PRODUCTIONALIZE

101 PYTHON AUTOMATION SCRIPTS STREAMLINING TASKS AND BOOSTING PRODUCTIVITY

JOIN NOW FOR MORE FREE GUIDES!

TUSHAR AGGARWAL

JOIN NOW :

PRODUCTIONALIZE.BEEHIIV.COM

MEDIUM.COM/PRODUCTIONALIZE

LINKEDIN GROUP

X.COM/PRODUCTIONALIZE

🐍IMAGINE A WORLD WHERE YOU CAN SIT BACK, RELAX, AND LET PYTHON HANDLE ALL THE MIND-NUMBING, REPETITIVE TASKS THAT EAT UP YOUR PRECIOUS TIME. NO MORE TEDIOUS FILE ORGANIZATION, MANUAL DATA ENTRY, OR GETTING LOST IN THE WEEDS OF COMPLEX WORKFLOWS. WITH THE POWER OF PYTHON AUTOMATION AT YOUR FINGERTIPS, YOU'LL BLAZE THROUGH YOUR TO-DO LIST AT LIGHTNING SPEED, LEAVING MERE MORTALS IN AWE OF YOUR SUPERHUMAN PRODUCTIVITY.🐍

📊WHETHER YOU'RE A BUSY PROFESSIONAL JUGGLING MULTIPLE PROJECTS, A DATA ANALYST DROWNING IN SPREADSHEETS, OR A CURIOUS TINKERER LOOKING TO MAKE YOUR LIFE EASIER, PYTHON IS THE KEY TO UNLOCKING A WHOLE NEW WORLD OF EFFICIENCY AND POSSIBILITY. AND THE BEST PART? YOU DON'T NEED TO BE A CODING GENIUS TO HARNESS ITS POWER!🐍

IN THIS GAME-CHANGING EBOOK, I'VE DISTILLED THE MOST POTENT, TIME-SAVING AUTOMATION TECHNIQUES INTO 101 EASY-TO-FOLLOW SCRIPTS THAT YOU CAN IMPLEMENT TODAY. FROM ORGANIZING YOUR DIGITAL CLUTTER TO EXTRACTING INSIGHTS FROM MASSIVE DATASETS, THESE SCRIPTS WILL TRANSFORM THE WAY YOU WORK AND GIVE YOU BACK THE MOST PRECIOUS RESOURCE OF ALL: YOUR TIME.⏰

JOIN NOW FOR MORE FREE GUIDES!

1. WEB SCRAPING WITH BEAUTIFULSOUP

BeautifulSoup is a Python library that allows you to parse HTML and XML documents. It provides a simple and intuitive way to navigate and search the parsed data using various methods and selectors. BeautifulSoup is particularly useful for extracting data from static web pages.

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
import requests
from bs4 import BeautifulSoup

# Send a GET request to the website
url = 'https://www.example.com/news'
response = requests.get(url)
# Create a BeautifulSoup object and parse the HTML
soup = BeautifulSoup(response.content, 'html.parser')
# Find all the article titles
titles = soup.find_all('h2', class_='article-title')
# Print the titles
for title in titles:
    print(title.text.strip())
```

2. WEB SCRAPING WITH SELENIUM

Selenium, on the other hand, is a powerful tool for automating web browsers. It allows you to interact with web pages, fill out forms, click buttons, and extract data from dynamic websites that heavily rely on JavaScript. Selenium supports multiple web browsers and provides a flexible API for automating web interactions.

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
from selenium import webdriver
from selenium.webdriver.common.by import By

# Create a new instance of the Chrome driver
driver = webdriver.Chrome()
# Navigate to the login page
driver.get('https://www.example.com/login')
# Find the username and password input fields
# and enter the credentials
username_field = driver.find_element(By.ID, 'username')
username_field.send_keys('your_username')
password_field = driver.find_element(By.ID, 'password')
password_field.send_keys('your_password')
# Find and click the login button
login_button = driver.find_element(By.XPATH,
  '//button[@type="submit"]')
login_button.click()
# Close the browser
driver.quit()
```

JOIN NOW FOR MORE FREE GUIDES!

3. TASK AUTOMATION WITH PYAUTOGUI

PyAutoGUI provides a wide range of functions for mouse and keyboard control, including moving the mouse, clicking, dragging, typing text, and pressing keys. It also offers image recognition capabilities, allowing you to locate and interact with specific graphical elements on the screen.PyAutoGUI is particularly useful for automating tasks that involve interacting with desktop applications or performing repetitive actions across multiple software tools.

