my\_project/

│── main.py # Main script

│── config/

│ └── commands.ini # INI file with URLs

│── modules/

│ ├── \_\_init\_\_.py # Makes it a package

│ ├── logger.py # Logging setup

│ ├── api\_fetcher.py # API request handler

│ ├── html\_parser.py # Web page parser

My\_project

Common

propertyReader.py

excelMethods.py

Modules

UnixAcess

unixAccess.py

DBAccess

DBAccess.py

BitBucketAccess

BitBucketAccess.py

Python example to use curl commands and commands to be read from INI file and certifcate installation steps for https URL access

**ChatGPT said:**

Here’s a breakdown of your request:

1. **Python script to execute cURL commands**
2. **Reading cURL commands from an INI file**
3. **Steps to install certificates for HTTPS URL access**

**1. Python Script to Execute cURL Commands**

This script reads cURL commands from an INI file and executes them.

**Example INI File (commands.ini)**

ini

CopyEdit

[API\_REQUESTS]

request1 = curl -X GET https://example.com/api/v1/resource -H "Authorization: Bearer TOKEN"

request2 = curl -X POST https://example.com/api/v1/data -H "Content-Type: application/json" -d '{"key": "value"}'

**Python Script to Read and Execute cURL Commands**

python

CopyEdit

import configparser

import subprocess

# Read the INI file

config = configparser.ConfigParser()

config.read("commands.ini")

# Get all commands from the section

section = "API\_REQUESTS"

if section in config:

for key, cmd in config[section].items():

print(f"Executing {key}: {cmd}")

try:

result = subprocess.run(cmd, shell=True, capture\_output=True, text=True)

print("Output:", result.stdout)

print("Error:", result.stderr)

except Exception as e:

print(f"Failed to execute {key}: {e}")

else:

print(f"Section {section} not found in INI file.")

**2. Certificate Installation for HTTPS URL Access**

When using cURL to access HTTPS endpoints, you may need to install certificates if the request fails due to SSL issues.

**Windows**

1. Download the **CA certificate bundle** from cURL CA Bundle.
2. Place the downloaded cacert.pem file in a known directory (e.g., C:\curl\cacert.pem).
3. Set the environment variable:

powershell

CopyEdit

setx CURL\_CA\_BUNDLE "C:\curl\cacert.pem"

1. Restart your command prompt.

**Linux/Mac**

1. Update the CA certificates:

bash

CopyEdit

sudo apt update && sudo apt install ca-certificates -y # Debian/Ubuntu

sudo yum install ca-certificates -y # RHEL/CentOS

1. If using a custom CA, add it:

bash

CopyEdit

sudo cp my\_cert.pem /usr/local/share/ca-certificates/

sudo update-ca-certificates # Debian/Ubuntu

sudo update-ca-trust extract # RHEL/CentOS

1. Verify using:

bash

CopyEdit

curl --cacert /path/to/my\_cert.pem https://example.com

Would you like me to modify the script to handle specific error cases, such as SSL verification failures?

**You said:**

extend the Example to work for Windows machine , and extend the Example to access web application and use buitiful soup to process the HTML response and have exeption handling in process

**ChatGPT said:**

Here's an extended example that:

* Works on **Windows** and **Linux**
* Reads **cURL commands** from an INI file
* **Executes requests using requests** (instead of subprocess)
* **Handles SSL certificate issues** on Windows
* **Parses HTML responses using BeautifulSoup**
* **Includes exception handling**

**1. Install Required Packages**

Ensure you have these installed:

bash

CopyEdit

pip install requests beautifulsoup4

**2. Example INI File (commands.ini)**

Define your API/Web URLs and request configurations.

ini

CopyEdit

[API\_REQUESTS]

request1 = https://example.com/api/v1/resource

request2 = https://example.com/api/v1/data

[WEB\_PAGES]

page1 = https://example.com

page2 = https://news.ycombinator.com

**3. Python Script (execute\_requests.py)**

This script:  
✅ Reads URLs from the INI file  
✅ **Handles SSL certificates**  
✅ **Parses HTML** responses with **BeautifulSoup**  
✅ Provides **error handling**

python

CopyEdit

import configparser

import requests

from bs4 import BeautifulSoup

import os

import certifi

# Read INI file

config = configparser.ConfigParser()

config.read("commands.ini")

# SSL Certificate Handling for Windows

if os.name == "nt":

os.environ["REQUESTS\_CA\_BUNDLE"] = certifi.where()

def fetch\_api\_data(name, url):

"""Fetches API data from the given URL."""

try:

response = requests.get(url, verify=True, timeout=10)

response.raise\_for\_status() # Raise HTTP errors (4xx, 5xx)

print(f"API Response for {name}: {response.json()}") # Assuming JSON response

except requests.exceptions.RequestException as e:

print(f"Error fetching API data ({name}): {e}")

def fetch\_and\_parse\_html(name, url):

"""Fetches and parses HTML content from a webpage."""

try:

response = requests.get(url, verify=True, timeout=10)

response.raise\_for\_status()

soup = BeautifulSoup(response.text, "html.parser")

title = soup.title.string if soup.title else "No title found"

print(f"Web Page ({name}) Title: {title}")

