

## **Develop a SEO tool to analyse live web pages**

### **Project Description:**

Search Engine Optimization (SEO) is an important aspect of a web page to gain importance for a search engine to be able to display it earlier in the search list. The optimization is based on a lot of factors such as title, description, header tags and keyword density. Different search engines will have their own mechanisms that calculate the score of a keyword on the page and thus work out its ranking in the search order.

The project is to develop a generic SEO toolset which will help a web developer analyse live web pages for keywords and other components of the page which contribute to SEO. We should be able to provide the keywords of interest and understand its density pattern across the various components of the HTML page. The web page analysis should be developed to be done in a batch mode where we can analyse hundreds of web pages and the results should be saved as reports in spread sheets with graph presentations where necessary.

The project would not only give exposure to a real life problem solving using Python, but would also make learners understand about Internet and Search Engines and how they relate to each other.

### **What is Python?**

Python is my scripting language of choice. Its easy syntax, batteries-included attitude, and library for just about everything make it a great choice for a great many things. It makes tasks like writing titles and descriptions go quicker.

In short, knowing Python (or any other scripting language) gives a SEO the tools to get results for their clients quicker.

### **Users of Python**

- YouTube: Originally written in Python and mysql
- Yahoo!: Yahoo acquired Four11, whose address and mapping lookup services are implemented in Python
- Yahoo! Maps uses Python
- DropBox, a cloud based file hosting service runs on Python
- Google: Many components of the Google spider and search engine are written in Python

## **Traditional Uses of Python**

- Embedded Scripting
- Image processing
- Artificial intelligence
- GUI's
- Database Programming
- Internet Scripting
- System Administration
- Automation

## **Software and Hardware requirements**

- OS- Windows(64/32 bit), LINUX , MAC
- SW- Python(3+)
- SW- MySQL / SQL
- RAM- MIN 500 MB

## **Tool Used**

- Beautiful Soup
- XlsxWriter
- Sqlite3
- List
- Dictionary
- Tuple

## **PIP**

pip is a package management system used to install and manage software packages written in Python. Many packages can be found in the Python Package Index (PyPI)

`pip install package-name`

## **URLLIB**

urllib is a package that collects several modules for working with URLs:

`urllib.request` for opening and reading URLs

`urllib.error` containing the exceptions raised by `urllib.request`

`urllib.parse` for parsing URLs `urllib.robotparser` for parsing robots.txt files

EX: `urllib.urlopen(url[, data[, proxies[, context]])`

## **WEB SCRAPING WITH BEAUTIFUL SOUP**

Web scraping is to retrieve data that exists on a website, and convert it into a format that is usable for analysis. Webpages are rendered by the browser from HTML and CSS code, but much of the information included in the HTML underlying any website is not interesting to us.

We begin by reading in the source code for a given web page and creating a Beautiful Soup object with the Beautiful Soup function. Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

## **OPENPYXL**

Openpyxl is a Python library for reading and writing Excel 2010 xlsx/xlsm/xltx/xltm files. Professional support for openpyxl is available from Clark Consulting & Research and Adimian. Donations to the project to support further development and maintenance are welcome.

It was born from lack of existing library to read/write natively from Python the Office Open XML format.

## **SQLITE3**

The SQLite3 can be integrated with Python using `sqlite3` module which was written by Gerhard Haring. It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249. You do not need to install this module separately because it's being shipped by default along with Python version 2.5.x onwards.

To use `sqlite3` module, you must first create a connection object that represents the database and then optionally you can create cursor object which will help you in executing all the SQL statements.