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
import pyautogui
import time

# Wait for 2 seconds to allow the user
#to switch to the desired window
time.sleep(2)
# Open the text editor (assuming it's
#in the Applications folder on macOS)
pyautogui.press('command')
pyautogui.typewrite('space')
pyautogui.typewrite('textedit')
pyautogui.press('enter')
# Wait for the text editor to open
time.sleep(2)
# Type the message
pyautogui.typewrite('Hello, World!')
# Save the file
pyautogui.hotkey('command', 's')
pyautogui.typewrite('message.txt')
pyautogui.press('enter')
# Close the text editor
pyautogui.hotkey('command', 'q')
```

4. DATA ANALYSIS AUTOMATION WITH PANDAS

With Pandas, you can automate various data analysis tasks, including data cleaning, transformation, filtering, aggregation, and visualization. It integrates well with other Python libraries, such as NumPy for numerical computing and Matplotlib for data visualization.

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
import pandas as pd
# Read the CSV file into a DataFrame
df = pd.read_csv('sales_data.csv')
# Display the first few rows of the DataFrame
print(df.head())
# Calculate total sales by product category
sales_by_category = df.groupby(
  'Category')['Sales'].sum()
print(sales_by_category)
# Filter the DataFrame to include only
#sales above a certain threshold
high_sales = df[df['Sales'] > 1000]
print(high_sales)
# Create a new column based on a condition
df['Discount'] = df['Price'].apply(
  lambda x: 0.1 if x > 50 else 0)
print(df.head())
# Save the modified DataFrame to a new CSV file
df.to_csv('updated_sales_data.csv', index=False)
```

5. TEST AUTOMATION WITH PYTEST

Test automation is crucial for ensuring the quality and reliability of software applications. Python offers several testing frameworks, and one of the most popular and feature-rich frameworks is PyTest.PyTest is a testing framework that makes it easy to write and run tests for Python code.

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
# calculator.py
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b
# test_calculator.py
import pytest
from calculator import add, subtract

def test_add():
    assert add(2, 3) == 5
    assert add(-1, 1) == 0
    assert add(0, 0) == 0

def test_subtract():
    assert subtract(5, 3) == 2
    assert subtract(-1, 1) == -2
    assert subtract(0, 0) == 0

def test_add_invalid_input():
    with pytest.raises(TypeError):
        add('2', 3)

def test_subtract_invalid_input():
    with pytest.raises(TypeError):
        subtract('5', 3)
```

6. FILE ORGANIZATION

Imagine you have a folder filled with various files documents, images, videos, etc. Manually sorting these files into appropriate subfolders can be a tedious task. However, with a Python script, we can automate this process effortlessly.

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
import os
import shutil
# Define the main folder path
main_folder = "/path/to/your/folder"
# Create subfolders for different file types
subfolders = {
    "Documents": [".pdf", ".doc", ".docx", ".txt"],
    "Images": [".jpg", ".jpeg", ".png", ".gif"],
    "Videos": [".mp4", ".avi", ".mov"],
    "Audio": [".mp3", ".wav", ".aac"],
    "Others": []
}
# Iterate over files in the main folder
for filename in os.listdir(main_folder):
    file_path = os.path.join(main_folder, filename)
    # Check if it's a file (not a folder)
    if os.path.isfile(file_path):
        # Get the file extension
        _, extension = os.path.splitext(filename)
        # Determine the appropriate subfolder
        subfolder = "Others"
        for folder, extensions in subfolders.items():
            if extension.lower() in extensions:
                subfolder = folder
                break
        # Create the subfolder if it doesn't exist
        subfolder_path = os.path.join(main_folder,
          subfolder)
        os.makedirs(subfolder_path, exist_ok=True)
        # Move the file to the appropriate subfolder
        destination_path = os.path.join(subfolder_path,
          filename)
        shutil.move(file_path, destination_path)
        print(f"Moved {filename} to {subfolder} folder.")
```

7. SENDING PERSONALIZED EMAILS

Suppose you have a list of recipients and want to send personalized emails to each of them. Here's how you can automate this task using Python:

