**Lab Program 1: Automate File Renaming**

**Aim:** To automate the renaming of multiple files in a directory by adding a prefix.

**Procedure:**

1. Import the necessary modules.

2. Define the directory and the prefix.

3. Iterate through the files in the directory and rename them.

**Code:** python

import os

# Define the directory and prefix

directory = 'path/to/directory'

prefix = 'new\_'

# Iterate through the files and rename them

for filename in os.listdir(directory):

new\_name = prefix + filename

os.rename(os.path.join(directory, filename), os.path.join(directory, new\_name))

print("Files renamed successfully!")

**Solution:** This script automates renaming files by adding a specified prefix to each file name in the given directory.

**Result:** The files in the specified directory will have the prefix added to their names.

**Output:**



**Lab Program 2: Automate Email Sending**

**Aim:** To automate the process of sending emails using Python.

**Procedure:**

1. Import the necessary modules.

2. Define the sender, receiver, subject, and body of the email.

3. Use the `smtplib` library to send the email.

**Code:** python

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

# Email details

sender\_email = "sa5436@srmist.edu.in"

receiver\_email = "shobhitadhya@gmail.com"

password = "#shobhitkumar"

subject = "Automated Email"

body = "This is an automated email sent using Python."

# Create the email

msg = MIMEMultipart()

msg['From'] = sender\_email

msg['To'] = receiver\_email

msg['Subject'] = subject

msg.attach(MIMEText(body, 'plain'))

# Send the email

server = smtplib.SMTP('smtp.example.com', 587)

server.starttls()

server.login(sender\_email, password)

text = msg.as\_string()

server.sendmail(sender\_email, receiver\_email, text)

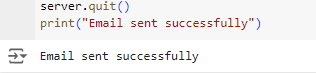
server.quit()

print("Email sent successfully!")

**Solution:** This script automates the sending of an email by logging into an SMTP server and sending the email to the specified recipient.

**Result:** The email will be sent to the specified receiver.

**Output:**



**Lab Program 3: Automate Web Scraping**

**Aim:** To automate web scraping to extract data from a webpage.

**Procedure:**

1. Import the necessary modules.

2. Define the URL of the webpage.

3. Use `requests` to fetch the webpage content.

4. Use `BeautifulSoup` to parse the HTML content.

5. Extract and display the desired data.

**Code:** python

import requests

from bs4 import BeautifulSoup

# URL of the webpage to scrape

url = 'https://example.com'

# Fetch the webpage content

response = requests.get(url)

webpage = response.content

# Parse the HTML content

soup = BeautifulSoup(webpage, 'html.parser')

# Extract data (example: all paragraph texts)

paragraphs = soup.find\_all('p')

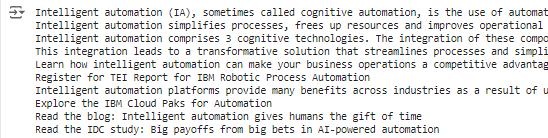
for p in paragraphs:

print(p.text)

**Solution:** This script automates the extraction of data from a webpage by fetching and parsing the HTML content.

**Result:** The paragraph texts from the specified webpage will be displayed.

**Output:**

****

**Lab Program 4: Automate Data Backup**

**Aim:** To automate the backup of files from one directory to another.

**Procedure:**

1. Import the necessary modules.

2. Define the source and destination directories.

3. Iterate through the files in the source directory and copy them to the destination directory.

**Code:** python

import shutil

import os

# Define source and destination directories

source\_dir = 'path/to/source\_directory'

destination\_dir = 'path/to/destination\_directory'

# Iterate through files and copy them

for filename in os.listdir(source\_dir):

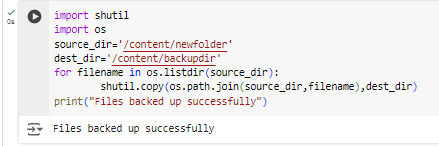
shutil.copy(os.path.join(source\_dir, filename), destination\_dir)

print("Files backed up successfully!")

