 **GRT INSTITUTE OF**

**ENGINEERING AND**

**TECHNOLOGY, TIRUTTANI - 631209**

**Approved by AICTE, New Delhi Affiliated to Anna University, Chennai**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**PROJECT TITLE**

***Disaster recovery with IBM cloud virtual servers***

**COLLEGE CODE:1103**

**Phase-4**

**Swathy.R**

3rd year, 5th sem

Reg no.:110321104053

**Swathy1883@gmail.com**

**DISASTER RECOVERY WITH IBM CLOUD VIRTUAL SERVERS**

**TESTING RECOVERY PLAN:**

“DisasterRecoveryManager" that simulates a disaster recovery management system. It allows you to switch between different sites (Production Site, DR Site, and Primary Site) and perform actions at those sites. When you run the code, it enters a loop to interact with the user and provide options to switch between sites or exit the program.

* **DisasterRecoveryManager** class:
* It has an **\_\_init\_\_** method that initializes the three site names and sets the **current\_site** to the primary site by default.
* The **switch\_to\_site** method allows you to switch to a different site by changing the **current\_site** attribute.
* The **run\_production**, **run\_dr**, and **run\_primary** methods allow you to change the **current\_site** based on the user's choice and display a message about the current site.
* In the **if \_\_name\_\_ == "\_\_main\_\_":** block:
* An instance of the **DisasterRecoveryManager** class is created.
* A loop is set up to continuously present a menu of options to the user until they choose to exit.

