Unit -3

Regular Expressions

Regular Expression

- A RegEx, or Regular Expression, is sequence of characters that forms a search pattern.
- It can be used to check if string contains specified search pattern.
- Python has a built-in package called re, which can be used to work with Regular Expressions.
- **Import the re module:** import re
- There are various characters, which would have special meaning when they are used in regular expression. To avoid any confusion while dealing with regular expressions, we would use Raw Strings as **r'expression'**.

Sequence Characters

- Sequence Characters match only one character in the string.

Character	Description	Example
\A	Returns a match if the specified characters are at the beginning of the string	"\AThe"
\b	Returns a match where the specified characters are at the beginning or at the end of a word	r"\bain" r"ain\b"
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word	r"\Bain" r"ain\B"
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s"
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"
\Z	Returns a match if the specified characters are at the end of the string	"Spain\Z"

Special Characters

- Characters with special significance

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
|	Either or	"falls|stays"
()	Capture and group	

Quantifiers

- Quantifiers represent more than one character to be matched in string.

Character	Description	Example
+	1 Or more repetitions of pattern.	\s+
*	0 Or more repetitions of regex pattern.	\w*
?	0 Or 1 repetitions of regex pattern.	\w?
{m}	Exactly m occurrences	\d{10}
{m,n}	From m to n	\d{1,6}

Regular Expression Functions

- The re module offers a set of functions that allows us to search a string for a match:

Function	Description
findall	Returns a list containing all matches
search	Returns Match object if there is match anywhere in string
split	Returns a list where string has been split at each match
sub	Replaces one or many matches with a string
match	This function attempts to match RE *pattern* to *string*

Search Function

- The search() method searches the string from beginning till the end and returns first occurrence of the matching string, otherwise it returns None.
- We can use group() method to retrieve the string from the object returned by this method.
- Here is syntax for this function – **re.search(pattern, string, flags=0)**

Sr.No.	Parameter & Description
1	Pattern: This is the regular expression to be matched.
2	String:This is the string, which would be searched to match the pattern anywhere in the string.
3	Flags:specify different flags using bitwise OR (|)

Search Function Example

- search for strings starting with m and having total 3 characters.

```
import re
str="man sun map run"
result=re.search(r'm\w\w',str)
if result:
        print result.group()
```

- Output:- man

Match Function

- The match() method searches the string in the beginning and if the matching string is found, it returns an object that contains the resultant string, otherwise it returns None.
- We can use group() method to retrieve the string from the object returned by this method.
- Here is syntax for this function – **re.match(pattern, string, flags=0)**

Sr.No.	Parameter & Description
1	Pattern: This is the regular expression to be matched.
2	String:This is the string, which would be searched to match the pattern anywhere in the string.
3	Flags:specify different flags using bitwise OR (|)

Match Function Example

- search for strings starting with m and having total 3 characters.

```
import re
str="man sun map run"
result=re.match(r'm\w\w',str)
print result.group()
```

- output :- man

Findall Function

- The match() method searches the string from the beginning till the end and returns all occurrences of the matching string in the form of a list object.
- If the matching strings are not found, then it returns an empty list.
- We can retrieve the resultant strings from the list using a for loop.
- syntax for this function – **re.findall(pattern, string, flags=0)**

Sr.No.	Parameter & Description
1	Pattern: This is the regular expression to be matched.
2	String:This is the string, which would be searched to match the pattern anywhere in the string.
3	Flags:specify different flags using bitwise OR (|)

Find all Function Example

- search for strings starting with m and having total 3 characters.

```
import re
str="man sun map run"
result=re.findall(r'm\w\w',str)
print result
```

- output :- ['man' , 'mop']

Split Function

- The split() method splits the string according to the regular expression and the resultant pieces are returned as a list.
- If there are no string pieces, then it returns an empty list.
- We can retrieve the resultant strings from the list using a for loop.
- syntax for this function – **re.split(pattern, string)**

Sr.No.	Parameter & Description
1	Pattern: This is the regular expression to be matched.
2	String: This is the string, which would be searched to match the pattern anywhere in the string.