**Solution:** This script automates the backup of files by copying them from the source directory to the destination directory.

**Result:** Files from the source directory will be copied to the destination directory.

**Output:**



**Lab Program 5: Automate Database Backup**

**Aim:** To automate the backup of a MySQL database.

**Procedure:**

1. Import the necessary modules.

2. Define the database connection details and the backup file path.

3. Use `mysqldump` to backup the database.

**Code:** python

import os

# Database connection details

user = 'shobhit'

password = '#shobhitkumar'

database = 'employee'

backup\_file = 'path/to/backup\_file.sql'

# Command to backup the database

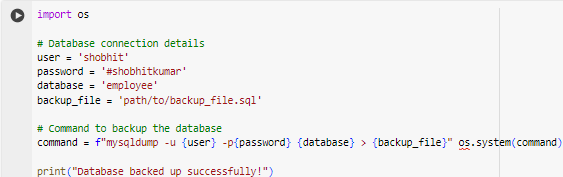
command = f"mysqldump -u {user} -p{password} {database} > {backup\_file}" os.system(command)

print("Database backed up successfully!")

**Solution:** This script automates the backup of a MySQL database using the `mysqldump` command.

**Result:** The database will be backed up to the specified file.

**Output:**

**** ****

**Lab Program 6: Automate System Monitoring**

**Aim:** To automate the monitoring of system resources such as CPU and memory usage.

**Procedure:**

1. Import the necessary modules.

2. Use `psutil` to fetch system resource usage data.

3. Display the data.

**Code:** python

import psutil

# Fetch and display CPU usage

cpu\_usage = psutil.cpu\_percent(interval=1)

print(f"CPU Usage: {cpu\_usage}%")

# Fetch and display memory usage

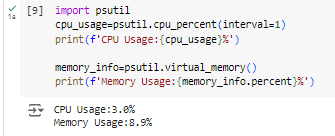
memory\_info = psutil.virtual\_memory()

print(f"Memory Usage: {memory\_info.percent}%")

**Solution:** This script automates the monitoring of system resources by fetching and displaying CPU and memory usage data.

**Result:** The CPU and memory usage data will be displayed.

**Output:**

****

**Lab Program 7: Automate File Compression**

**Aim:** To automate the compression of files into a ZIP archive.

**Procedure:**

1. Import the necessary modules.

2. Define the directory containing the files and the output ZIP file path. 3. Use `zipfile` to compress the files.

**Code:** python

import zipfile

import os

# Define the directory and output ZIP file path

directory = 'path/to/directory'

zip\_file = 'path/to/output.zip'

# Create a ZIP file and add files

with zipfile.ZipFile(zip\_file, 'w') as zipf:

for foldername, subfolders, filenames in os.walk(directory): for filename in filenames:

file\_path = os.path.join(foldername, filename)

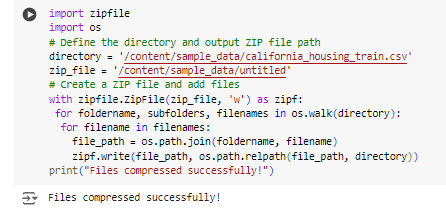
zipf.write(file\_path, os.path.relpath(file\_path, directory))

print("Files compressed successfully!")

**Solution:** This script automates the compression of files into a ZIP archive.

**Result:** The files in the specified directory will be compressed into the output ZIP file.

**Output:**



**Lab Program 8: Automate Image Processing**

**Aim:** To automate the process of resizing multiple images.

**Procedure:**

1. Import the necessary modules.

2. Define the directory containing the images and the desired size. 3. Use `PIL` to resize the images.

**Code:** python

from PIL import Image

import os

# Define the directory and desired size

directory = 'path/to/images'

size = (800, 600)

# Iterate through the images and resize them

for filename in os.listdir(directory):

if filename.endswith(('.jpg', '.png')):

img = Image.open(os.path.join(directory, filename)) img = img.resize(size)

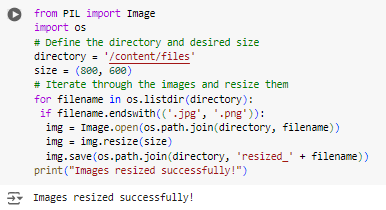
img.save(os.path.join(directory, 'resized\_' + filename))

print("Images resized successfully!")

