26] Process of copying, moving & renaming files.

a) Copying files :- This fucn is used to copy files from source to destin.

Syntax:- import shutil
shutil.copy (src, dest)

ex:-

```
import shutil
src-file = 'path / to / source / file.txt'
dest-dir= 'path / to / destination / ~~new-file.txt~~'
shutil.copy (src-file , dest-dir)
```

b) Moving file:- used to move files from source path to destn

Syntax :- import shutil
shutil.move (src , dest)

ex:-

```
import shutil
src-file = 'path / to / source / file.txt'
dest-dir = 'path / to / destination / newfile.txt'
```

c) Renaming file :- use to rename the existing file use os.rename function

Syntax:- import os
os.rename (src, dest)

ex:-

```
import os
src = 'path / to / source / old-file.txt'
dest = 'path / to / source / new-file.txt'
```

27) explain reading, creating adding & extract zip file.

ⓐ Creating zip file :- to create zip file use the zipfile. Zipfile () class & use write method

Syntax :-
```
import zipfile
with zipfile.zipfile("zipfile-path", "w") as z:
    z. write (file-to-add)
```
ex :-
```
import zipfile
with zipfile.Zipfile("Example. zip", "w") as z:
    z. write ("file1.txt")
```

ⓑ Adding ~~zip~~ file :- to add files to zip files.
you can use zipfile. zipfile () class use write method

Syntax :-
```
import zipfile
with zipfile. Zipfile ("zipfile-path", "a") as z:
    z. write ("file_to-add")
```
ex:-
```
import zipfile
with zipfile. zipfile("example.zip", "a") as z:
    z. write ("file2.txt")
```

ⓒ Reading a file :- use read method to read files.

Syntax :-
```
import zipfile
with zipfile. Zipfile ("zipfile-path", "r") as z:
data = z. read (file-to-read)
```

ⓓ extracting zip file :- use to extract content of zip file by using extractall () method.

Syntax :-
```
import zipfile
with. zipfile. Zipfile ("zip file-path", "r") as z:
    z. extractall ("path")
```

28) diff. modules used to scrape webpages ?

ⓐ Requests :- It's python library for making HTTP requests used to fetch data from websites

Syntax :- import requests
response = requests.get (url)

ex :- import requests
url = 'https://example.com"
r = requests.get (url)
print (r)

ⓑ Beautiful Soup :- for parsing HTML & XML documts.

Syntax :- ~~import~~ from bs4 import BeautifulSoup
soup = BeautifulSoup (html-content, 'html.parser')

ex :- from bs4 import BeautifulSoup
content = 'https://example.html"
soup = BeautifulSoup (content, 'html-parser")
print (soup.p. text)

ⓒ Selenium :- for automating browser & interacting w̄ dynamic content.

Syntax :- from selenium import webdriver
driver = webdriver.chrome ()
driver.get (url)

ex :- from Selenium import webdriver
~~dri~~ url = 'https://ex.com'
d = webdriver. Chrome ()
d. get (url)
print (d. page_source)

(d) Scrapy :- It's python framework for web scraping. provides high level API for crawling websites & extracting data.

Syntax :- Create a scrapy spider class.

ex :-
```
import scrapy
class MYSpider (scrapy.spider)
name = 'example_spider'
url = ['https:// ex.com']
def parse (self, response) :
    pass
```

(e) urllib :- Is python library for fetching data from URLS. it can be used to download files, images from web

Syntax :-
```
from urllib.request import urlopen
r = urlopen (url)
```
ex :-
```
from urllib.request import urlopen
url = "https:// ex.com"
r = urlopen (url)
print (r.read().decode ('utf-8'))
```

(f) mechanicalSoup

(g) pandas

(h) LXML

(29) parse HTML ω BeautifulSoup

* parsing HTML :-

(a) import module

```
from bs4 import Beautifulsoup
```

(b) Create BeautifulSoup object :-

```
r = 'https://ex.com'
soup = Beautifulsoup(r, 'html-parser')
```

(c) navigate & search HTML documt..

