TU856/858-2 Object Oriented Programming with Python
Semester 1, Lab 4, 16.10.2020
Dr. Bianca Schoen-Phelan

Please join the virtual classroom in brightspace first, before going to individual lab links. I will provide explanations and hints and tricks regarding the lab at the beginning of the lab sessions. These are recorded.

Please sign your group's lab sign in sheet!
Lab solutions will be discussed in the Monday lecture. Labs are not marked.
Don't forget about your revision item for the week!

Link to the lab groups (click).
Link to python documentation (click).

Objectives:

- File handling
- Exceptions

ONLINE guideline:

- We meet in the virtual classroom in brightspace first. Just like the last lab.
- You will be able to stay within your own lab group throughout this online lab. Each tutor offers a teams link. You will be required to sign a sign-in sheet for each lab group.

1. **Wordcloud:**
    a. A word (or Tag) cloud is a visual representation for text data typically used to depict keyword metadata (tags) on websites, or to visualise free form text. Tags are usually single words, and the importance of each tag is shown with font size or colour. This format is useful for quickly perceiving the most prominent terms and for locating a term to determine its relative prominence. We are going to use Abraham Lincoln's Gettysburg address of 1863 - this is famous for many things, including for being a short speech. We are going to construct a simple web page. Each word will be enclosed in span tags and will have its individual font size set. A possible output might look like this:
    b. **Note: This is not a lab on HTML. The HTML tag code is provided to you!**

Figure 1: Example Wordcloud of the speech

2. **Fetch and merge the lab 4 files from the class repository BiancaSP/OOP2020-21:**
    a. Make sure that you are working on the master branch

```
$ git branch
 lab2
  lab3
* master
$ git fetch upstream
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 1), reused 5 (delta 1),
pack-reused 0
Unpacking objects: 100% (5/5), done.
From https://github.com/BiancaSP/OOP2020-21
   7aef0c3..85a0fd9  master     -> upstream/master
$ git merge upstream/master
Updating 7aef0c3..85a0fd9
Fast-forward
 Labs/gettisburg.txt      |  9 +++++++++
```

```
 Labs/lab4_word_cloud.py | 69
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++
 2 files changed, 78 insertions(+)
 create mode 100644 Labs/gettisburg.txt
 create mode 100644 Labs/lab4_word_cloud.py
$ git push origin master
```

b. There are two files in this lab:
   i. lab4_word_cloud.py
   ii. gettisburg.txt

c. Create a new branch before starting to edit your new lab files:
```
$ git checkout -b lab4
```
And verify that you are now working on this new branch
```
$ git branch
 lab2
  lab3
* lab4
  master
```

d. **After the lab** you may want to update your own github repository with this new branch you are working on now to preserve a copy of your solution online.:
```
$ git add Labs/lab4_word_cloud.py
$ git commit -m "my solution to the lab from 16.10.2020"
Labs/lab4_word_cloud.py
$ git log --oneline
6636317 (HEAD -> lab4) my own solution to the lab
16.10.2020
85a0fd9 (upstream/master, origin/master, origin/HEAD,
master) Lab Files for Week 4
7aef0c3 solution to lab 3
0216e30 lab 3 original file
d527792 Merge branch 'master' of
https://github.com/BiancaSP/OOP2020-21
a0cb601 lab 2 solution
9571621 removed loop exercise
37053a3 Delete test2
1378cbb Delete test1
59b0a7d test2
4ec5365 test
1cd04c3 original file lab 2
6ff1dbf File Lab 1
$ git push origin lab4
```
Verify in the browser that the material is up on your gitHub repository.

