Some examples of Python scripts that can be used to resolve different types of incidents:

1. Log Analysis Script:

- Objective: Parse log files to identify errors and anomalies, aiding in troubleshooting incidents.

**```python**

import re

def analyze\_log\_file(log\_file):

error\_count = 0

with open(log\_file, 'r') as f:

for line in f:

if re.search('ERROR', line):

error\_count += 1

# Add further analysis or error handling logic here

return error\_count

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

log\_file\_path = 'path/to/logfile.log'

errors\_found = analyze\_log\_file(log\_file\_path)

print(f"Found {errors\_found} errors in the log file.")

```

2. Service Restart Script:

- Objective: Automatically restart a service upon encountering errors or failures.

**```python**

import subprocess

def restart\_service(service\_name):

try:

subprocess.run(['systemctl', 'restart', service\_name], check=True)

print(f"Service {service\_name} restarted successfully.")

except subprocess.CalledProcessError as e:

print(f"Failed to restart service {service\_name}: {e}")

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

service\_to\_restart = 'service\_name'

restart\_service(service\_to\_restart)

```

3. Database Backup Script:

- Objective: Automate the process of taking backups of databases to prevent data loss during incidents.

**```python**

import os

import subprocess

import datetime

def backup\_database(database\_name, backup\_dir):

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

backup\_file = os.path.join(backup\_dir, f"{database\_name}\_{timestamp}.sql")

try:

subprocess.run(['pg\_dump', '-U', 'username', '-d', database\_name, '-f', backup\_file], check=True)

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

except subprocess.CalledProcessError as e:

print(f"Failed to create backup: {e}")

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

database\_to\_backup = 'database\_name'

backup\_directory = 'path/to/backup'

backup\_database(database\_to\_backup, backup\_directory)

```

4. Notification Script:

- Objective: Send notifications to relevant stakeholders or teams about ongoing incidents or their resolution.

**```python**

import smtplib

from email.mime.text import MIMEText

def send\_notification(subject, body, recipients):

sender\_email = 'your\_email@example.com'

smtp\_server = 'smtp.example.com'

port = 587

msg = MIMEText(body)

msg['Subject'] = subject

msg['From'] = sender\_email

msg['To'] = ', '.join(recipients)

try:

server = smtplib.SMTP(smtp\_server, port)

server.starttls()

server.login(sender\_email, 'your\_password')

server.sendmail(sender\_email, recipients, msg.as\_string())

print("Notification sent successfully.")

except Exception as e:

print(f"Failed to send notification: {e}")

finally:

server.quit()

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

notification\_subject = 'Incident Notification'

notification\_body = 'This is to inform you about the ongoing incident.'

notification\_recipients = ['recipient1@example.com', 'recipient2@example.com']

send\_notification(notification\_subject, notification\_body, notification\_recipients)

```

These scripts provide basic functionality to address common incidents. Customize and extend them as per your specific requirements and environment configurations. Ensure to handle errors gracefully and follow security best practices when implementing incident resolution scripts.