* Tag access :

```
tag = soup.p        // access first tag
```

* Attribute access

```
attr-value = tag ['attr-name']    // Access value
```

* Navigating parse tree :-

```
parent-tag = tag.parent       // Access parent tag
next-sibling = tag.next-sibling    // Access next
```

* Searching for tags :-

```
a = soup.find_all ('p')      // fill all <p> tags
```

(d) extract info² :-

```
print (soup.p.text)
```

Example :-
```
from bs4 import Beautifulsoup
Content = 'https:// html.com'
Soup = Beautifulsoup (content, 'html.parser')
title = soup.h1.text        // extract info²
List = [li.text for li in soup.find_all ('li')]
print (f 'title : {title}')
Print ('List Items :', list-items)
```

30. Explain controlling the browser with the Selenium module.

Selenium is a powerful tool for controlling a web browser through the program.

- ➔ **It is commonly used for web scraping, automated testing of web applications, and various other tasks that involve browser automation. The `selenium` module provides a convenient API for interacting with web browsers.**

Below are the basic steps and some examples to get you started:

Basic Steps:

1. **Import the Selenium module:**
   from selenium import webdriver

2. **Create a WebDriver instance:**
   Selenium supports different web browsers like Chrome, Firefox, Safari, etc. You need to download the appropriate WebDriver executable and provide its path.
   ➔ **driver = webdriver.Chrome('path/to/chromedriver.exe')**

3. **Navigate to a URL:**
   ➔ **driver.get('https://www.example.com')**

4. **Interact with elements on the page:**
   ➔ **Find element by ID:**
   element = driver.find_element_by_id('element_id')
   ➔ **Find element by name:**
   element = driver.find_element_by_name('element_name')
   ➔ **Find element by class name:**
   element = driver.find_element_by_class_name('element_class')
   ➔ **Find element by XPath:**
   element = driver.find_element_by_xpath('//path/to/element')
   ➔ **Click on an element:**
   element.click()
   ➔ **Type text into an input field:**
   element.send_keys('Text to type')

5. **Perform browser actions:**
   ➔ **Navigate back:**
   driver.back()
   ➔ **Refresh the page:**
   driver.refresh()
   ➔ **Close the browser:**
   driver.quit()

Example:

```
from selenium import webdriver
 Create a WebDriver instance (Chrome in this example)
driver = webdriver.Chrome('path/to/chromedriver.exe')
 Navigate to Google
driver.get('https://www.google.com')
```

```
 Find the search input field by name
search_box = driver.find_element_by_name('q')

 Type 'Selenium' into the search box
search_box.send_keys('Selenium')

 Submit the search form
search_box.submit()

 Print the titles of the search results
results = driver.find_elements_by_css_selector('h3')
for result in results:
    print(result.text)

 Close the browser
driver.quit()
```

31. Explain the process of reading Excel documents with example.

```
import pandas as pd

// Specify the path to your Excel file
excel_file_path = 'path/to/your/excel_file.xlsx'

 //Read Excel file into a DataFrame
df = pd.read_excel(excel_file_path)

 //Display the entire DataFrame
print("DataFrame from Excel:")
print(df)

 //Display basic information about the DataFrame
print("\nDataFrame Information:")
print(df.info())

 //Display basic statistics about the DataFrame
print("\nDataFrame Statistics:")
print(df.describe())

 //Access specific column
print("\nAccessing specific columns:")
print(df['ColumnName'])   Replace 'ColumnName' with the actual column name

 //Access specific row using iloc (index-based)
print("\nAccessing specific rows:")
print(df.iloc[0])   Access the first row

 //Access specific rows using loc (label-based)
print("\nAccessing specific rows by label:")
print(df.loc[df['ColumnName'] == 'SpecificValue'])   Replace 'ColumnName' and
'SpecificValue' as needed
```

32. How do you extract text from pdf and word documents using python.
To extract text from pdf use PyPDF2 library for pdf files and python-docu for word files.

- ➔ **Extract text from pdf using PyPDF2:-**

```
import PyPDF2
def extract_text_from_pdf(pdf_file_path):
   with open(pdf_file_path, 'rb') as file:
       //Create a PDF reader object
      pdf_reader = PyPDF2.PdfFileReader(file)

     //  Get the total number of pages
      num_pages = pdf_reader.numPages

      // Extract text from each page
      text = ''
      for page_num in range(num_pages):
         page = pdf_reader.getPage(page_num)
         text += page.extractText()
      return text

// Specify the path to your PDF file
pdf_file_path = 'path/to/your/pdf_file.pdf'

// Extract text from the PDF
pdf_text = extract_text_from_pdf(pdf_file_path)

// Display the extracted text
print(pdf_text)
```

