Choosing two objectives and solve them using Python:

* **Objective 1: System Health Monitoring Script**
* **Objective 2: Automated Backup Solution**

**Objective 1: System Health Monitoring Script**

**Task:** Develop a script that monitors the health of a Linux system. It should check CPU usage, memory usage, disk space, and running processes. If any of these metrics exceed predefined thresholds (e.g., CPU usage > 80%), the script should send an alert to the console or a log file.

import psutil

import logging

# Set up logging

logging.basicConfig(filename='system\_health.log', level=logging.INFO, format='%(asctime)s - %(message)s')

# Define thresholds

CPU\_THRESHOLD = 80.0 # Percentage

MEMORY\_THRESHOLD = 80.0 # Percentage

DISK\_THRESHOLD = 80.0 # Percentage

def check\_cpu\_usage():

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

if cpu\_usage > CPU\_THRESHOLD:

logging.warning(f'High CPU usage detected: {cpu\_usage}%')

return cpu\_usage

def check\_memory\_usage():

memory\_info = psutil.virtual\_memory()

if memory\_info.percent > MEMORY\_THRESHOLD:

logging.warning(f'High memory usage detected: {memory\_info.percent}%')

return memory\_info.percent

def check\_disk\_usage():

disk\_info = psutil.disk\_usage('/')

if disk\_info.percent > DISK\_THRESHOLD:

logging.warning(f'High disk usage detected: {disk\_info.percent}%')

return disk\_info.percent

def check\_running\_processes():

processes = psutil.pids()

logging.info(f'Number of running processes: {len(processes)}')

return len(processes)

if \_\_name\_\_ == "\_\_main\_\_":

cpu = check\_cpu\_usage()

memory = check\_memory\_usage()

disk = check\_disk\_usage()

processes = check\_running\_processes()

print(f'CPU Usage: {cpu}%')

print(f'Memory Usage: {memory}%')

print(f'Disk Usage: {disk}%')

print(f'Running Processes: {processes}')

**Objective 2: Automated Backup Solution**

**Task:** Write a script to automate the backup of a specified directory to a remote server or a cloud storage solution. The script should provide a report on the success or failure of the backup operation.

**Python Script:**

import os

import shutil

import logging

from datetime import datetime

# Set up logging

logging.basicConfig(filename='backup.log', level=logging.INFO, format='%(asctime)s - %(message)s')

# Define the source and backup directory

SOURCE\_DIR = '/path/to/source'

BACKUP\_DIR = '/path/to/backup'

def perform\_backup(source\_dir, backup\_dir):

try:

timestamp = datetime.now().strftime('%Y%m%d\_%H%M%S')

backup\_path = os.path.join(backup\_dir, f'backup\_{timestamp}')

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

logging.info(f'Backup successful: {backup\_path}')

return True

except Exception as e:

logging.error(f'Backup failed: {str(e)}')

return False

if \_\_name\_\_ == "\_\_main\_\_":

if perform\_backup(SOURCE\_DIR, BACKUP\_DIR):

print('Backup completed successfully.')

else:

print('Backup failed.')