(25) Web scrapping:

- Web scrapping is the process of gathering information from the internet. even copy pasting lyrics of your fav song is a form of web scrapping.
- the words "Web scrapping" usally refers to process that involves automation

(1) Web scrapping is a technique to fetch data from websites.

There are two methods are used for data collection (fetch).

(i) copy-paste (manually). Manual Extraction Techniques

(ii) Web scrapping. Automated Extraction Techniques.

(i) copy-paste

- copy-past data process is tedius and time consuming.
- manually collecting or gathering data from web.

(ii) Web scrapping

- Web scrapping is alltomation of data extraction process from web.
- automatically load and extract data from websites.

Uses of Web scrapping

(i) Brand monitoring and competition Analysis.
(ii) Machine Learning.
(iii) Financial Data Analysis.
(iv) social Media Analysis.
(v) SEO monitoring.
(vi)

(2) Techniques of Web scrapping.

- There are Two ways to extracting data from websites.
  (i) Manual Extraction Techniques.
  (ii) Automation Extraction Techniques.

(i) Manual Web scrapping Techniques.

- copy pasting data from websites.
- tedious and time consuming.
- sepeteline is an effective way to scrapping data

(ii) Automated Extraction Technique.

- Web scrapping software used for extracting data from websites
  (i) HTML parsing.
  (ii) Dom parsing
  (iii) Web scrapping software.

(i) HTML parsing:

- parsing means make something understandable to be analysing
- to wit, that means convert information in one form to another
- HTML parsing means taking in the code and extracting relavent information from it based on user requirements.
- mainly using javascript and HTML page.

(ii) Dom parsing:

- Dom stands form Document object model.
- It defines interface that enable users to modify and update structure and xml documents.

(iii) Web scrapping software:

- using tools. extract data from web
  - import.io
  - Webstro-io
  - Dexi.io
  - Scrappinghub.
  - parsehub

③ challenges to Web scrapping.

(i) Data Warehousing.

- Difficult to storing, extracting, and searching it data Warehouse infrastructure not properly built.
- large-scale data extraction, there needs to be a perfect Data Warehousing system.

(ii) Website structure changes:

(iii) Anti scrapping Technologys

(iv). Quality of Data Extracted.

(ii) Website structure changes:

- Every website update user interface to attractiveness and experience
- This required various structural changes.
-

(iii) Anti scrapping Technologys:

- apply dynamic coding algorithm to prevent any bot intervention and use IP blocking mechanism.
- require a lot of time, money, work around Anti scrapping Techn

(iv). Quality of Data Extracted:

- Records that do not meet quality of information required will affect the overall integrity of data.
- Difficult task to meet quality of Data in real-time.

(26) Regular Expression (Regex).

- A regular expression is a powerful tool for matching text on pre-defined pattern.
- Regular expression can detect the presence of text by matching with a particular pattern, also split into one or more sub patterns.
- If we want to go represent a group of strings according to particular pattern then we should go for REs.
- Regular expressions are widely used in UNIX world.
- Regular expression are powerful language for matching text pattern.
- Regular Expression, is a sequence of characters that forms a search pattern.
- Regex can be used to check if a string contains the specified search pattern.
- python has built in package called re, which can be used to work with Regular expressions.

```
import re.
```

- re module offers a set of function that allows us to search a string for a match.

(i) Findall
(ii) search
(iii) split.
(iv). sub.

(i) findall

- Return a list containing all matches.
- The list contains the matches, in the order they found.
- If no matches are no found, an empty list is returned.

```
X = re.findall(" ", text).
```

(ii) search:

- Return a match object if there is match anywhere in a string.
- search() function searches the string for a match, and returns a match object if there is match.
- if there is more than one match, only the first occurrence of the

will be returned.
- if No matches found, the value None is returned.
  x = re.search(" ", text)

(iii) split:
- Return list where the string has been split at each match.
- Return a list where the string has been split at each match (split())
- control the Number of occurances by specifying maxsplit parameter.
  x = re.split(" ", text).

(iv) sub:
- Replace one or many matches with a string.
- sub() function replaces the matches with the text of your choice.
- control the Number of replacements by specifying count parameter.
  x = re.sub(" ", text) ← single.
  x = re.sub(" ", text, 2) ← two occerances.

(2) Metacharacters:
character.

[]	A set of characters.	"[a-z]"
\	signal special sequence [used to escape special characters]	"\d"
.	Any character [except Newline character]	"he..o"
^	starts with	"^hello"
$	Ends with.	"world$"
*	Zero or more occerances.	"a1x*"
+	one or more occurances.	"a1x+"
{ }	Exactly specified number of occerances.	"a1{2}"
|	Either or	"falls|stays"
( )	capture and group.	

- A special sequence is a \ followed by one of the characters in list below.

(3) special sequence:

Character	
\A	Returns a match if specified characters at the beginning of string
\b	Returns a match where the specified characters at beginning or at end of word. ε ← in the beginning (treated as raw string).
\B.	Returns match specified characters are present but not beginning of word (or at the end).
\d	Returns match where string contains digits
\D	Returns match where string does not contain digit
\s	Returns match where string contains white space characters.
\S	Returns match where the string DOES NOT contain a white space characters.
\w	Returns match where the string ~~DOES NOT~~ contain any word characters. (a-z, 0-9, _).
\W	Returns match where string DOES NOT contain any word character.
\Z	Returns match if the specified characte at the end of string.

④ sets:

set	Description.
[aεn]	Return match where one of specified characters (a, ε, or n) are present.
~~[a=n]~~	
~~[^aεn]~~	
[a-n]	Returns match any lowercase characters, alphabetically between a and n.
[^aεn]	Returns match for any character EXCEPT a, ε, and n.
[0123]	Returns match where any specified digits (0, 1, 2, and 3) are present.
[0-9]	Returns match for any digit between 0 to 9.
[0-5][0-9]	Returns match for any two digits Numbers 00 and 59
[a-zA-Z]	Returns match for any character alphabetically between a and z, lowercase or Uppercase.
[+]	In sets, +, *, ., 1, (), $, £} has no special meaning, so [+] means, return a match for any + character in a string.