**PROGRAM:**

**class DisasterRecoveryManager:**

**def \_\_init\_\_(self):**

**self.primary\_site = "Primary Site"**

**self.production\_site = "Production Site"**

**self.dr\_site = "DR Site"**

**self.current\_site = self.primary\_site**

**def switch\_to\_site(self, site):**

**print(f"Switching to {site}...")**

**self.current\_site = site**

**def run\_production(self):**

**if self.current\_site == self.production\_site:**

**print("Already in the Production Site.")**

**else:**

**self.switch\_to\_site(self.production\_site)**

**print("Running in the Production Site.")**

**def run\_dr(self):**

**if self.current\_site == self.dr\_site:**

**print("Already in the DR Site.")**

**else:**

**self.switch\_to\_site(self.dr\_site)**

**print("Running in the DR Site.")**

**def run\_primary(self):**

**if self.current\_site == self.primary\_site:**

**print("Already in the Primary Site.")**

**else:**

**self.switch\_to\_site(self.primary\_site)**

**print("Running in the Primary Site.")**

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

**drm = DisasterRecoveryManager()**

**while True:**

**print("\nChoose an option:")**

**print("1. Switch to Production Site")**

**print("2. Switch to DR Site")**

**print("3. Switch to Primary Site")**

**print("4. Exit")**

**choice = input()**

**if choice == "1":**

**drm.run\_production()**

**elif choice == "2":**

**drm.run\_dr()**

**elif choice == "3":**

**drm.run\_primary()**

**elif choice == "4":**

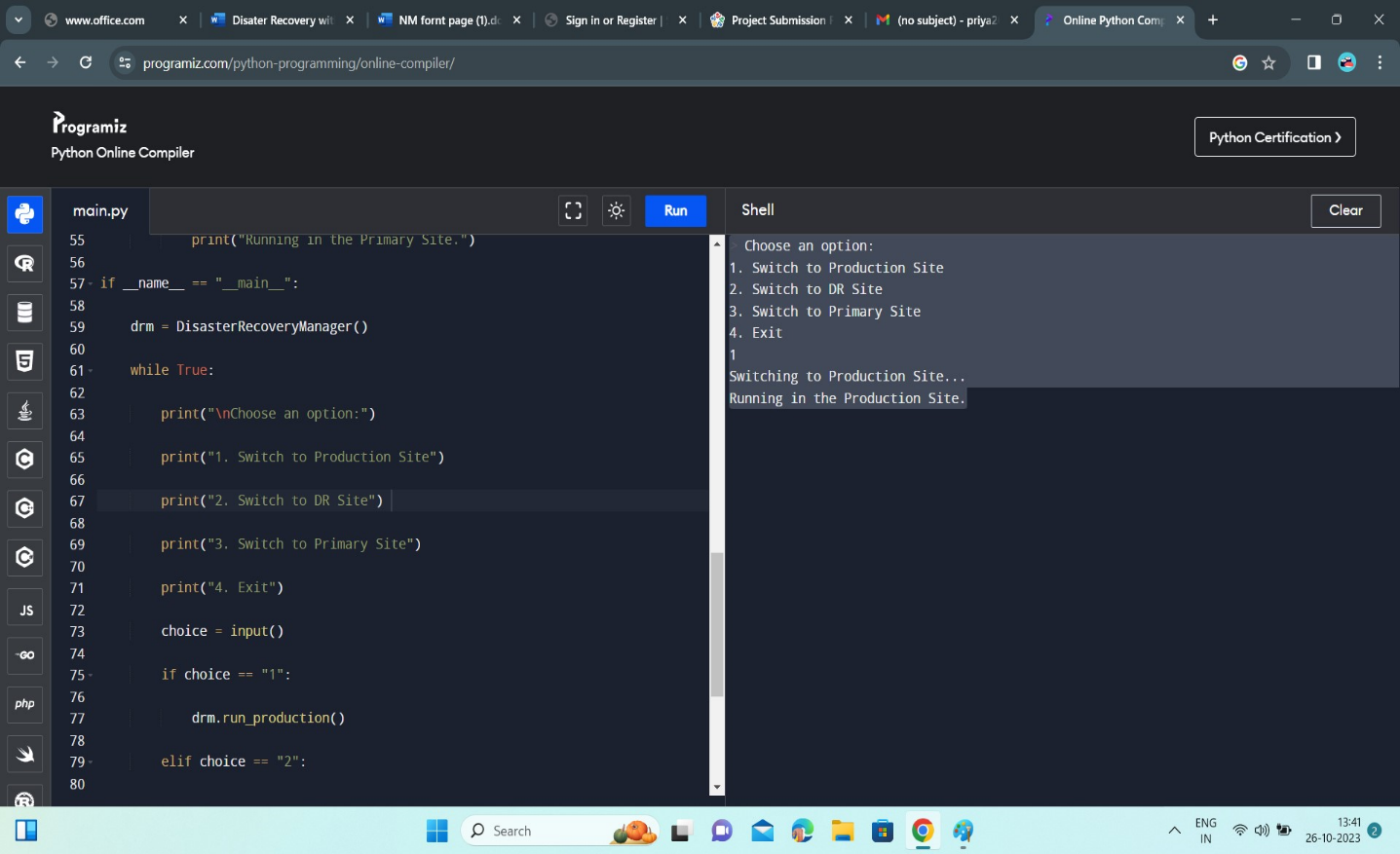
**print("Exiting the disaster recovery manager.")**

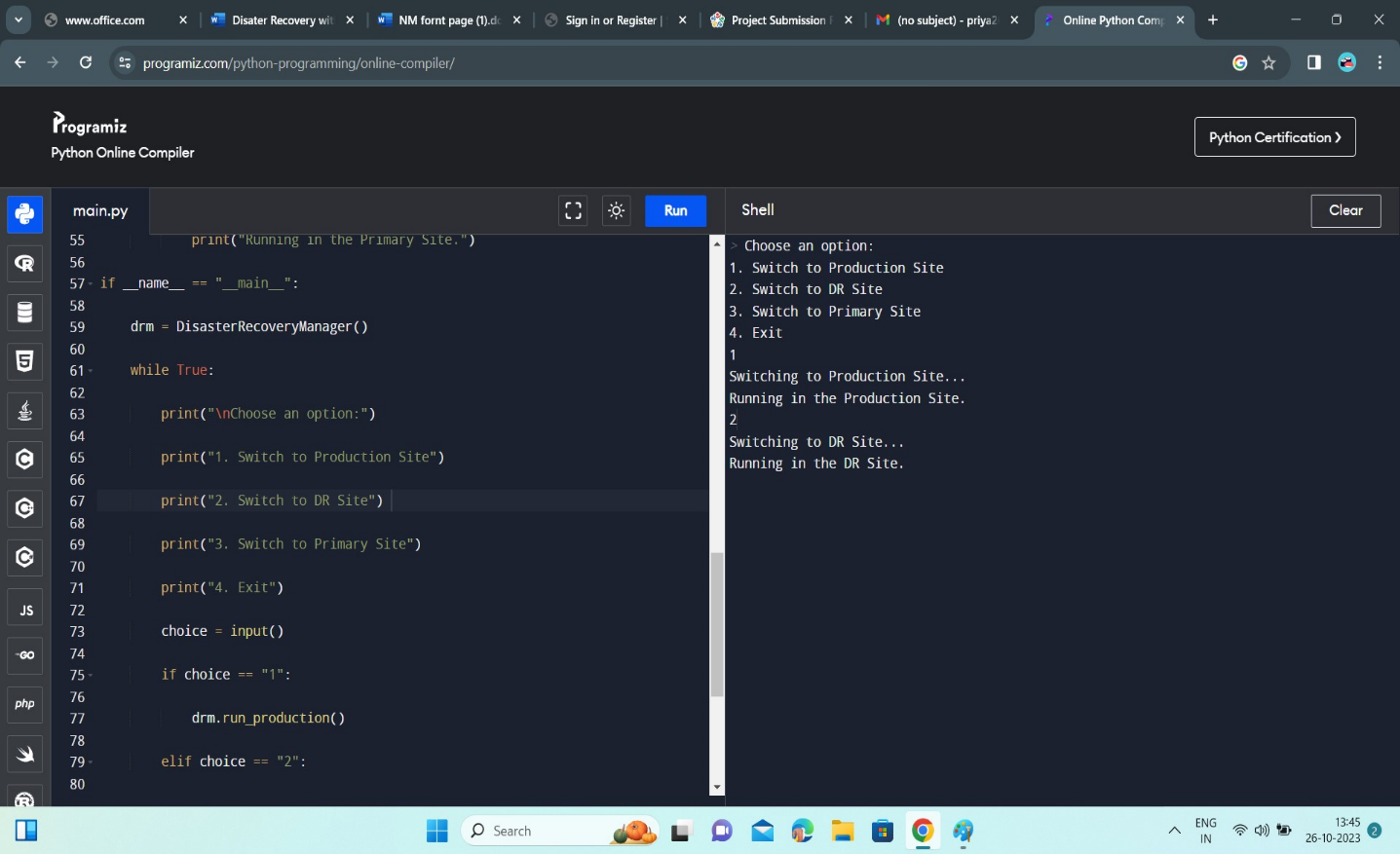
**break**

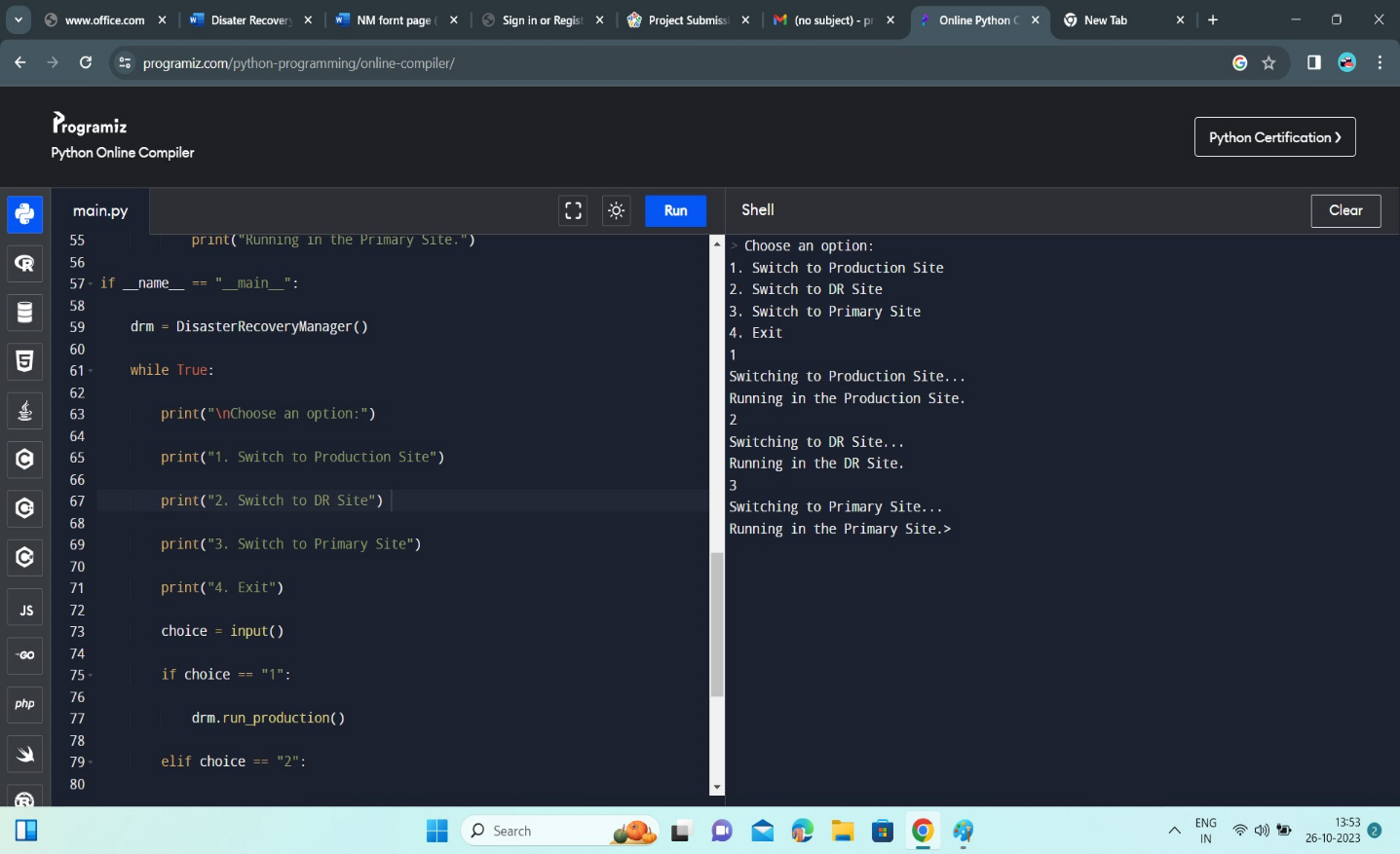
**else:**

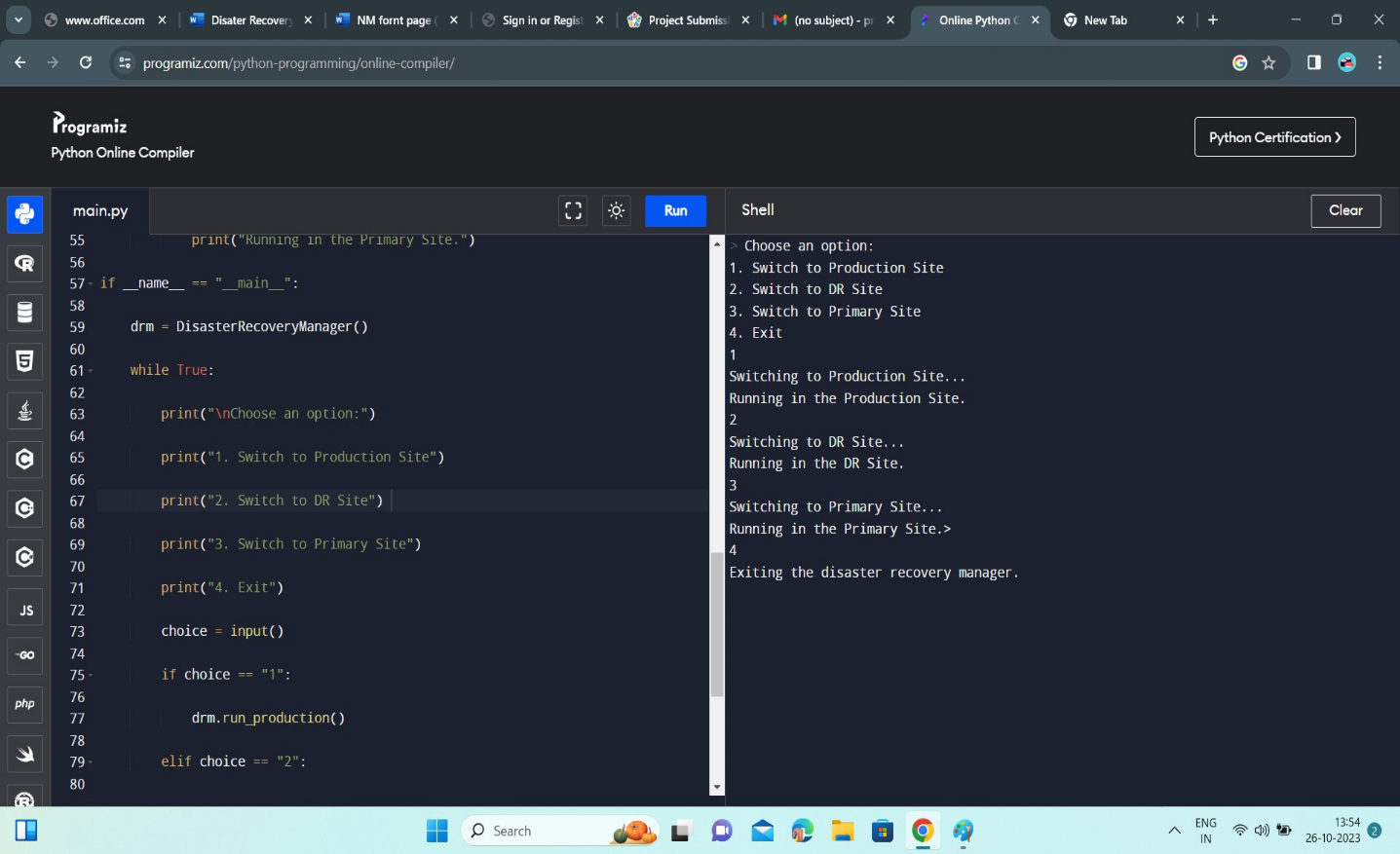
**print("Invalid choice. Please choose a valid option."**

