sure your your Canvas submission contains the correct files by downloading it after posting it on Canvas.

2.1 Working with sequences

2.1.1 Simple sum (10p)

Write a function mysum(l) that takes as input a list l and outputs the sum of all the elements in the list. Do not use the predefined sum() function, implement it yourself.

[ ]:
```python
def mysum(l):
    #YOUR CODE HERE




    return result

# This call should return 45
```

```
mysum(range(10))
```

2.1.2 Complex sum (10p)

Write a function sum_list(l) that takes as input a list l and outputs another list q of the same length as l and that contains at position j the sum of all the elements from positions 0, 1, ...j, meaning q[j] = l[0] + l[1] + ... + l[j].

[ ]:
```
def sum_list(l):
    q = []

    # YOUR CODE HERE





    return q

# This call should return [1, 6, 4, 7]
sum_list([1, 5, -2, 3])
```

2.1.3 Generic function (10p)

Does your function sum_list(l) work with strings too? If not, change it so that the function works with **both** numbers and strings.

[ ]:
```
# This call should return ['u', 'un', 'unc', 'uncc']
print(sum_list(['u', 'n', 'c', 'c']))

# This call should return [1, 6, 4, 7]
print(sum_list([1, 5, -2, 3]))
```

2.1.4 [5111] Brute force sorting (20p)

Define a function force_sort(l) that takes as input a list of integers l and outputs a list of the same integers in sorted order. The function should use a brute-force sorting algorithm, meaning that it generates permutations of the initial list until it finds a permutation that is sorted. You should write code to generate permutations yourself (do not use library functions for that). Write a separate function check_sorted(l) that returns True if and only if the input list is sorted.

[ ]:
```
def check_sorted(l):
    # YOUR CODE HERE






    return True
```

```
def force_sort(l):
    # YOUR CODE HERE






    return []

# This call should return [-1, 3, 5, 9, 13].
print(force_sort([9, 13, -1, 5, 3]))

# This call should return False.
print(check_sorted([-1, 3, 13, 5, 9]))
```

2.1.5 Bonus points: One liner (10p)

Define a function sum_elements(l1, l2) that adds the two lists given as arguments, element-wise. Assume the two lists have the same length. You should do this with only one line of code.

Hint: use list comprehensions.

```
[ ]: def sum_elements(l1, l2):
         # YOUR CODE HERE
         return []

     # This call should return [3, 5, 7, 9].
     sum_elements([1, 2, 3, 4], [2, 3, 4, 5])
```

2.2 Working with strings

2.2.1 Lines and tokens (10p)

Write a function stats(s) that returns in a tuple the number of lines and the number of tokens in an input string s. For this exercise, we consider that a token is any maximal string that does not contain white spaces such as ' ', tabs, or newlines.

Hint: you can use the string methods splitlines() and split().

```
[ ]: def stats(s):
         clines, ctokens = 0, 0

         # YOUR CODE HERE
```

```
    return clines, ctokens

# Note how strings can be written on multiple lines in Python code.
s = 'There\'s a passage in the Principia Discordia where Malaclypse complains␣
 ↪to the Goddess ' \
    'about the evils of human society. "Everyone is hurting each other, the␣
 ↪planet is rampant ' \
    'with injustices, whole societies plunder groups of their own people,␣
 ↪mothers imprison sons, ' \
    'children perish while brothers war."\n' \
    '\n' \
    'The Goddess answers: "What is the matter with that, if it\'s what you want␣
 ↪to do?"\n' \
    '\n' \
    'Malaclypse: "But nobody wants it! Everybody hates it!"\n' \
    '\n' \
    'Goddess: "Oh. Well, then stop."\n' \
    'https://slatestarcodex.com/2014/07/30/meditations-on-moloch/'

# This will display the string value.
print(s)

# This should display 8 lines and 76 tokens.
lines, tokens = stats(s)
print()
print('There are', lines, 'lines and', tokens, 'tokens.')
```

2.2.2 Character substitutions (10p)

Write a function letterize(s) that takes as input a string s and returns another string that is a copy of s where all non-alphabet characters are replaced with the whitespace character ' '.