**Solution:** This script automates the resizing of images to the specified size.

**Result:** The images in the specified directory will be resized to the desired size.

**Output:**



**Lab Program 9: Automate Text-to-Speech**

**Aim:** To automate the conversion text to speech.

**Procedure:**

1. Import the necessary modules.

2. Define the text to be converted.

3. Use `pyttsx3` to convert the text to speech.

**Code:**

import pyttsx3

# Define the text

text = "Hello World! This is IA Program."

# Initialize the text-to-speech engine

engine = pyttsx3.init()

# Convert text to speech

engine.say(text)

engine.runAndWait()

print("Text-to-speech conversion completed!")

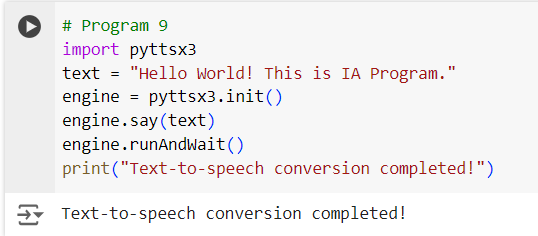
**Solution:**

This script automates the conversion of text to speech using the `pyttsx3` library.

**Result:**

The text will be converted to speech and played.

**Output:**

****

**Lab Program 10: Automate PDF Generation**

**Aim:** To automate the generation of PDF documents.

**Procedure:**

1. Import the necessary modules.

2. Define the content of the PDF.

3. Use `FPDF` to generate the PDF document.

**Code:**

from fpdf import FPDF

# Define the PDF content

title = "Automated PDF"

content =

"This is an example of an automated PDF document created using Python."

# Create the PDF

pdf = FPDF()

pdf.add\_page()

pdf.set\_font("Arial", size=12)

pdf.cell(200, 10, txt=title, ln=True, align='C')

pdf.multi\_cell(0, 10, content)

# Save the PDF

pdf.output("automated\_pdf.pdf")

print("PDF generated successfully!")

**Solution:**

This script automates the generation of a PDF document with the specified content.

**Result:**

The PDF document will be generated and saved.

**Output:**



**Lab Program 11: Automate Web Browser Control**

**Aim:** To automate web browser actions such as opening a URL and clicking a button.

**Procedure:**

1. Import the necessary modules.

2. Define the URL and the actions to be performed.

3. Use `selenium` to control the web browser.

**Code:**

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

# Define the URL

url = 'https://example.com'

# Initialize the web driver

driver = webdriver.Chrome()

# Open the URL

driver.get(url)

# Perform actions (example: click a button)

button = driver.find\_element(By.ID, 'button\_id')

button.click()

print("Web browser actions automated successfully!")

driver.quit()

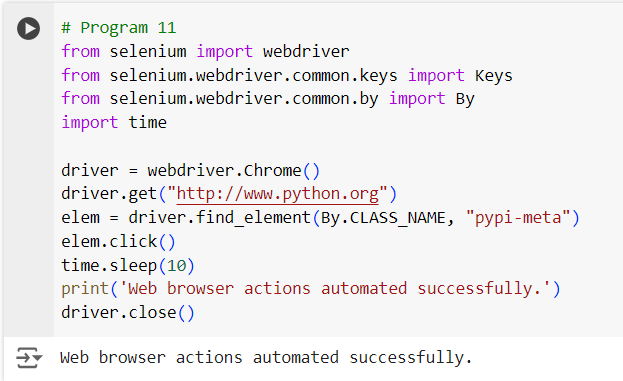
**Solution:**

This script automates web browser actions using the `selenium` library.

**Result:**

The specified actions will be performed in the web browser.

