EXPERIMENT NO: 04

TITLE: - Write down a Python Code for Crawling to the website

CLASS: - F. Y. B. Tech. (ECO) **BATCH: -**

DATE:

ROLL NO: -

Title: Write down a Python Code for Crawling to the website

Equipment: Computer with jupyter notebook.

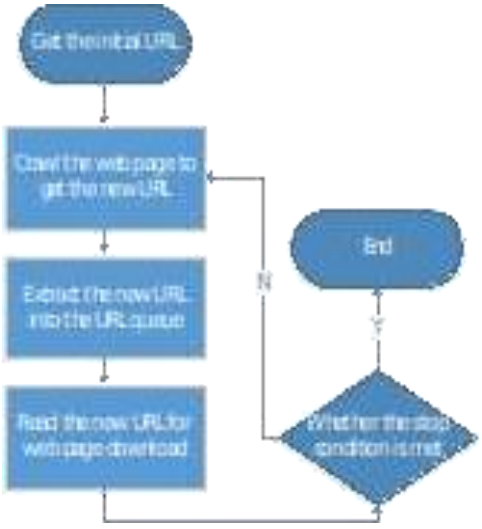
Theory:

With the advent of the era of big data, the need for network information has increased widely. Many different companies collect external data from the Internet for various reasons: analyzing competition, summarizing news stories, tracking trends in specific markets, or collecting daily stock prices to build predictive models. Therefore, web crawlers are becoming more important. Web crawlers automatically browse or grab information from the Internet according to specified rules.

CLASSIFICATION OF WEB CRAWLERS

According to the implemented technology and structure, web crawlers can be divided into general web crawlers, focused web crawlers, incremental web crawlers, and deep web crawlers.

BASIC WORKFLOW OF WEB CRAWLERS



The basic workflow of a general web crawler is as follows:

- Get the initial URL. The initial URL is an entry point for the web crawler, which links to the web page that needs to be crawled;
- While crawling the web page, we need to fetch the HTML content of the page, then parse it to get the URLs of all the pages linked to this page.
- Put these URLs into a queue;

- Loop through the queue, read the URLs from the queue one by one, for each URL, crawl the corresponding web page, then repeat the above crawling process;
- Check whether the stop condition is met. If the stop condition is not set, the crawler will keep crawling until it cannot get a new UR

ENVIRONMENTAL PREPARATION FOR WEB CRAWLING

1. Make sure that a browser such as Chrome, IE or other has been installed in the environment.
2. Download and install Python
3. Download a suitable IDL
   This article uses Visual Studio Code
4. Install the required Python packages
   Pip is a Python package management tool. It provides functions for searching, downloading, installing, and uninstalling Python packages. This tool will be included when downloading and installing Python. Therefore, we can directly use 'pip install' to install the libraries we need.

```
1. pip install beautifulsoup4
2. pip install requests
3. pip install lxml
```

- BeautifulSoup is a library for easily parsing HTML and XML data.
- lxml is a library to improve the parsing speed of XML files.

• requests is a library to simulate HTTP requests (such as GET and POST). We will mainly use it to access the source code of any given website.

Conclusion :

Question:-

1. What is web crawling?
2. What are different libraries used in web crawling?

EXPERIMENT NO: 05

TITLE: - TensorFlow to multiply two constants

CLASS: - F. Y. B. Tech. (ECO) **BATCH: -**

DATE:

ROLL NO: -

Title:- Write down a Python Code using TensorFlow to multiply two constants & print the result

APPRATUS: - Computer with jupyter notebook.

THEORY: -

Tensorflow is an open-source Python framework that is mainly used to build and train machine learning models. Tensorflow provides a range of functions to deal with data structures, such as single dimension or multi-dimensional arrays known as **tensors**.

Tensor multiplication is an important part of building machine learning models, as all data (images or sounds) is represented in the form of tensors. The **tensorflow.math.multiply() function** is used for this purpose.

Syntax

```
1
tensorflow.math.multiply(x, y, name)
```

Parameters

- x: Input tensor to be multiplied.
- y: Input tensor that multiplies. y must have the same data type as x.
- name: Defines the name of the operation. This parameter is optional.

Return value

The function returns a tensor object with the data type of the input tensors

CONCLUSION: -

Question:

1. what is tensorfolw?
2. What are different methods used in tensorflow?

EXPERIMENT NO: 06

TITLE: - TensorFlow in python

CLASS: - F. Y. B. Tech. (ECO) **BATCH: -**

DATE:

ROLL NO: -

Title : Write down a Python Code to start up an interactive Session, run the result and close the Session automatically again after printing the output using TensorFlow.

Equipment: Computer with jupyter notebook

Theory:

tf.Graph and tf.Session in TensorFlow:

- A graph defines the computation. It doesn't compute anything, it doesn't hold any values, it just defines the operations that you specified in your code.
- A session allows to execute graphs or part of graphs. It allocates resources (on one or more machines) for that and holds the actual values of intermediate results and variables.

Let's look at an example.

Update 2020-03-04: Sessions are gone in TensorFlow 2. There is one global runtime in the background that executes all computation, whether run eagerly or as a compiled tf.function.

Defining the Graph

We define a graph with a variable and three operations: variable returns the current value of our variable. initialize assigns the initial value of 42 to that variable. assign assigns the new value of 13 to that variable.

Conclusion:

Question:

1.what is tensorflow session?