Hint: you can use the string methods isalpha().

[ ]:
```
def letterize(s):
    result = ''

    # YOUR CODE HERE




    return result

r = letterize(s)
print(r)

# The length of the result should be 537.
```

```
print(len(r))
```

2.2.3 Token substitutions (10p)

Write a function substitute(text, source, target) that replaces each occurrence of the source string in text with the target string and returns the result. You are not allowed to use the `replace()` function, implement this yourself, e.g. using `find()`.

[ ]:
```
def substitute(text, source, target):
    result = ''

    # YOUR CODE HERE






    return result

# Note how strings can be written on multiple lines in Python code.
text = 'There\'s a passage in the Principia Discordia where Malaclypse␣
 ↪complains to the Goddess ' \
        'about the evils of human society. "Everyone is hurting each other, the␣
 ↪planet is rampant ' \
        'with injustices, whole societies plunder groups of their own people,␣
 ↪mothers imprison sons, ' \
        'children perish while brothers war."\n' \
        '\n' \
        'The Goddess answers: "What is the matter with that, if it\'s what you␣
 ↪want to do?"\n' \
        '\n' \
        'Malaclypse: "But nobody wants it! Everybody hates it!"\n' \
        '\n' \
        'Goddess: "Oh. Well, then stop."\n'
print(substitute(text, 'Malaclypse', 'Mazikeen'))
```

2.2.4 Swapping tokens (10p)

Write a function swap(text, source, target) that replaces each occurrence of the source string in text with the target string and each target string with the source string.

[ ]:
```
def swap(text, source, target):
    result = ''

    # YOUR CODE HERE
    return result

# Note how strings can be written on multiple lines in Python code.
text = 'There\'s a passage in the Principia Discordia where Malaclypse␣
 ↪complains to the Goddess ' \
        'about the evils of human society. "Everyone is hurting each other, the␣
 ↪planet is rampant ' \
        'with injustices, whole societies plunder groups of their own people,␣
 ↪mothers imprison sons, ' \
        'children perish while brothers war."\n' \
        '\n' \
        'The Goddess answers: "What is the matter with that, if it\'s what you␣
 ↪want to do?"\n' \
        '\n' \
        'Malaclypse: "But nobody wants it! Everybody hates it!"\n' \
        '\n' \
        'Goddess: "Oh. Well, then stop."\n'
print(swap(text, 'Malaclypse', 'Goddess'))
```

2.3 Working with text files

2.3.1 Lines and paragraphs (20p)

Write a function text_stats(fname) that read a text file **line by line** and returns a tuple containing the following elements:

1. The total number of *lines* in the file.
2. The total number of *text lines* in the file.

- A *text line* is defined as a line that contains at least one non white space character. For example, a line that contains two white spaces followed by a tab character is not considered a text line.
- An *empty line* is a line that is not a text line.

3. The total number of *text paragraphs* in the file.

- A *text paragraph* is a maximal sequence of text lines. Thus, a text paragraph (a) must be preceded by an empty line or the beginning of the file, (b) must be followed by an empty line or the end of the file, and (c) must not contain any empty lines.

[ ]:
```
def text_stats(fname):
    # YOUR CODE HERE




    return 0, 0, 0
```

```
# This call should return a tuple that specifies 6 paragraphs.
print(text_stats('../data/colorless.txt'))
```

2.4 Bonus points (10p)

Write a function flatten(l) that takes as input a *deep* list that may contain other lists that may contain other lists ... and returns a *shallow* list that contains just the atmoic elements of the original list.

[ ]:
```
def flatten(l):
    result = []

    # YOUR CODE HERE







    return result

l = [1, [2, 3], [4, [5, 6, [7, [8, 9]]], 10], 11, 12]

# This call should print [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
print(flatten(l))
```