EX: `sqlite3.connect (database [, timeout ,other optional arguments])`

## Source Code

```
from bs4 import BeautifulSoup
from urllib.request import urlopen
from openpyxl import load_workbook
import xlswriter
import sqlite3

#sqlite database location

conn=sqlite3.connect("C:/Users/Administrator/AppData/Local/Programs/Python/Python36-32/database3.db")

#conn.execute("create table datas(url varchar(30),keywords varchar(20),density varchar(10))")

workbook1=xlswriter.Workbook('outputfile.xlsx')


url=input("enter a url \n")
o=url
print(o)
fo=urlopen(url)
s=fo.read()
soup=BeautifulSoup(s,"html.parser")
for script in soup(["script","style"]):
    script.extract()
text=soup.get_text()
lines=(line.strip() for line in text.splitlines())
lis=list(lines)
st="".join(lis)
q=st.split()
qlen=len(q)
wordfreq = [q.count(w) for w in q]
```

```
out={}
```

```
wrds=input("enter five words ..\n")
```

```
s2=wrds.split()
```

```
for w1 in s2:
```

```
    w1=w1.lower()
```

```
    for a,b in zip(q,wordfreq):
```

```
        a=a.lower()
```

```
        if w1==a:
```

```
            out[w1]=b
```

```
print(out)
```

```
v=list(out.values())
```

```
k=list(out.keys())
```

```
length=len(v) + 3
```

```
den=[]
```

```
j=0
```

```
for fre in v:
```

```
    den.append((fre/qlen)*100)
```

```
chart = workbook1.add_chart({'type': 'column'})
```

```
wsheet = workbook1.add_worksheet()
```

```
#writing file
```

```
wsheet.write('A1',o )
```

```
wsheet.write_column('A3', k)
```

```
wsheet.write_column('B3', v)
```

```
wsheet.write_column('C3',den)
```

```

l='=Sheet1!$B$3:$B$'+str(length)

chart.add_series({'values':l})

wsheet.insert_chart('C10', chart)


workbook1.close()

i=0

try:

    while i<=4:

        conn.execute("insert into datas(url,keywords,density)values(?,?,?)"',(o,k[i],v[i]));

        i=i+1

    conn.commit()

except BaseException:

    print("No more word(s) found ")


with open('dboutput.csv', 'w') as write_file:

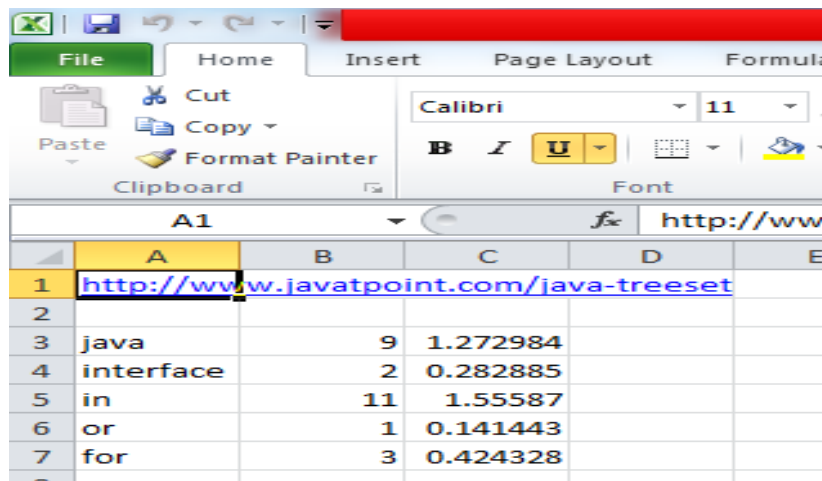
    cursor = conn.cursor()

    for row in cursor.execute('SELECT * FROM datas'):

        write_file.writelines(row)

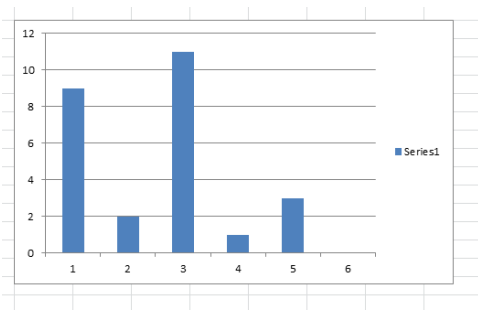
```

### Data in Excel:



|   | A   | B  | C        | D | E |
|---|---|----|----------|---|---|
| 1 | <a href="http://www.javatpoint.com/java-treeset">http://www.javatpoint.com/java-treeset</a> |    |          |   |   |
| 2 |   |    |          |   |   |
| 3 | java  | 9  | 1.272984 |   |   |
| 4 | interface   | 2  | 0.282885 |   |   |
| 5 | in  | 11 | 1.55587  |   |   |
| 6 | or  | 1  | 0.141443 |   |   |
| 7 | for   | 3  | 0.424328 |   |   |

## Chart Generated



## Conclusion

With the help of various techniques in python I have completed the “Develop a SEO tool to analyse live web pages”.