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.

Manish Kumar Govind :

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.

Working with sequences

Simple sum

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.

In [18]:
```
def mysum(l):
    #YOUR CODE HERE
    result =0
    for i in l :
        result+=i

    return result

# This call should return 45
mysum(range(10))
```

Out[18]:
```
45
```

Complex sum

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]`.

In [19]:
```
def sum_list(l):
    q = []
    res = l[0]
    # YOUR CODE HERE
    for i in range(1,len(l)):
        q.append(res)
        res+=l[i]

    q.append(res)

    return q

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

Out[19]:
```
[1, 6, 4, 7]
```

Generic function

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

In [20]:
```
# 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]))
```

```
['u', 'un', 'unc', 'uncc']
[1, 6, 4, 7]
```

Bonus points: One liner

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.

In [21]:
```
def sum_elements(l1, l2):
    # YOUR CODE HERE
    return [x+y  for x,y in zip(l1,l2)]

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

Out[21]:
```
[3, 5, 7, 9]
```

Working with strings

Lines and tokens

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()`.

In [22]:
```
def stats(s):
    clines, ctokens = 0, 0

    # YOUR CODE HERE
    clines = len(s.splitlines())
    ctokens = len(s.split())

    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 God
    'about the evils of human society. "Everyone is hurting each other, the planet is ra
    'with injustices, whole societies plunder groups of their own people, mothers impris
    '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' \
    '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.')
```

```
There's a passage in the Principia Discordia where Malaclypse complains to the Goddess a
bout 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 so
ns, children perish while brothers war."

The Goddess answers: "What is the matter with that, if it's what you want to do?"

Malaclypse: "But nobody wants it! Everybody hates it!"

Goddess: "Oh. Well, then stop."
https://slatestarcodex.com/2014/07/30/meditations-on-moloch/

There are 8 lines and 76 tokens.
```

Character substitutions

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()`.

In [23]:
```
def letterize(s):
    result = ''

    for i in s :
        if not i.isalpha():
            result+=' '
        else :
            result+=i
    # YOUR CODE HERE
    return result
```

```
r = letterize(s)
print(r)

# The length of the result should be 537.
print(len(r))
```

```
There s a passage in the Principia Discordia where Malaclypse complains to the Goddess a
bout 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 so
ns  children perish while brothers war  The Goddess answers  What is the matter with
that  if it s what you want to do  Malaclypse  But nobody wants it  Everybody hates i
t  Goddess  Oh  Well  then stop  https  slatestarcodex com  meditations o
n moloch
537
```

Token substitutions

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.

In [24]:
```
def substitute(text, source, target):
    result = ''

    # YOUR CODE HERE

    i=0
    while i < len(text)  :
        if(text[i:i+len(source)] == source):
            result += target
            i=i+len(source)
        else :
            result += text[i]
            i = i+1

    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
        'about the evils of human society. "Everyone is hurting each other, the planet i
        'with injustices, whole societies plunder groups of their own people, mothers im
        'children perish while brothers war."\n' \
        '\n' \
        'The Goddess answers: "What is the matter with that, if it\'s what you want to d
        '\n' \
        'Malaclypse: "But nobody wants it! Everybody hates it!"\n' \
        '\n' \
        'Goddess: "Oh. Well, then stop."\n'
print(substitute(text, 'Malaclypse', 'Mazikeen'))
```

```
There's a passage in the Principia Discordia where Mazikeen complains to the Goddess abo
ut the evils of human society. "Everyone is hurting each other, the planet is rampant wi
th injustices, whole societies plunder groups of their own people, mothers imprison son
s, children perish while brothers war."

The Goddess answers: "What is the matter with that, if it's what you want to do?"

Mazikeen: "But nobody wants it! Everybody hates it!"

Goddess: "Oh. Well, then stop."
```

Swapping tokens

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.

In [25]:
```
def swap(text, source, target):
    result = ''

    # YOUR CODE HERE

    i=0
    while i < len(text)  :
        if(text[i:i+len(source)] == source):
            result += target
            i=i+len(source)
        elif(text[i:i+len(target)] == target):
            result += source
            i=i+len(target)
        else :
            result += text[i]
            i = i+1

    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
        'about the evils of human society. "Everyone is hurting each other, the planet i
        'with injustices, whole societies plunder groups of their own people, mothers im
        'children perish while brothers war."\n' \
        '\n' \
        'The Goddess answers: "What is the matter with that, if it\'s what you want to d
        '\n' \
        'Malaclypse: "But nobody wants it! Everybody hates it!"\n' \
        '\n' \
        'Goddess: "Oh. Well, then stop."\n'
print(swap(text, 'Malaclypse', 'Goddess'))
```

```
There's a passage in the Principia Discordia where Goddess complains to the Malaclypse a
bout 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 so
ns, children perish while brothers war."

The Malaclypse answers: "What is the matter with that, if it's what you want to do?"

Goddess: "But nobody wants it! Everybody hates it!"

Malaclypse: "Oh. Well, then stop."
```

Working with text files

Lines and paragraphs

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.

In [26]:
```
def text_stats(fname):
    # YOUR CODE HERE
    total_lines = 0
    text_lines = 0
    text_paragraphs = 1
    flag = False


    try:
        with open(fname, 'r') as f:
            for line in f:
                total_lines += 1
                if line.strip():
                    text_lines += 1
                    flag = True
                else:
                    if flag:
                        text_paragraphs += 1
                        flag = False
    except FileNotFoundError:
        return None

    return total_lines, text_lines, text_paragraphs

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

```
(26, 16, 6)
```

Bonus points

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.

In [27]:
```
result = []
def flatten(l):

    # YOUR CODE HERE
    for val  in l:
        if(isinstance(val,int)):
            result.append(val)
        else :
            flatten(val)

    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))
```

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

In [ ]: