**Topics**

1. **Introduction to Automation**:
   * Automating repetitive tasks in DevOps: file management, job scheduling, log monitoring, and server health checks.
   * Tools and libraries:
     + os for file operations.
     + subprocess for executing system commands.
     + schedule for task scheduling.
2. **Python Scripts for Common Automation Tasks**:
   * Automating system commands.
   * Managing configuration files.
   * Monitoring server health and logs.
3. **Scheduling Tasks**:
   * Using Python libraries (schedule, time) to automate jobs at specific intervals.
4. **Real-Life Automation Use Cases**:
   * Disk space monitoring.
   * Log analysis.
   * Backup creation.

**Practical Exercises**

**1. Automate Disk Space Monitoring**

* Objective: Check disk usage and alert if it exceeds a threshold.
* Script:

python

Copy code

import shutil

threshold = 80 # Percentage

total, used, free = shutil.disk\_usage("/")

used\_percentage = (used / total) \* 100

print(f"Disk Usage: {used\_percentage:.2f}%")

if used\_percentage > threshold:

print("Alert: Disk usage exceeds threshold!")

else:

print("Disk usage is within limits.")

**2. Automate Log Analysis**

* Objective: Find and report critical errors from a log file.
* Script:

python

Copy code

log\_file = "app.log"

critical\_errors = []

with open(log\_file, "r") as file:

for line in file:

if "CRITICAL" in line:

critical\_errors.append(line.strip())

if critical\_errors:

print("Critical Errors Found:")

for error in critical\_errors:

print(error)

else:

print("No critical errors found.")

**3. Automate Command Execution**

* Objective: Automate execution of system commands.
* Script:

python

Copy code

import subprocess

command = ["ls", "-l"]

try:

result = subprocess.run(command, check=True, capture\_output=True, text=True)

print("Command Output:")

print(result.stdout)

except subprocess.CalledProcessError as e:

print(f"Error executing command: {e}")

**4. Schedule a Python Job**

* Objective: Schedule a task to run every minute.
* Install the schedule library:

bash

Copy code

pip install schedule

* Script:

python

Copy code

import schedule

import time

def greet():

print("Hello! This job runs every minute.")

schedule.every(1).minutes.do(greet)

while True:

schedule.run\_pending()

time.sleep(1)

**5. Automate Server Health Monitoring**

* Objective: Check server response times and log results.
* Script:

python

Copy code

import requests

import time

servers = ["https://jsonplaceholder.typicode.com/posts/1",

"https://jsonplaceholder.typicode.com/posts/1000"]

def check\_server\_health():

with open("server\_health.log", "a") as log:

for server in servers:

try:

response = requests.get(server, timeout=5)

status = f"{time.ctime()} - {server} - Status: {response.status\_code}\n"

except requests.exceptions.RequestException as e:

status = f"{time.ctime()} - {server} - Error: {e}\n"

log.write(status)

# Schedule the health check

schedule.every(5).minutes.do(check\_server\_health)

print("Server health monitoring started...")

while True:

schedule.run\_pending()

time.sleep(1)

**6. Automate Backup Creation**

* Objective: Create periodic backups of a directory.
* Script:

python

Copy code

import os

import shutil

import schedule

import time

source\_dir = "/path/to/source"

backup\_dir = "/path/to/backup"

def create\_backup():

backup\_path = os.path.join(backup\_dir, f"backup\_{time.strftime('%Y%m%d%H%M%S')}")

shutil.copytree(source\_dir, backup\_path)

print(f"Backup created: {backup\_path}")

# Schedule the backup job

schedule.every().day.at("02:00").do(create\_backup)

print("Backup scheduling started...")

while True:

schedule.run\_pending()

time.sleep(1)

**7. Automate Configuration File Updates**

* Objective: Modify configuration files automatically.
* Script:

python

Copy code

config\_file = "config.txt"

def update\_config(key, value):

with open(config\_file, "r") as file:

lines = file.readlines()

with open(config\_file, "w") as file:

for line in lines:

if line.startswith(key):

file.write(f"{key}={value}\n")

else:

file.write(line)

update\_config("server\_name", "new\_server")

print("Configuration updated.")

**8. Challenge: Automate Continuous Log Monitoring**

* Objective: Continuously monitor logs and trigger actions for specific patterns.
* Script:

python

Copy code

import time

log\_file = "app.log"

def monitor\_logs():

with open(log\_file, "r") as file:

file.seek(0, os.SEEK\_END) # Move to the end of the file

while True:

line = file.readline()

if line:

if "ERROR" in line:

print(f"Error detected: {line.strip()}")

time.sleep(1)

print("Log monitoring started...")

monitor\_logs()