You will find this code provided in your github repo in the py file.

```
<!DOCTYPE html>
<html>
<head lang="en">
<meta charset="UTF-8"
<title>Tag Cloud Generator</title>
</head>
<body>
<div style="text-align: center; vertical-align: middle;
font-family: arial; color: white; background-color: black;
border: 1px solid black">
<!-- Your span elements should be inserted here→
</div>
</body>
</html>
```

The format of a span element is:

```
<span style="font-size: XXpx"> WORD </span>
```

For example:

```
<span style="font-size: 20px"> hello world </span>
```

3. **Write a html file from Python that creates a word cloud similar to the one shown in Figure 1.**

- Your Python code should count the occurrences of each word so you can calculate the size of each word
- Create a completed HTML page and write it to a .html file. The HTML file is already created for you in the py file. You only need to insert the program logic.
    - For testing you should open your html in a browser of your choice, or see 4)
- Remember about the debugger if you encounter any logical errors.
- When you have implemented a basic working program, improve it by disregarding stop words. You find a list of common stop words in the English language here: https://gist.github.com/sebleier/554280
    - You can start off with a few individual ones,
    - Then move on to reading the list of stop words from a file into an appropriate data type and include a loop over it for your word cloud generation

4. Running the lab file will create an output.html file. Load this file into a browser window. You can also double click on the file in the explorer window in Pycharm. This will first open the text file and then give you option to open it in a browser of your choice. You will see something like the following image:

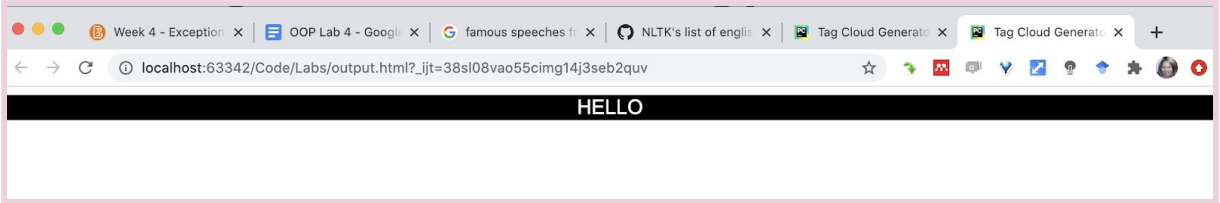

Figure 2: output.html created by the lab file

Hints:

1. Read the speech from the file and populate a **`dictionary`** data type. Use each word as a key to the dictionary. And the value of each word is the frequency of the word appearing in the text. The higher frequency words get displayed in larger font sizes in your word cloud.
    - An example dictionary might look like {'fathers': 1; '':2; 'those':1….}
    - An example for a dictionary creation and usage:
    -

```
my_motorbikes = {
    "brand": "Kawasaki",
    "model": "Ninja",
    "year" : 2010
}

print(my_motorbikes)

m = my_motorbikes.get("model")
print(m)

my_motorbikes["year"] = 2019
print(my_motorbikes)

for bike in my_motorbikes: # gets all keys
    print(bike)

for bike in my_motorbikes:
    print(my_motorbikes[bike]) # gets all values

for bike in my_motorbikes.values(): # does the same as before just
with values
    print(bike)

for key, value in my_motorbikes.items(): #gets both pieces of info
    print(key,value)

# checking if a key exists
if "model" in my_motorbikes:
    print("yes")

if "cc" in my_motorbikes:
    print("also here")
else:
    print("something not there")
```

```
# number of items
print(len(my_motorbikes))

# add an item
my_motorbikes["colour"] = "kawasaki green"
print(my_motorbikes)

# remove with pop, several options here how to remove an item
my_motorbikes.pop("colour")
print(my_motorbikes)

# empty the dictionary
my_motorbikes.clear()
print(my_motorbikes)

# copies need to be explicitly done via copy()
more_bikes = my_motorbikes.copy()
print(more_bikes)

# or using dict()
bikes_copy = dict(my_motorbikes)
print(bikes_copy)
```

- Python documentation on dictionaries: https://docs.python.org/3/tutorial/datastructures.html#dictionaries

2. You will need to create a span tag for each word in the dictionary. The font size will vary depending on the frequency of a word. For example, you can use count*10 for a size. So, words that appear once = 10px, words that appear twice = 20px size.

Well done!