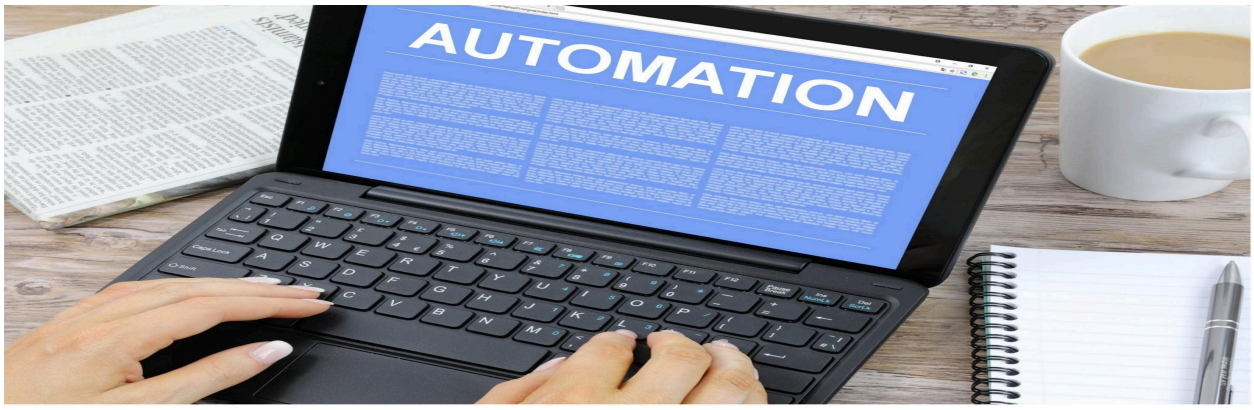

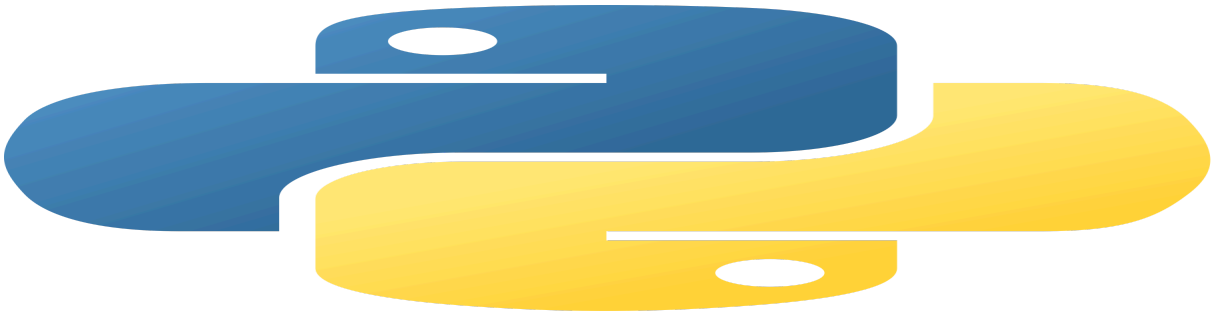

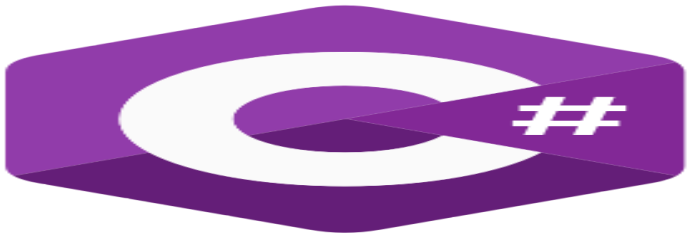

Automation Tutorial (with C# + Python Examples)

What “automation” means

Automation is writing code that runs **repeatable tasks** for you:

- on a schedule (daily/weekly)
- when a file changes

- when a condition is true (e.g., “website is down”)
- in bulk (rename 500 files, archive logs, backup folders)

Tooling you’ll use

Windows (recommended for these examples)

- **Task Scheduler** (run scripts on a schedule)
- **PowerShell** (optional helper)
- **Python 3.x**
- **.NET SDK** (C#)

Part 1: A Real Automation Project

Project: Daily “Cleanup + Backup + Log” Automation

The automation will:

1. Create a timestamped backup (ZIP) of a folder
2. Delete temporary files older than X days
3. Write a log file of what happened

You’ll build it in **Python** and in **C#**.