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
sender_email = 'your_email@example.com'
sender_password = 'your_email_password'
recipients = [
    {'name': 'John Doe', 'email': 'john@example.com'},
    {'name': 'Jane Smith', 'email': 'jane@example.com'},
    {'name': 'Mike Johnson', 'email': 'mike@example.com'}
]
for recipient in recipients:
    message = MIMEMultipart()
    message['From'] = sender_email
    message['To'] = recipient['email']
    message['Subject'] = 'Personalized Email'
    body = f"Dear {recipient['name']},\n
      \nThis is a personalized email just for you!"
    message.attach(MIMEText(body, 'plain'))
    server = smtplib.SMTP('smtp.example.com', 587)
    server.starttls()
    server.login(sender_email, sender_password)
    server.send_message(message)
    server.quit()
```

8. AUTOMATING WEB FORM SUBMISSION

Let's say you want to automate the process of filling out and submitting a web form. Here's how you can achieve this using Selenium:

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
# Create a new instance of the Chrome driver
driver = webdriver.Chrome()
# Navigate to the web form page
driver.get('https://example.com/form')
# Fill out the form fields
name_field = driver.find_element(By.NAME, 'name')
name_field.send_keys('John Doe')
email_field = driver.find_element(By.NAME, 'email')
email_field.send_keys('john@example.com')
country_select = Select(driver.find_element(By.NAME,
  'country'))
country_select.select_by_visible_text('United States')
# Submit the form
submit_button = driver.find_element(By.XPATH,
  '//button[@type="submit"]')
submit_button.click()
# Close the browser
driver.quit()
```

9. SCHEDULING A DAILY TASK

Suppose you want to schedule a Python script to run daily at a specific time. Here's how you can achieve this using the schedule library:

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
import schedule
import time

def daily_task():
    # Code for the daily task goes here
    print("Running daily task...")

schedule.every().day.at("09:00").do(daily_task)

while True:
    schedule.run_pending()
    time.sleep(1)
```

10. UPDATING EXCEL SPREADSHEET DATA

Let's say you have an Excel spreadsheet containing data, and you want to update specific cells based on certain conditions. Here's how you can automate this task using openpyxl:

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
from openpyxl import load_workbook

# Load the Excel workbook
workbook = load_workbook('data.xlsx')
sheet = workbook.active

# Update cell values based on conditions
for row in sheet.iter_rows(min_row=2, values_only=True):
    if row[1] > 100:
        sheet.cell(row=row[0], column=3).value = 'High'
    else:
        sheet.cell(row=row[0], column=3).value = 'Low'

# Save the updated workbook
workbook.save('updated_data.xlsx')
```

11. MERGING PDF FILES

Suppose you have multiple PDF files that you want to merge into a single PDF document. Here's how you can automate this task using PyPDF2:

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
from PyPDF2 import PdfMerger

# Create a PdfMerger object
merger = PdfMerger()

# List of PDF files to merge
pdf_files = ['file1.pdf', 'file2.pdf', 'file3.pdf']

# Append each PDF file to the merger object
for pdf_file in pdf_files:
    merger.append(pdf_file)

# Write the merged PDF to a new file
merger.write("merged.pdf")
merger.close()
```

12. BACKING UP FILES TO A REMOTE SERVER

Suppose you want to automate the process of backing up specific files to a remote server using SSH. Here's how you can achieve this using Python and the paramiko library:

```python
#101 Python Automation Scripts-github.com/tushar2704
#Please adjust according to you requirements.
import os
import paramiko
# SSH connection details
hostname = 'your_remote_server'
username = 'your_username'
password = 'your_password'
# Local directory and files to backup
local_dir = '/path/to/local/directory'
files_to_backup = ['file1.txt', 'file2.txt',
  'file3.txt']
# Remote directory for backups
remote_dir = '/path/to/remote/backup/directory'

# Create an SSH client
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(
  paramiko.AutoAddPolicy())
ssh.connect(hostname, username=username,
  password=password)
# Create an SFTP client
sftp = ssh.open_sftp()
# Backup each file
for file_name in files_to_backup:
    local_path = os.path.join(local_dir, file_name)
    remote_path = os.path.join(remote_dir, file_name)
    sftp.put(local_path, remote_path)
# Close the SFTP and SSH connections
sftp.close()
ssh.close()
```

TUSHAR AGGARWAL

PRODUCTIONALIZE

READY TO BECOME A PYTHON AUTOMATION SUPERHERO? 🦸

JOIN NOW TO RECEIVE FULL 105 PAGES DONE FOR YOU PDF!

JOIN NOW :

PRODUCTIONALIZE.BEEHIIV.COM

MEDIUM.COM/PRODUCTIONALIZE

LINKEDIN GROUP

X.COM/PRODUCTIONALIZE

AMAZING AI PRODUCT DISCOUNT FOR NEWSLETTER SUBSCRIBER COMING UP SOON!

REFER YOUR 25 FRIENDS FOR $25 GIFT CARDS!
PRODUCTIONALIZE.BEEHIIV.COM/SUBSCRIBE?REF=UY5IXFN0OW

Code Source: My Medium Blog here

JOIN NOW FOR MORE FREE GUIDES!