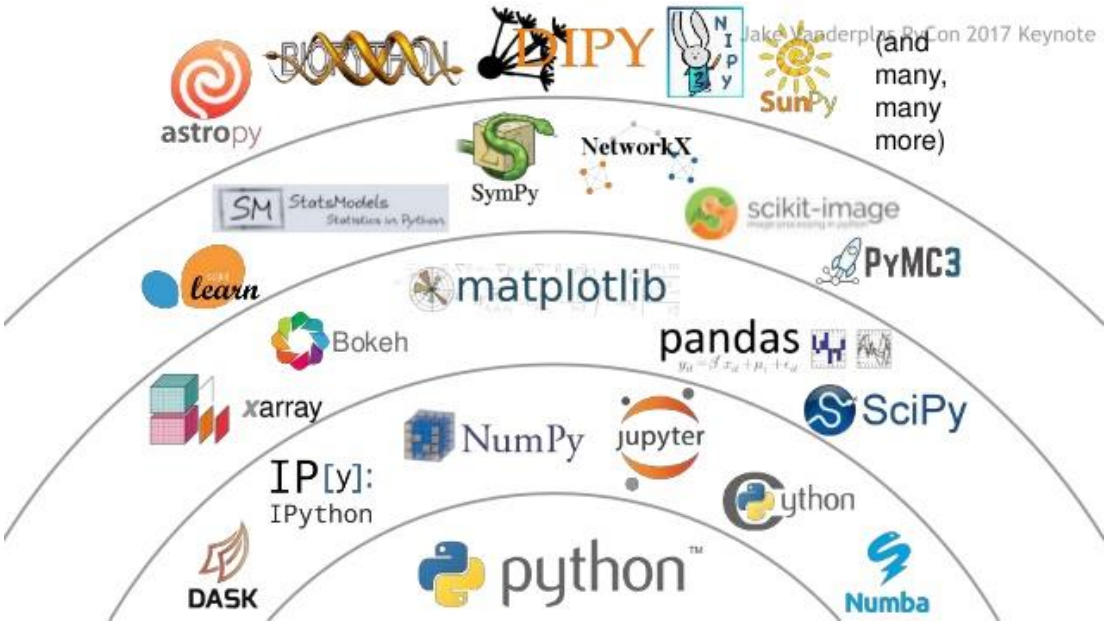
DEVELOPING SEO TOOL TO ANALYZE LIVE WEB PAGES

Name : Vignesh J Muralidharan

INDEX



1. Objective:

The project is to develop a generic toolset which will help a Web developer analyze Web pages for keywords and other components of the page which contribute to SEO and the keywords of interest with understanding its density pattern across the various components of the HTML page.

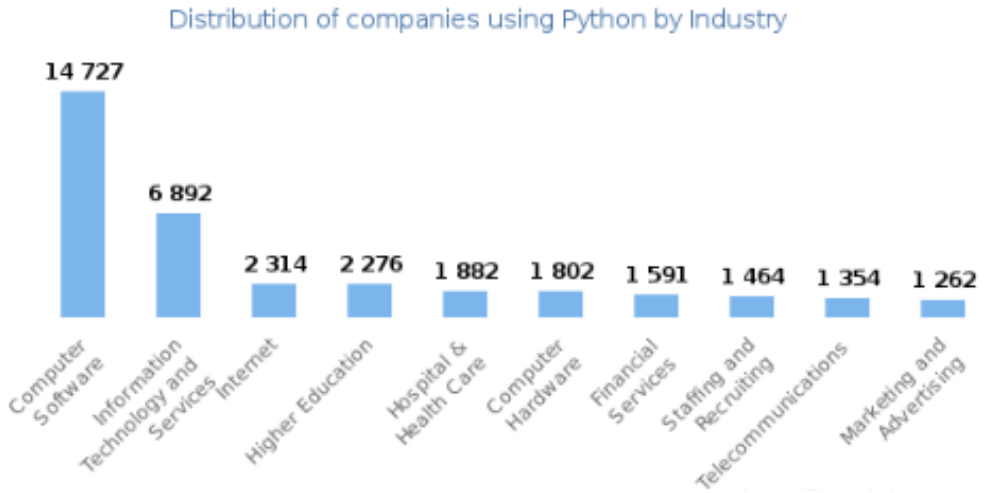
2. Python Introduction:

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by python software foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. Python is also considered as a dynamic, interpreted language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and we will lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not makes sense as it runs. An excellent way to see how python code works is to run the python interpreter and type code right into it. If we ever have a question like "What happens if I add an int to the list?" Just typing it into the python interpreter is a fast and likely the best way to see what happens.

3. Reasons to use Python:

- Readable and Maintainable Code : While writing a software application, we must focus on the quality of its source code to simplify maintenance and updates. The syntax rules of Python allow us to express concepts without writing additional code. At the same time, Python, unlike other programming languages, emphasizes on code readability, and allows us to use English keywords instead of punctuations.
- Multiple Programming Paradigms : Like other modern programming languages, Python also supports several programming paradigm. It supports object oriented and structured programming fully. Also, its language features support various concepts in functional and aspect-oriented programming.
- Compatible with Major Platforms and Systems : Python supports many operating systems. We can even use Python interpreters to run the code on specific platforms and tools. Also, Python is an interpreted programming language. It allows us to run the same code on multiple platforms without recompilation
- Robust Standard Library : Its large and robust standard library makes Python score over other programming languages. The standard library allows us to choose from a wide range of modules according to the precise needs. Each module further enables to add functionality to Python application without writing additional code.
- Open Source Frameworks and Tools : As an open source programming language, Python helps you to curtail software development cost significantly. We can even use several open source Python frameworks, libraries and development tools to curtail development time without increasing development cost.

Figure 1: Industries which uses python: *(Reference Taken from : iDataLabs Tech Products)*



4. SEO (Search Engine Optimization):

We should realize that web is a magnificent spot to discover or give data. SEO is an ongoing process and we stay updated always with respect to the latest updates by search engines. It utilizes of strategies intended on convey pages to the highest point of web search tool results pages for focused keywords to expand guest numbers. Web search tool optimizers have some expertise in advancing sites for customers. SEO speaks to the methodologies used to "optimize" the websites with search engines in mind, usually with the goal to improve the search ranking. SEO doesn't depend on the back-end technology of the website. If the output of the site is clean and sematic HTML then search engines should be able to crawl it just fine.

Figure 2 : Uses of SEO Process *(Reference Taken from : Book – SEO for Dummies, Peter Kent)*

5. Modules used in this project :

Python has a package manager called pip, which is system independent, it connects to python package index and downloads and installs the packages so it will require us to be connected to the internet. With pip, we can also update, remove packages that are already installed.

Import urlopen : This is a python module that can be used for fetching URL's. It defines functions and classes to help with URL actions (basic and digest authentication, redirections, cookies, etc). urlopen function offers a very simple interface which is also capable of fetching different protocols like HTTP, FTP,..).

Import BeautifulSoup : It provides idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

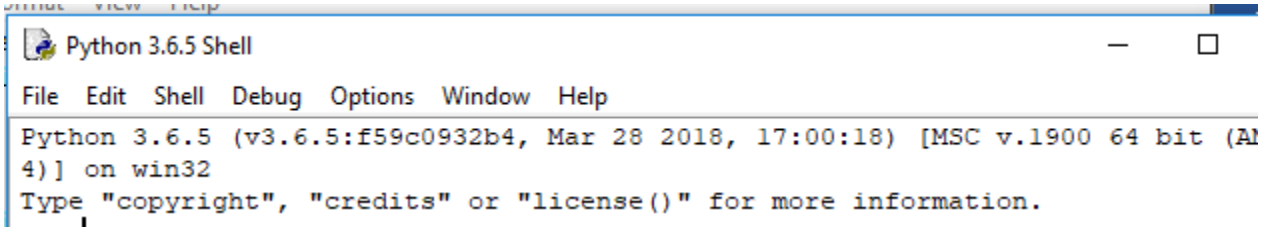
Import re (Regex) : The function returns None if the matching attempt fails, and a match object otherwise. Since None evaluates to False, we can easily use re in an if statement. The match object stores details about the part of the string matched by the regular expression pattern.

Import xlsxwriter : It is a tiny wrapper library to read, manipulate and write data in xlsx and xlsm format using openpyxl. The library makes information processing involving excel files an enjoyable task. The data in excel files can be turned into array or dict with minimal code and vice versa.

6. Steps used to attain the goals :

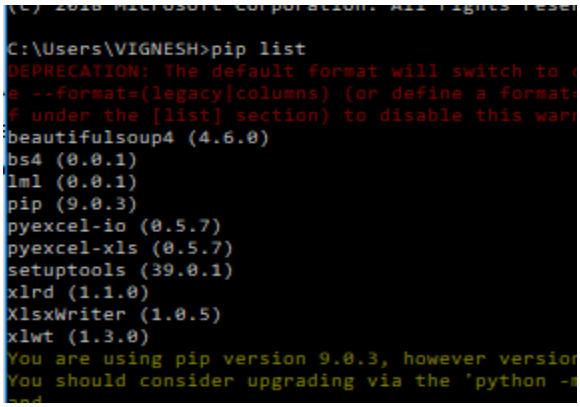
i. Installing correct python software:

The most stable python software listed at the time of beginning of the course was 3.6.5 and that was installed for the IDLE python from python.org for a 64 bit laptop.



ii. Installing modules using pip :

Before starting the project I tried to see all the installed modules in python and to upgrade the list I installed all the required modules using command prompt (CMD) with function “pip install” for pyexcel_xls, bs4, beautifulsoup4, xlrd, xlsxwriter.



iii. URL Input from user :

To make it little commutable the project was decided to get the URL from the user and so the request of urllib to import urlopen was made. With the exception handling concept the “try” of entering the URL was made. The except value error alone was made invalid else to return the text in the block.

iv. Usage of Regex :

To easily tackle many basic patterns using ordinary characters the regex is used to match the objects and compile the data extracted from the URL to use in the pattern. But it was not used here

v. Usage of Beautiful Soup :

To make the codes little readable of the imputed URL I have used beautiful Soup with the html.parsar to make it in text format and extract the not use of script and style using extract function.

```
fetchingurl=urlopen(url)
Soup=BeautifulSoup(fetchingurl,"html.parser")
for script in Soup(["script","style"]):
    script.extract()
    text=Soup.get_text()
```

vi. Splitting the text to get density

After we get the text from the given url the text needs to be split since we are generating the counts individually for each word and we need to figure out the density of all the words and update the list. In this scenario I have listed the text with split function and using density formula I used a for loop to fetch the density of the whole text and adding to the repeatability if words are accounted in website and range the density higher for the values. The density formula was used as given in the lab exercise and used here.

```
text=checkingtheurl()#the text
alllist=text.split()#all the ma
c=0#the total number of words
for x in alllist:
    c+=1
density={alllist[0]:1}# the den
for x in alllist:
    cx=0
    for b in alllist:#if the wo
        if(b==x):
            cx+=1
    density.update({x:cx/c})


li=list(density.values())#list
li.sort()
li.reverse()
```

vii. Zipping the values

After I got the density values list from top to bottom from high to low it was transferred into Dictionary with keywords listing in the variable "li" and the equivalent density values in "li2". With the zip function later only the top 15 words has been zipped with equivalent density values and listed later. The values have been zipped with the keys and the 1st for loop is for updating the dictionary with the all the keys and values in the website and the second 2nd for loop is for updating only the top 15 words which is equivalent to the i.

```
D={}
li2=sorted(density, key=density.__getitem__)
li2.reverse()
for key,value in zip(li2,li):
    D.update({key:value})
li=list(D.keys())
li2=list(D.values())

i=0
print("the top fifteen words in the given url are:\n")
for x,y in zip(li,li2):
    print(x,":",y)
    i+=1
    if(i==15):
        break
word=input("enter the word for which you want to know
```

viii. Creating workbook and worksheet

The variable name D is taken into the function writeExcelOutput later after importing xlsxwriter and write into an output file called "OutputExcel.xlsx" here we are creating to fetch the data and put into the wookbook. The global worksheet is created and with i=0 this is the cell position which will put in the specific column. The data in for loop from D is printed with 'x' as the keyword, 'y' as the count of the words in the website and 'z' as the density in the third column. This will be iterated with 15 times for i==15 and prepared the wook book with the worksheet.

```
worksheet=""
def writeExcelOutput(D):
    import xlsxwriter
    workbook=xlsxwriter.Workbook('outputexcel.xlsx')
    global worksheet
    worksheet=workbook.add_worksheet()
    i=0
    for data1,data2 in zip(D.keys(),D.values()):
        print(data1,":",data2)
        #print(data[1])#for counts
        x='B'+str(i)
        y='C'+str(i)
        z='D'+str(i)
        worksheet.write(x,data1)
        worksheet.write(y,data2*c)
        worksheet.write(z,data2)
        i=i+1 # number of times iteration
        if(i==15):
            break
    prepareChart(workbook);
    workbook.close()
    print('Successfully wrote')
```

ix. Preparing chart in Excel from python

We can create any chart from python and export to excel. Here for the usage I have used the line chart where we can specify with the ({'type':'line'}) and the add.series function will help in inputting the values from row and column wise into the excel. Here I wanted to list 15 words and so the sheet has been prepared to input 15 words.

In the chart we can also set the title: here I have set the title as Name vs Count and the x axis was the top 15 words in the website and y axis is the count. The set_style function will help in inputting different types of chart with colors and all and the chart location was placed as N10.

```
#LINE CHART
def prepareChart(workbook):
    global worksheet
    chart1=workbook.add_chart({'type':'line'})
    chart1.add_series({

        'categories':'=Sheet1!$B$1:$B$15',
        'values':'=Sheet1!$C$1:$C$15',

        })
    chart1.set_title({'name':'Name vs Count'})
    chart1.set_x_axis({'name':'top 15 words'})
    chart1.set_y_axis({'name':'count'})

    chart1.set_style(10)#SYTLE 10 IS COLORFULL
    worksheet.insert_chart('N10', chart1)#will c
    workbook.close()

writeExcelOutput(D)
```

7. Results and discussion – Step wise

➢ While running the code the first encounter we need to enter is the url which we need to analyze.

```
====== RESTART: D:\U.S.A\MS Statistical Sciences\Python\finalproject.py ======
Enter a website you want to analyze with url:

http://webpages.uidaho.edu/~brian/stat404.html
```

➢ After we enter the URL list of top 15 words with the density count is analyzed and displayed in shell like this

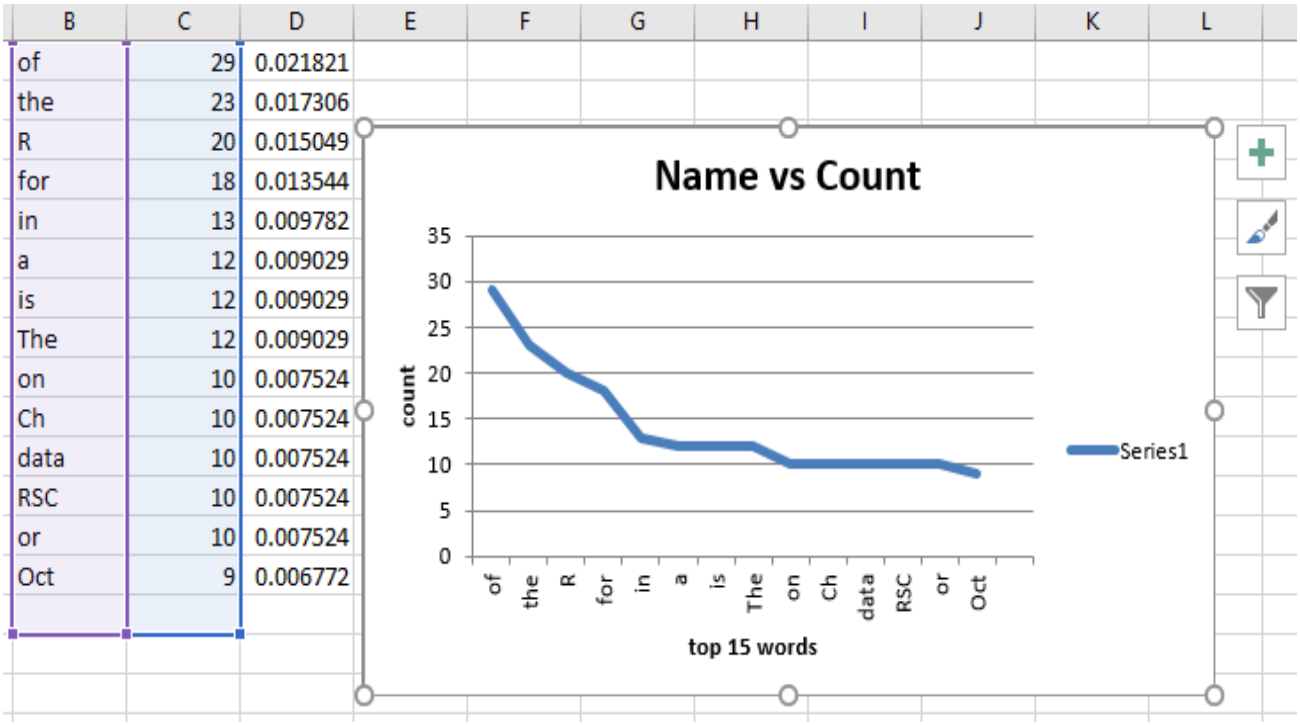
```
the top fifteen words in the given url are:

and : 0.0218209179834462
of : 0.0218209179834462
the : 0.017306245297215 95
R : 0.015048908954100828
for : 0.013544018058690745
in : 0.009781790820165538
a : 0.009029345372460496
is : 0.009029345372460496
The : 0.009029345372460496
on : 0.007524454477050414
Ch : 0.007524454477050414
data : 0.007524454477050414
RSC : 0.007524454477050414
or : 0.007524454477050414
Oct : 0.006772009029345372
```

➢ To confirm if the excel has been created the results display this in the shell after listing all the words.

```
Successfully wrote
```

- The excel file which is the output will be created in the same path where the code has been stored and looks like this



8. Conclusion

Most of the data analysis in the industries will do the same part. Some researchers if he needs any kind of analysis to be done with the information available in the website then we need this techniques to make sure we are going in the right path. Most of the time webscrapping is used to analyze how much our website is visible to others and how to make it more visibility with analyzing the keywords and this helps a lots in many applications. Python sometimes makes us feel surprised and also learnable to improve our knowledge and this example makes me surprised that this was the concept going on behind the website analysis and data extraction.

9. Reference

- Python book – NIIT
- Tutorials point website
- Govind pandit youtube python channel
- 12th std CBSE python book

10.Appendix and codes

```
#INSTALLING AND IMPORTING
from urllib.request import urlopen #request for url open module
from bs4 import BeautifulSoup #request for script readable with this module
import pyexcel_xls #request to export and edit the data with excel document
import re #reques the use of regex module installation
#USAGE OF BEAUTIFUL SOUP
def checkingtheurl():
    try:
        print("Enter a website you want to analyze with url:\n")
        url=input()
        from urllib.request import urlopen
        fetchingurl=urlopen(url)
        Soup=BeautifulSoup(fetchingurl,"html.parser")
        for script in Soup(["script","style"]):
            script.extract()
            text=Soup.get_text()
    except ValueError:
        print("The url entered is invalid !!\n")
        checkingtheurl()
    else:
        return(text
#GETTING KEYWORDS AND FINDING DENSITY

text=checkingtheurl()#the text of all the in the URL
alllist=text.split()#all the main text has been splited with the codes
c=0#the total number of words
for x in alllist:
    c+=1
density={alllist[0]:1}# the density of each word
for x in alllist:
    cx=0
    for b in alllist:#if the words match its gets counted and upadate the density
        if(b==x):
            cx+=1
    density.update({x:cx/c})#DENSITY EQUATION
#LISTING AND SORTING
li=list(density.values())#list of density values from top to bottom (high to low)
li.sort()
li.reverse()
#PUTTING INTO DICTIONARY AND ZIPPING THE VALUES WITH THE KEYWORDS
D={}
li2=sorted(density, key=density.__getitem__)#list of all the density values with spaces
li2.reverse()
for key,value in zip(li2,li): #ziping the both key words and the equalent density values
    D.update({key:value})
li=list(D.keys())
li2=list(D.values())
#LISTING TOP FIFTEEN WORDS AND ZIP THEM TOGETHER WITH THE DENSITY VALUES
i=0
print("the top fifteen words in the given url are:\n")
for x,y in zip(li,li2):
    print(x,":",y)
    i+=1
    if(i==15):
        break
word=input("enter the word for which you want to know the density of:\n")
#CREATING WORKBOOK AND WORKSHEET
worksheet=""
def writeExcelOutput(D):
    import xlsxwriter
    workbook=xlsxwriter.Workbook('outputexcel.xlsx')
    global worksheet
    worksheet=workbook.add_worksheet()
    i=0
    for data1,data2 in zip(D.keys(),D.values()):
        print(data1,":",data2)
        #print(data[1])#for counts
        x='B'+str(i)
        y='C'+str(i)
        z='D'+str(i)
        worksheet.write(x,data1)
        worksheet.write(y,data2*c)
        worksheet.write(z,data2)
        i=i+1 # number of times iteration
        if(i==15):
            break
    prepareChart(workbook);
    workbook.close()
    print('Successfully wrote')
##########################
#PREPARING CHART IN EXCEL
#LINE CHART
def prepareChart(workbook):
    global worksheet
    chart1=workbook.add_chart({'type':'line'})
    chart1.add_series({
        'categories':'=Sheet1!$B$1:$B$15',
        'values':'=Sheet1!$C$1:$C$15',
        })
    chart1.set_title({'name':'Name vs Count'})
    chart1.set_x_axis({'name':'top 15 words'})
    chart1.set_y_axis({'name':'count'})
    chart1.set_style(10)#SYTLE 10 IS COLORFULL
    worksheet.insert_chart('N10', chart1)#will create chart in that area
    workbook.close()
writeExcelOutput(D)
```