Part 2: Python Automation Example

A) Python Script: Zip Backup + Cleanup + Logging

Save as: `automation_backup_cleanup.py`

```
import os
import shutil
import zipfile
from datetime import datetime, timedelta
```

```
SOURCE_FOLDER = r"C:\DataToBackup"
BACKUP_FOLDER = r"C:\Backups"
TEMP_FOLDER = r"C:\Temp"
LOG_FILE = r"C:\Backups\automation_log.txt"

DAYS_TO_KEEP_TEMP = 7

def log(msg: str):
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    line = f"[{timestamp}] {msg}"
    print(line)
    os.makedirs(os.path.dirname(LOG_FILE), exist_ok=True)
    with open(LOG_FILE, "a", encoding="utf-8") as f:
        f.write(line + "\n")

def zip_backup(source_dir: str, backup_dir: str):
    os.makedirs(backup_dir, exist_ok=True)
    stamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    zip_name = os.path.join(backup_dir, f"backup_{stamp}.zip")

    log(f"Starting backup: {source_dir} -> {zip_name}")

    with zipfile.ZipFile(zip_name, "w", zipfile.ZIP_DEFLATED) as z:
        for root, _, files in os.walk(source_dir):
            for file in files:
                full_path = os.path.join(root, file)
                rel_path = os.path.relpath(full_path, source_dir)
                z.write(full_path, rel_path)

    log(f"Backup complete: {zip_name}")

def cleanup_old_files(folder: str, days_to_keep: int):
    if not os.path.exists(folder):
        log(f"Temp folder does not exist: {folder}")
        return

    cutoff = datetime.now() - timedelta(days=days_to_keep)
    deleted = 0

    for root, _, files in os.walk(folder):
        for file in files:
            path = os.path.join(root, file)
            try:
```

```
            mtime = datetime.fromtimestamp(os.path.getmtime(path))
            if mtime < cutoff:
                os.remove(path)
                deleted += 1
        except Exception as e:
            log(f"Failed to delete {path}: {e}")

    log(f"Cleanup complete. Deleted {deleted} files older than {days_to_keep} days from {folder}.")

def main():
    try:
        zip_backup(SOURCE_FOLDER, BACKUP_FOLDER)
        cleanup_old_files(TEMP_FOLDER, DAYS_TO_KEEP_TEMP)
        log("Automation run successful.")
    except Exception as e:
        log(f"Automation run FAILED: {e}")

if __name__ == "__main__":
    main()
```

Run it manually

python automation_backup_cleanup.py

Part 3: C# Automation Example (.NET)