**OUTPUT:**









**BACKUP TESTING:**

This Python script simulates a simple disaster recovery scenario for virtual servers, with the ability to create a backup server and perform a failover operation.

* Import necessary libraries: **random** and **time** are imported, but they are not used in the script.
* Initialize the statuses of primary and secondary servers.
* **create\_backup\_server()**: This function simulates the creation of a backup server. It changes the **secondary\_server\_status** to "running" after a simulated delay of 5 seconds.
* **simulate\_failover()**: This function simulates a failover by stopping the primary server and starting the secondary server. It changes the **primary\_server\_status** to "stopped" and the **secondary\_server\_status** to "running."
* **main()**: This is the main function of the program. It displays the current statuses of the primary and secondary servers and provides a menu for the user to choose actions.
* Inside the **main()** function, a loop is created to keep the program running until the user chooses to exit.
* The menu options are as follows:
* "1" simulates a primary server failure by calling **simulate\_failover()** if the primary server is running.
* "2" creates a backup server by calling **create\_backup\_server()** if the secondary server is stopped.
* "3" exits the program.
* If the user enters an invalid choice, an error message is displayed.

**PROGRAM:**

**import random**

**import time**

**# Simulate the primary and secondary servers**

**primary\_server\_status = "running"**

**secondary\_server\_status = "stopped"**

**def create\_backup\_server():**

**global secondary\_server\_status**

**print("Creating a backup virtual server...")**

**# Simulate the creation of a backup server**

**time.sleep(5)**

**secondary\_server\_status = "running"**

**print("Backup virtual server created and running.")**

**def simulate\_failover():**

**global primary\_server\_status, secondary\_server\_status**

**print("Simulating a failover...")**

**# Simulate primary server failure**

**primary\_server\_status = "stopped"**

**# Simulate secondary server taking over**

**secondary\_server\_status = "running"**

**print("Failover complete.")**

**def main():**

**print("Disaster Recovery Simulation")**

**print("Primary Virtual Server Status:", primary\_server\_status)**

**print("Secondary Virtual Server Status:", secondary\_server\_status)**

**while True:**

**print("\nOptions:")**

**print("1. Simulate Primary Server Failure")**

**print("2. Create Backup Server")**

**print("3. Exit")**

**choice = input("Enter your choice: ")**

**if choice == "1":**

**if primary\_server\_status == "running":**

**simulate\_failover()**

**else:**

**print("Primary server is already stopped.")**

**elif choice == "2":**

**if secondary\_server\_status == "stopped":**

**create\_backup\_server()**

**else:**

**print("Backup server is already running.")**

**elif choice == "3":**

**print("Exiting the program.")**

**break**

**else:**

**print("Invalid choice. Please try again.")**

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

**main()**

**OUTPUT:**