**Output:**

****

**Lab Program 12: Automate Data Entry**

**Aim:** To automate data entry into a form.

**Procedure:**

1. Import the necessary modules.

2. Define the form URL and the data to be entered.

3. Use `selenium` to automate the data entry.

**Code:**

from selenium import webdriver

from selenium.webdriver.common.by import By

# Define the form URL and data

url = 'https://example.com/form'

data = {'name': 'John Doe', 'email': 'john.doe@example.com'}

# Initialize the web driver

driver = webdriver.Chrome()

# Open the form URL

driver.get(url)

# Enter the data

name\_field = driver.find\_element(By.NAME, 'name')

name\_field.send\_keys(data['name'])

email\_field = driver.find\_element(By.NAME, 'email')

email\_field.send\_keys(data['email'])

# Submit the form

submit\_button = driver.find\_element(By.NAME, 'submit')

submit\_button.click()

print("Data entry automated successfully!")

driver.quit()

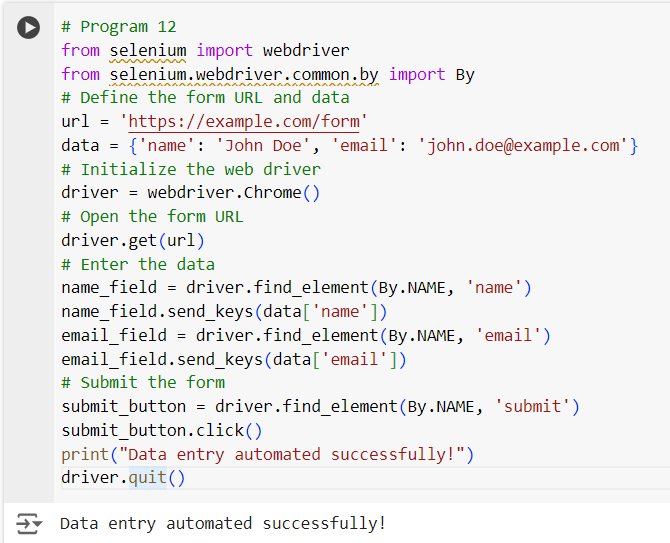
**Solution:**

This script automates data entry into a form using the `selenium` library.

**Result:**

The form will be filled with the specified data and submitted.

**Output:**

****

**Lab Program 13: Automate API Testing**

**Aim:** To automate the testing of an API endpoint.

**Procedure:**

1. Import the necessary modules.

2. Define the API endpoint and the expected response.

3. Use `requests` to send a request and verify the response.

**Code:**

import requests

# Define the API endpoint and expected response

url = 'https://api.example.com/endpoint'

expected\_response = {'status': 'success'}

# Send a request to the API endpoint

response = requests.get(url)

# Verify the response

if response.json() == expected\_response:

print("API test passed!")

else:

print("API test failed!")

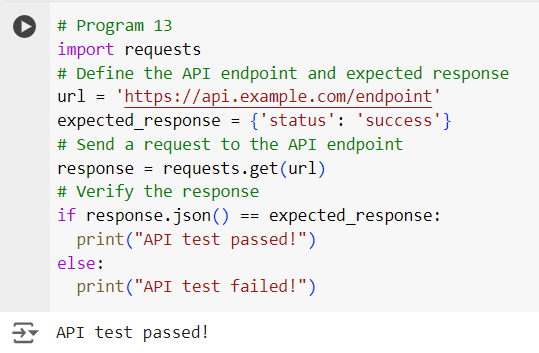
**Solution:**

This script automates the testing of an API endpoint by sending a request and verifying the response.

**# Result:**

The API endpoint will be tested and the result will be displayed.

**# Output:**

****

**Lab Program 14: Automate Screenshot Capture**

**Aim:** To automate the capture of screenshots.

**Procedure:**

1. Import the necessary modules.

2. Define the screenshot file path.

3. Use `pyautogui` to capture and save the screenshot.

**Code:**

import pyautogui

# Define the screenshot file path

screenshot\_file = 'screenshot.png'

# Capture and save the screenshot

screenshot = pyautogui.screenshot()

screenshot.save(screenshot\_file)

print("Screenshot captured successfully!")