A) Create a Console App

dotnet new console -n AutomationTool
cd AutomationTool

Replace `Program.cs` with:

```
using System;
using System.IO;
using System.IO.Compression;
using System.Linq;
```

```
class Program
{
    static readonly string SourceFolder = @"C:\DataToBackup";
    static readonly string BackupFolder = @"C:\Backups";
    static readonly string TempFolder = @"C:\Temp";
    static readonly string LogFile = @"C:\Backups\automation_log.txt";
    static readonly int DaysToKeepTemp = 7;

    static void Main()
    {
        try
        {
            ZipBackup(SourceFolder, BackupFolder);
            CleanupOldFiles(TempFolder, DaysToKeepTemp);
            Log("Automation run successful.");
        }
        catch (Exception ex)
        {
            Log("Automation run FAILED: " + ex.Message);
        }
    }

    static void Log(string message)
    {
        var line = $"[{DateTime.Now:yyyy-MM-dd HH:mm:ss}] {message}";
        Console.WriteLine(line);

        Directory.CreateDirectory(Path.GetDirectoryName(LogFile)!);
        File.AppendAllText(LogFile, line + Environment.NewLine);
    }

    static void ZipBackup(string sourceDir, string backupDir)
    {
        Directory.CreateDirectory(backupDir);

        var stamp = DateTime.Now.ToString("yyyyMMdd_HHmmss");
        var zipPath = Path.Combine(backupDir, $"backup_{stamp}.zip");

        Log($"Starting backup: {sourceDir} -> {zipPath}");

        ZipFile.CreateFromDirectory(sourceDir, zipPath, CompressionLevel.Optimal,
includeBaseDirectory: false);

        Log($"Backup complete: {zipPath}");
```

```
    }

    static void CleanupOldFiles(string folder, int daysToKeep)
    {
        if (!Directory.Exists(folder))
        {
            Log($"Temp folder does not exist: {folder}");
            return;
        }

        var cutoff = DateTime.Now.AddDays(-daysToKeep);
        var files = Directory.EnumerateFiles(folder, "*", SearchOption.AllDirectories);

        int deleted = 0;

        foreach (var file in files)
        {
            try
            {
                var lastWrite = File.GetLastWriteTime(file);
                if (lastWrite < cutoff)
                {
                    File.Delete(file);
                    deleted++;
                }
            }
            catch (Exception ex)
            {
                Log($"Failed to delete {file}: {ex.Message}");
            }
        }

        Log($"Cleanup complete. Deleted {deleted} files older than {daysToKeep} days from
{folder}.");
    }
}
```

Build and run

dotnet build
dotnet run

Part 4: Scheduling the Automation (Windows)

A) Schedule Python with Task Scheduler

1. Open **Task Scheduler**
2. **Create Basic Task**
3. Trigger: **Daily**
4. Action: **Start a Program**
5. Program/script: `python`
6. Add arguments:
   `C:\Path\automation_backup_cleanup.py`

B) Schedule C# .exe with Task Scheduler

1. Build release:

dotnet publish -c Release -r win-x64 --self-contained false

2. Task Scheduler “Start a Program”
3. Point it to:
   `...\AutomationTool\bin\Release\net8.0\win-x64\publish\AutomationTool.exe`

Part 5: Another Automation Pattern (Condition-based)

A) Python: “Check Website and Log Status”

```
import requests
from datetime import datetime

URL = "https://example.com"
LOG = "status_log.txt"
```

```
try:
    r = requests.get(URL, timeout=5)
    status = f"{r.status_code} {r.reason}"
except Exception as e:
    status = f"DOWN ({e})"

line = f"[{datetime.now():%Y-%m-%d %H:%M:%S}] {URL} -> {status}"
print(line)
with open(LOG, "a", encoding="utf-8") as f:
    f.write(line + "\n")
```

B) C#: “Check Website and Log Status”

```
using System;
using System.Net.Http;
using System.IO;
using System.Threading.Tasks;

class Program
{
    static async Task Main()
    {
        string url = "https://example.com";
        string log = "status_log.txt";

        using var client = new HttpClient { Timeout = TimeSpan.FromSeconds(5) };

        string status;
        try
        {
            var resp = await client.GetAsync(url);
            status = $"{(int)resp.StatusCode} {resp.ReasonPhrase}";
        }
        catch (Exception ex)
        {
            status = $"DOWN ({ex.Message})";
        }

        var line = $"[{DateTime.Now:yyyy-MM-dd HH:mm:ss}] {url} -> {status}";
        Console.WriteLine(line);
        File.AppendAllText(log, line + Environment.NewLine);
    }
}
```

Schedule these to run every 5–10 minutes to create a basic uptime monitor.