- ➔ **Extract text from word document using python-docu:-**

```
from docx import Document
def extract_text_from_word(docx_file_path):
   doc = Document(docx_file_path)

   // Extract text from paragraphs
   text = ''
   for paragraph in doc.paragraphs:
      text += paragraph.text + '\n'
   return text

// Specify the path to your Word document
docx_file_path = 'path/to/your/word_file.docx'

// Extract text from the Word document
word_text = extract_text_from_word(docx_file_path)

// Display the extracted text
print(word_text)
```

33. What are the procedures for Connecting to an SMTP Server?

Connecting to an SMTP (Simple Mail Transfer Protocol) server is a common task when working with sending emails programmatically. Below are the general procedures for connecting to an SMTP server using Python. The example uses the built-in `smtplib` library.

Procedure for Connecting to an SMTP Server:

1. Import the necessary libraries:

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
```

2. Set up your email and server details:

```
sender_email = "your_email@gmail.com"
receiver_email = "recipient_email@example.com"
subject = "Subject of the email"
body = "Body of the email"
smtp_server = "smtp.gmail.com"   Update with your SMTP server address
smtp_port = 587   Update with the appropriate port for your SMTP server
username = "your_email@gmail.com"
password = "your_email_password"
```

3. Create a connection to the SMTP server:

```
server = smtplib.SMTP(smtp_server, smtp_port)
server.starttls()   Use TLS (Transport Layer Security) for secure connection
```

4. Log in to your email account:

```
server.login(username, password)
```

5. Compose the email:

```
message = MIMEMultipart()
message["From"] = sender_email
message["To"] = receiver_email
message["Subject"] = subject
message.attach(MIMEText(body, "plain"))
```

6. Send the email:

```
server.sendmail(sender_email, receiver_email, message.as_string())
```

7. Close the connection:

```
server.quit()
```

Full Example:

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
```

```
// Set up your email and server details
sender_email = "your_email@gmail.com"
receiver_email = "recipient_email@example.com"
subject = "Subject of the email"
body = "Body of the email"

//smtp_server = "smtp.gmail.com"
smtp_port = 587
username = "your_email@gmail.com"
password = "your_email_password"

 //Create a connection to the SMTP server
server = smtplib.SMTP(smtp_server, smtp_port)
server.starttls()   Use TLS (Transport Layer Security) for secure connection

// Log in to your email account
server.login(username, password)

// Compose the email
message = MIMEMultipart()
message["From"] = sender_email
message["To"] = receiver_email
message["Subject"] = subject
message.attach(MIMEText(body, "plain"))

 //Send the email
server.sendmail(sender_email, receiver_email, message.as_string())

// Close the connection
server.quit()
```

35. How do you manipulate Images with Pillow.

To manipulate images in Python, the `Pillow` library is commonly used.

-> `Pillow` is an updated fork of the Python Imaging Library (PIL) and provides a rich set of features for working with images.

➔ **<u>Below are some common tasks you can perform using Pillow:</u>**

Example Image Manipulation Tasks:

```
        from PIL import Image, ImageDraw, ImageFilter

# Open an image file

        image_path = "path/to/your/image.jpg"
        original_image = Image.open(image_path)

# Display the original image

        original_image.show()
```

```
# Resize the image

        resized_image = original_image.resize((300, 200))
        resized_image.show()
# Crop a region from the image

        box = (100, 100, 400, 300)  # (left, top, right, bottom)
        cropped_image = original_image.crop(box)
        cropped_image.show()
# Rotate the image

        rotated_image = original_image.rotate(45)
        rotated_image.show()
# Add text to the image

        draw = ImageDraw.Draw(original_image)
        font_size = 30
        font_path = "path/to/your/font.ttf"
        text_position = (50, 50)
        text_color = (255, 255, 255)
        text_content = "Hello, Pillow!"
        font = ImageFont.truetype(font_path, font_size)
        draw.text(text_position, text_content, fill=text_color, font=font)
        original_image.show()
# Apply a blur filter

        blurred_image = original_image.filter(ImageFilter.BLUR)
        blurred_image.show()
# Save the manipulated image

        resized_image.save("path/to/your/resized_image.jpg")
        cropped_image.save("path/to/your/cropped_image.jpg")
        rotated_image.save("path/to/your/rotated_image.jpg")
        original_image.save("path/to/your/modified_image.jpg")
# Close the images

        original_image.close()
        resized_image.close()
        cropped_image.close()
        rotated_image.close()
        blurred_image.close()
```