|  |  |  |
| --- | --- | --- |
| **College of Engineering**  Computer Science & Eng. Dept.  **Course:** CMP 321L Programminglanguages Lab | A picture containing logo  Description automatically generated | **Course Professor:** Dr. Michel Pasquier  **Lab Instructor:** Praveena Kolli  **Office:** EB2-126  **Phone**: 971-6-5152352  **e-mail**: pkolli@aus.edu  **Semester**: Summer 2022 |

**Lab 8 – Revision**

**Objectives:**

* Practicing various data structures learned
* Code Python scripts that make use of regular expressions

**Due date: End of the lab. (**Only one team member needs to submit.)

**Rules:**

(1) Usage: **You should explore and make good use of the Python features you learned in class.** (2) Scope: **You should only use those features that have been explained in detail in class.**

(3) Style: Follow standard Python programming style and conventions.

(4) Logic: Add appropriate comments to your code to explain your solution.

(Code / answers that do not follow the above specifications will be penalized.)

***Warning:* You need to come to the lab properly prepared i.e.**

(1) Make sure you have studied and understood the class material.

(2) Read the lab doc, think about the problems, and prepare questions as needed.

If you do not, completing the lab in 2.45 hours may become too much of a challenge!

**Useful resources:**

<http://regex101.com/#python>

<https://docs.python.org/3/library/>

<https://docs.python.org/3/library/re.html>

**Lab By:**

Rohan Mitra 85023

Keshav Ramesh 88595

**Github link for code**: <https://github.com/ro1406/CMP321Lab>

**Exercise 1: file input and output and suitable data structures [4 marks]**

1. Use *urllib* to download the 500 most frequent English words (sorted by frequency) from <https://sketchengine.co.uk/wp-content/uploads/word-list/english/english-word-list-total.csv>. Process the file and store the words (only) in a suitable data structure *W*, keeping them in order. Print out the 20 most frequent English words and their rank, as follows: 1. 'the' - 2. 'and' - 3. 'to' - …
2. Download a story of famous computer bugs from <http://textfiles.com/100/famous.bug> as string *S.*
3. Remove from the text *S* all words that are not in *W.* Make sure your code is efficient.
4. Perform a word frequency count for the words in *S* and store the resulting data. Print the 20 most frequent words in string *S* using the same format shown in part (b).
5. Compare (using a program!) the 20 most frequent words in the text to the 20 most frequent words in the English language, and print out any major discrepancy. For example, it might be that the most frequent word is the same but the second most frequent word is different, etc. Keep efficiency in mind, as before.

**Code:**

import requests

def saveFile(url):

site=requests.get(url)

filename=url.split('/')[-1]

csv\_file = open(filename, 'wb')

csv\_file.write(site.content)

csv\_file.close()

return filename

fname=saveFile("https://sketchengine.co.uk/wp-content/uploads/word-list/english/english-word-list-total.csv")

print("The 20 top words in English:")

with open(fname,'r') as f:

lines=f.readlines()[4:24]

finalist=[(word[1].lower(),word[2]) for word in map(lambda x:x.split(';'), lines)]

engTopWords=list(map(lambda x:x[0],finalist))

for i,tup in enumerate(finalist):

print(f"{i+1}. '{tup[0]}' ",end="")

if tup!=finalist[-1]: print('- ',end="")

print()

print('-'\*90)

#Part b:

with open("TaleOfTwoCities.txt",'r') as f:

S=f.readlines()[0]

#part c:

newS=list(filter(lambda x: x in engTopWords,S.split()))

print('-'\*90)

#part d:

dic={}

for word in newS:

if word in dic:

dic[word]+=1

else:

dic[word]=1

print("All word frequencies from S Sorted:")

sortedS=sorted(dic,key=lambda x:dic[x],reverse=True)

print(sortedS)

print('-'\*90)

#part e:

for rank,(engWord,Sword) in enumerate(zip(engTopWords,sortedS)):

if engWord!=Sword:

print(f"{rank+1}) English popular '{engWord}' is not the same rank as '{Sword}' from S")

else:

print(f"{rank+1}) English popular '{engWord}' is the same rank as '{Sword}' from S")

**Screenshot:**

Text

Description automatically generated with medium confidence

**Exercise 2: More practice on regex [6marks]**

Write a python script that converts a given text file in simplified Markdown format to HTML and writes the result to another file, using regular expressions as much as possible.

A sample Markdown input file is given for testing purposes, which describes the *simplified Markdown syntax* that your program needs to understand i.e., paragraphs, headers, emphasis, lists, and code fragments (but *not* images, links, block quotes, or any other Markdown syntax).

**Help tools:** Markdown syntax is described at <http://daringfireball.net/projects/markdown/basics> .

Also, at <https://codebeautify.org/html-to-markdown> is an online tool to convert from HTML to Markdown, and vice versa – useful to explore and check your program output…

**Code:**

import re

def convert(s):

s=re.sub("<(\w+)>", "<code>\\1</code>", s) #Code Tags

s=re.sub("`&(.+);`", "<code>\\1</code>", s) #Code Tags

s=re.sub(r'(###)(.\*?)(###)',r'<h3>\2</h3>\n',s) #H3 Tags

s=re.sub(r'(###)(.\*?)(\n)',r'<h3>\2</h3>\n',s) #H3 Tags

s=re.sub(r'(.\*?)\n(--+)',r'<h2>\1</h2>',s) #H2 Tags

s=re.sub(r'(##)(.\*?)(##)',r'<h2>\2</h2>\n',s) #H2 Tags

s=re.sub(r'(.\*?)\n(=+)',r'<h1>\1</h1>',s) #H1 Tags

s=re.sub(r'(\\*\\*|\_\_)(.\*?)(\\*\\*|\_\_)',r'<strong>\2</strong>',s) #Strong

s=re.sub(r'(\\*|\_)(.\*?)(\\*|\_)',r'<em>\2</em>',s) #Emphasis

s=re.sub(r'(:)\n\n([123])',r'\1<ol>\2',s) #Ordered list start

s=re.sub(r'([123].\*?)\n\n',r'\1</ol>',s) #Ordered list end

s=re.sub(r'(:)\n\n([+\\*\-] )',r'\1<ul>\2',s) #Unordered list start

s=re.sub(r'([+\\*\-].\*?)\n\n',r'\1</ul>',s) #Unordered list end

s=re.sub(r'[\\*+\-] (.\*?\.)',r'<li>\1</li>',s) #Li tag UL

s=re.sub(r'[123]\.(.\*)',r'<li>\1</li>',s) #Li tag OL

res = ""

for i in s.split('\n\n'):

res += "".join('<p>' + i + '</p>')

res=re.sub(r' \n','<br>',res)

return res

with open("Lab8-InputFile.txt",'r') as f:

text=f.read()

s=convert(text)

myfile=open("Q2 Ans.html",'w')

myfile.write(s)

myfile.close()

**Screenshot:**

Text, table

Description automatically generated with medium confidence