EXPERIMENT NO: 07

TITLE: - Seaborn Library in Python

CLASS: - F. Y. B. Tech. (ECO) **BATCH: -**

DATE:

ROLL NO: -

Title: Write down a Python Code for experiment number 1 using Seaborn

Equipment: Computer with jupyter Notebook

Theory:

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

seaborn.factorplot() method is used to draw a categorical plot onto a FacetGrid.
Syntax : *seaborn.factorplot(x=None, y=None, hue=None, data=None, row=None, col=None, col_wrap=None, estimator=, ci=95, n_boot=1000, units=None, seed=None, order=None, hue_order=None, row_order=None, col_order=None, kind='strip', height=5, aspect=1, orient=None, color=None, palette=None, legend=True, legend_out=True, sharex=True, sharey=True, margin_titles=False, facet_kws=None, **kwargs)* ***Parameters :*** *This method is accepting the following parameters that are described below:*

- ***x, y :*** *This parameter take names of variables in data, Inputs for plotting long-form data.*
- ***hue :*** *(optional)This parameter take column name for colour encoding*
- ***data :*** *This parameter take DataFrame, Long-form (tidy) dataset for plotting. Each column should correspond to a variable, and each row should correspond to an observation.*
- ***row, col :*** *(optional) This parameter take names of variables in data, Categorical variables that will determine the faceting of the grid.*
- ***col_wrap :*** *(optional) This parameter take integer value, "Wrap" the column variable at this width, so that the column facets span multiple rows. Incompatible with a row facet.*
- ***estimator :*** *(optional) This parameter take callable that maps vector -> scalar, Statistical function to estimate within each categorical bin.*
- ***ci :*** *(optional) This parameter take float or "sd" or None value, Size of confidence intervals to draw around estimated values. If "sd", skip bootstrapping and draw the standard deviation of the observations. If None, no bootstrapping will be performed, and error bars will not be drawn.*
- ***n_boot :*** *(optional) This parameter take integer value, Number of bootstrap iterations to use when computing confidence intervals.*
- ***units :*** *(optional) This parameter take name of variable in data or vector data, Identifier of sampling units, which will be used to perform a multilevel bootstrap and account for repeated measures design.*
- ***seed :*** *(optional) This parameter take integer value, numpy.random.Generator, or numpy.random.RandomState, Seed or random number generator for reproducible bootstrapping.*
- ***order, hue_order :*** *(optional) This parameter take lists of strings, Order to plot the categorical levels in, otherwise the levels are inferred from the data objects.*
- ***row_order, col_order:*** *(optional) This parameter take lists of strings, Order to organize the rows and/or columns of the grid in, otherwise the orders are inferred from the data objects.*

- ***kind :*** *(optional) This parameter take string value, The kind of plot to draw (corresponds to the name of a categorical plotting function. Options are: "point", "bar", "strip", "swarm", "box", "violin", or "boxen".*
- ***height :*** *(optional) This parameter take float value, Height (in inches) of each facet.*
- ***aspect :*** *(optional) This parameter take float value, Aspect ratio of each facet, so that aspect * height gives the width of each facet in inches.*
- ***orient :*** *(optional) This parameter take value that should be "v" | "h", Orientation of the plot (vertical or horizontal). This is usually inferred from the dtype of the input variables, but can be used to specify when the "categorical" variable is a numeric or when plotting wide-form data.*
- ***color :*** *(optional) This parameter take matplotlib color, Color for all of the elements, or seed for a gradient palette.*
- ***palette :*** *(optional) This parameter take palette name, list, or dict, Colors to use for the different levels of the hue variable. Should be something that can be interpreted by color_palette(), or a dictionary mapping hue levels to matplotlib colors.*
- ***legend :*** *(optional) This parameter take boolean value, If True and there is a hue variable, draw a legend on the plot.*
- ***legend_out :*** *(optional) This parameter take boolean value, If True, the figure size will be extended, and the legend will be drawn outside the plot on the center right.*
- ***share{x, y} :*** *(optional) This parameter take bool, 'col', or 'row', If true, the facets will share y axes across columns and/or x axes across rows.*
- ***margin_titles :*** *(optional) This parameter take boolean value, If True, the titles for the row variable are drawn to the right of the last column. This option is experimental and may not work in all cases.*
- ***facet_kws :*** *(optional) This parameter take dictionary object, Dictionary of other keyword arguments to pass to FacetGrid.*
- ***kwargs :*** *This parameter take key, value pairings, Other keyword arguments are passed through to the underlying plotting function.*

Returns : *This method returns the FacetGrid object with the plot on it for further tweaking.*

Conclusion:

Ques

1. what are different methods used in seaborn library?

EXPERIMENT NO: 08

TITLE: - Bokeh visualization tool

CLASS: - F. Y. B. Tech. (ECO) **BATCH: -**

DATE:

ROLL NO: -

Title: Write down a Python Code for turning your data into a visualization using Bokeh.

Equipment: Computer with jupyter notebook

Prerequisites: Dynamic memory allocation, Structures and pointers.

Theory: Bokeh is a Python library which is used for data visualization through high-performance interactive charts and plots. It creates its plots using HTML and JavaScript languages. The output of the bokeh library can be generated on several platforms such as browser, HTML, server, and notebook. It is also possible to create the bokeh plots in Django and flask applications.

The bokeh library provides two visualization interfaces to the users:

- **models:** it is a low-level interface which provides high flexibility to the application developers.
- **plotting:** it is a high-level interface which is used for creating visual glyphs.

Conclusion:

Question:

1 Explain with example, difference between level of a node and height of node?

2 Discuss the various cases if node in binary search tree is to be deleted?