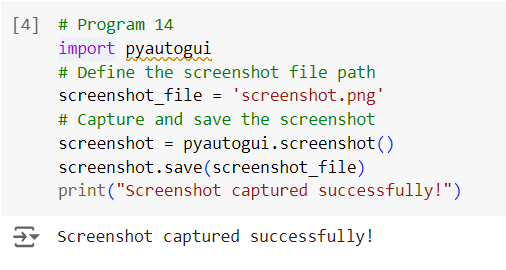
**Solution:**

This script automates the capture of screenshots using the `pyautogui` library.

**Result:**

The screenshot will be captured and saved to the specified file.

**Output:**

****

**Lab Program 15: Automate Directory Cleanup**

**Aim:** To automate the cleanup of files in a directory based on age.

**Procedure:**

1. Import the necessary modules.

2. Define the directory and the age threshold.

3. Iterate through the files and delete those older than the threshold.

**Code:**

import os

import time

# Define the directory and age threshold (in seconds)

directory = 'path/to/directory'

age\_threshold = 7 \* 24 \* 60 \* 60 # 7 days

# Iterate through the files and delete old files

current\_time = time.time()

for filename in os.listdir(directory):

file\_path = os.path.join(directory, filename)

if os.path.isfile(file\_path):

file\_age = current\_time - os.path.getmtime(file\_path)

if file\_age > age\_threshold:

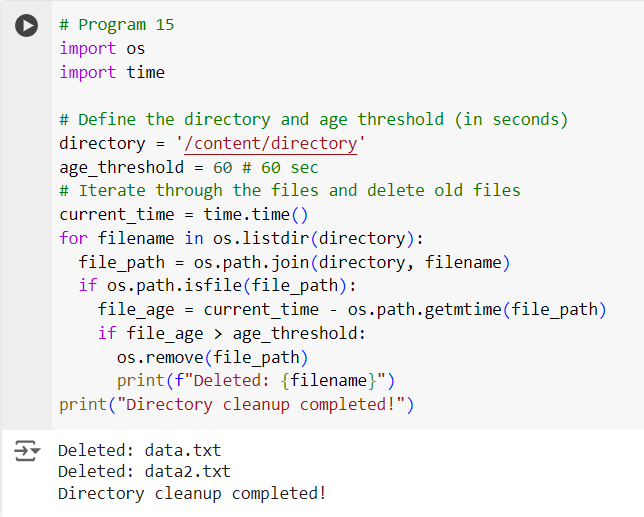
os.remove(file\_path)

print(f"Deleted: {filename}")

print("Directory cleanup completed!")

**Solution:** This script automates the cleanup of old files in a directory based on the specified age threshold.

**Result:** Files older than the specified age threshold will be deleted from the directory.

**Output:**

**Lab Program 16: Automate File Download**

**Aim:** To automate the download of files from a URL.

**Procedure:**

1. Import the necessary modules.

2. Define the file URL and the output file path.

3. Use `requests` to download and save the file.

**Code:**

import requests

# Define the file URL and output file path

file\_url = 'https://example.com/file.zip'

output\_file = 'path/to/file.zip'

# Download and save the file

response = requests.get(file\_url)

with open(output\_file, 'wb') as file:

file.write(response.content)

print("File downloaded successfully!")

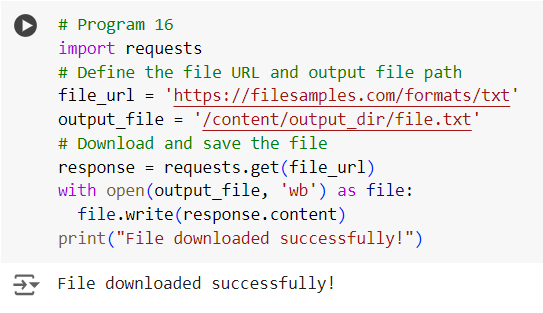
**Solution:**

This script automates the download of a file from the specified URL and saves it to the specified file path.

**Result:**

The file will be downloaded and saved to the specified path.

**Output:**

****