Split Function Example

- split a string into pieces where one or more non alpha numeric characters are found.

```
import re
str='This: is the: "Core" Python\'s book'
result=re.findall(r'\W+',str)
print result
```

- output :- ['This' , 'is', 'the', 'Core','Python', 's','book']

Using Regular Expressions on Files

- Use open() method to open the file. readlines() is used to all lines of the file.
- decode method will be used to convert the data into string.
- Example:

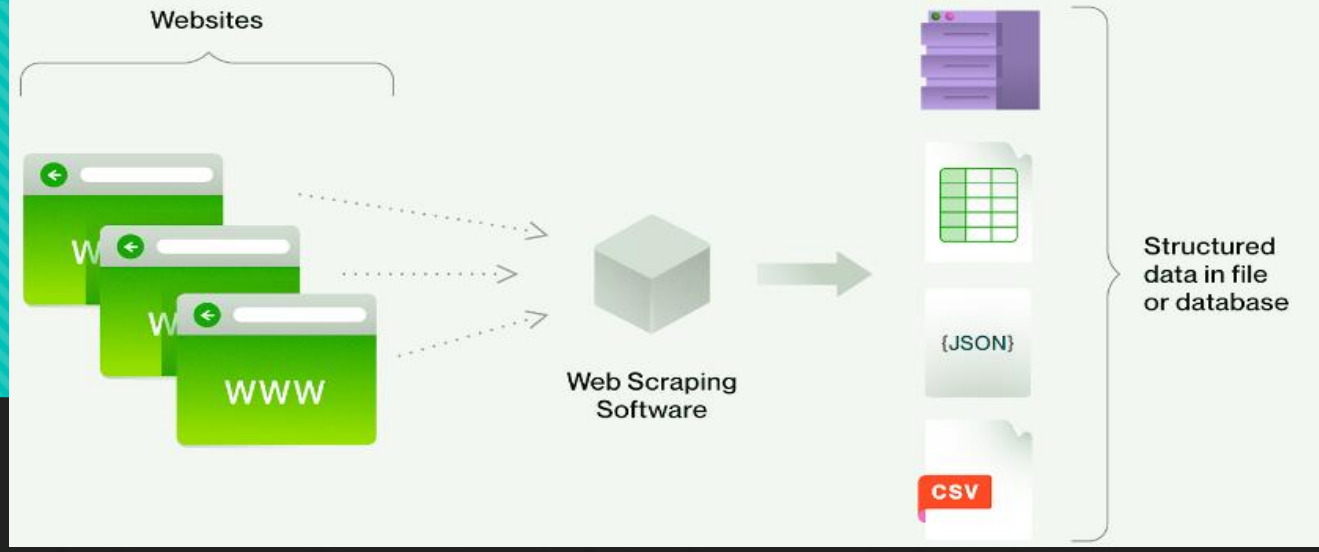
```
F=open('mails.txt','rb')
Data=F.readlines()
for line in Data:
        l=line.decode()
        result=re.findall(r'\S+@\S+',l)
        if result:
                print(result)
F.close()
```

Regular Expressions on HTML Files

- To open HTMl file, we have to use urlopen() method of urlib. Request module in python.
- Once file is open, read() method is used to read the data and decode() method will be used to convert the data into string format.
- Example:

```
Import re
from urllib.request import urlopen
url = "http://olympus.realpython.org/profiles/dionysus"
page = urlopen(url)
 html = page.read().decode("utf-8")
pattern = "<title.*?>.*?</title.*?>"
match_results = re.search(pattern, html, re.IGNORECASE)
title = match_results.group()
 title = re.sub("<.*?>", "", title)
print(title)  # Remove HTML tags
```

Screen Scraping



- Screen scraping, also known as web scraping, is a technique used to extract data from websites by simulating human interaction with web pages. It involves retrieving the HTML content of a web page and parsing it to extract the desired information.
- Screen scraping can be performed using programming languages like Python, along with libraries such as BeautifulSoup or Scrapy, which facilitate the extraction of data from HTML or XML documents. The process typically involves sending an HTTP request to a web page, receiving the response, and then parsing and manipulating the HTML content to extract specific data points.
- Screen scraping can be useful in various scenarios, such as:

1. Data extraction: Gathering information like product details, prices, reviews, or news articles from websites.
2. Data aggregation: Collecting data from multiple sources to create comprehensive databases or comparison platforms.
3. Monitoring: Tracking changes in website content, such as stock prices, weather updates, or social media trends.
4. Research and analysis: Gathering data for market research, sentiment analysis, or academic purposes.

Screen Scraping

- In python we need to install beautifulsoup module as : pip install beautifulsoup4
- Example:

```
from bs4 import BeautifulSoup
from urllib.request import urlopen

url = "https://developers.google.com/edu/python/introduction"
page = urlopen(url)
html = page.read().decode("utf-8")
soup = BeautifulSoup(html, "html.parser")
#print(soup.get_text())
paragraphs = soup.find_all("h2")
for paragraph in paragraphs:
    print(paragraph.get_text())
```