EXPLORE ON URL LIBRARY

URL LIBRARY

What is URL LIBRARY

- Urllib package is the URL handling module for python. It is used to fetch URLs (Uniform Resource Locators). It uses the ***urlopen*** function and is able to fetch URLs using a variety of different protocols.

Urllib is a package that collects several modules for working with URLs, such as:

- urllib.request for opening and reading.
- urllib.parse for parsing URLs
- urllib.error for the exceptions raised
- urllib.robotparser for parsing robot.txt files

urllib.request

- This module helps to define functions and classes to open URLs (mostly HTTP). One of the most simple ways to open such URLs is : ***urllib.request.urlopen(url)***

We can see this in an example:

```
import urllib.request
request_url = urllib.request.urlopen('https://www.geeksforgeeks.org/')
print(request_url.read( ))
```

Example 2:

```
import urllib.request
# open a connection to a URL using urllib
webUrl = urllib.request.urlopen('https://www.youtube.com/user/guru99com')
#get the result code and print it
print ("result code: " + str(webUrl.getcode()))
# read the data from the URL and print it
data = webUrl.read()
print (data)
```

urllib.parse

- This module helps to define functions to manipulate URLs and their components parts, to build or break them. It usually focuses on splitting a URL into small components; or joining different URL components into URL strings.

EXAMPLE :

```
from urllib.parse
import * parse_url = urlparse('https://www.geeksforgeeks.org / python-
langtons-ant/')
print(parse_url)
print("\n")
unparse_url = urlunparse(parse_url)
print(unparse_url)
```

Function	Use
urllib.parse.urlparse	Separates different components of URL
urllib.parse.urlunparse	Join different components of URL
urllib.parse.urlsplit	It is similar to urlparse() but doesn't split the params
urllib.parse.urlunsplit	Combines the tuple element returned by urlsplit() to form URL
urllib.parse.urldeflag	If URL contains fragment, then it returns a URL removing the fragment.

urllib.error

This module defines the classes for exception raised by urllib.request. Whenever there is an error in fetching a URL, this module helps in raising exceptions. The following are the exceptions raised :

- URLError – It is raised for the errors in URLs, or errors while fetching the URL due to connectivity, and has a 'reason' property that tells a user the reason of error.
- HTTPError – It is raised for the exotic HTTP errors, such as the authentication request errors. It is a subclass or URLError. Typical errors include '404' (page not found), '403' (request forbidden),
  and '401' (authentication required).

```
# URL Error
import urllib.request
import urllib.parse
# trying to read the URL but with no internet connectivity
try:
    x = urllib.request.urlopen('https://www.google.com')
    print(x.read())
# Catching the exception generated
except Exception as e :
    print(str(e))
```

```
# HTTP Error
import urllib.request
import urllib.parse
# trying to read the URL
try:
    x = urllib.request.urlopen('https://www.google.com / search?q = test')
    print(x.read())
# Catching the exception generated
except Exception as e :
    print(str(e))
```

urllib.robotparser

This module contains a single class, RobotFileParser. This class answers question about whether or not a particular user can fetch a URL that published robot.txt files. ***Robots.txt*** is a text file webmasters create to instruct web robots how to crawl pages on their website. The robot.txt file tells the web scraper about what parts of the server should not be accessed.

```
# importing robot parser class
import urllib.robotparser as rb
bot = rb.RobotFileParser( )
# checks where the website's robot.txt file reside
x = bot.set_url('https://www.geeksforgeeks.org / robot.txt')
print(x)
# reads the files
y = bot.read()
print(y)
```

```
# we can crawl the main site
z = bot.can_fetch('*', 'https://www.geeksforgeeks.org/')
print(z)

# but can not crawl the disallowed url
w = bot.can_fetch('*', 'https://www.geeksforgeeks.org / wp-admin/')
print(w)
```