# Extracting first paragraph text

paragraph = soup.find("p")

if paragraph:

print(f"First paragraph: {paragraph.get\_text()}\n")

else:

print("No paragraphs found\n")

except requests.exceptions.RequestException as e:

print(f"Error fetching web page ({name}): {e}")

# Process API Requests

if "API\_REQUESTS" in config:

for key, url in config["API\_REQUESTS"].items():

fetch\_api\_data(key, url)

# Process Web Pages

if "WEB\_PAGES" in config:

for key, url in config["WEB\_PAGES"].items():

fetch\_and\_parse\_html(key, url)

**4. Handling SSL Certificates on Windows**

If SSL issues persist on Windows:

1. Download **cacert.pem** from https://curl.se/ca/cacert.pem.
2. Set it as an environment variable:

powershell

CopyEdit

setx REQUESTS\_CA\_BUNDLE "C:\path\to\cacert.pem"

1. Or modify the script:

python

CopyEdit

requests.get(url, verify="C:/path/to/cacert.pem")

**5. Expected Output**

plaintext

CopyEdit

API Response for request1: {'status': 'success', 'data': {...}}

API Response for request2: {'status': 'success', 'message': 'Data saved'}

Web Page (page1) Title: Example Domain

First paragraph: This domain is for use in illustrative examples...

Web Page (page2) Title: Hacker News

First paragraph: Welcome to Hacker News, the best tech news aggregator...

This script now:  
✅ Works on **Windows/Linux**  
✅ Reads **API & web URLs** from INI  
✅ **Handles SSL issues**  
✅ **Parses HTML** using **BeautifulSoup**  
✅ Has **exception handling**

Would you like to extend this further with logging or additional parsing? 🚀

**You said:**

Yes please extend to use logging and additional parser. Share python sample to modularize python programs and import one program in another . consider the files are in different folders

**ChatGPT said:**

Here’s an **extended version** of the script with:  
✅ **Logging instead of print statements**  
✅ **Additional HTML parsing** (extracting links, headings)  
✅ **Modularization** (separate Python modules)  
✅ **Importing across different folders**

**Project Structure**

graphql

CopyEdit

my\_project/

│── main.py # Main script

│── config/

│ └── commands.ini # INI file with URLs

│── modules/

│ ├── \_\_init\_\_.py # Makes it a package

│ ├── logger.py # Logging setup

│ ├── api\_fetcher.py # API request handler

│ ├── html\_parser.py # Web page parser

**1. commands.ini (Configuration File)**

ini

CopyEdit

[API\_REQUESTS]

request1 = https://jsonplaceholder.typicode.com/posts/1

request2 = https://jsonplaceholder.typicode.com/users/1

[WEB\_PAGES]

page1 = https://example.com

page2 = https://news.ycombinator.com

**2. modules/logger.py (Logging Setup)**

python

CopyEdit

import logging

import os

# Create a logs directory if not exists

LOG\_DIR = "logs"

os.makedirs(LOG\_DIR, exist\_ok=True)

# Setup logging

logging.basicConfig(

filename=os.path.join(LOG\_DIR, "app.log"),

level=logging.INFO,

format="%(asctime)s - %(levelname)s - %(message)s",

)

# Function to get logger

def get\_logger(name):

return logging.getLogger(name)

**3. modules/api\_fetcher.py (API Request Handler)**

python

CopyEdit

import requests

from modules.logger import get\_logger

logger = get\_logger(\_\_name\_\_)

def fetch\_api\_data(name, url):

"""Fetch API data from the given URL."""

try:

response = requests.get(url, timeout=10)

response.raise\_for\_status() # Raise HTTP errors (4xx, 5xx)

logger.info(f"API Response for {name}: {response.json()}")

return response.json()

except requests.exceptions.RequestException as e:

logger.error(f"Error fetching API data ({name}): {e}")

return None

**4. modules/html\_parser.py (HTML Parsing)**

python

CopyEdit

import requests

from bs4 import BeautifulSoup

from modules.logger import get\_logger

logger = get\_logger(\_\_name\_\_)

def fetch\_and\_parse\_html(name, url):

"""Fetch and parse HTML content from a webpage."""

try:

response = requests.get(url, timeout=10)

response.raise\_for\_status()

soup = BeautifulSoup(response.text, "html.parser")

# Extract Title

title = soup.title.string if soup.title else "No title found"

logger.info(f"Web Page ({name}) Title: {title}")

# Extract first paragraph

paragraph = soup.find("p")

para\_text = paragraph.get\_text() if paragraph else "No paragraphs found"

logger.info(f"First paragraph: {para\_text}")

# Extract all links

links = [a["href"] for a in soup.find\_all("a", href=True)]

logger.info(f"Found {len(links)} links on {name}")

return {"title": title, "paragraph": para\_text, "links": links}

except requests.exceptions.RequestException as e:

logger.error(f"Error fetching web page ({name}): {e}")

return None

**5. main.py (Main Script)**

python

CopyEdit

import configparser

import os

from modules.api\_fetcher import fetch\_api\_data

from modules.html\_parser import fetch\_and\_parse\_html

from modules.logger import get\_logger

logger = get\_logger(\_\_name\_\_)

# Read INI File

config\_path = os.path.join("config", "commands.ini")

config = configparser.ConfigParser()

config.read(config\_path)

# Process API Requests

if "API\_REQUESTS" in config:

for key, url in config["API\_REQUESTS"].items():

response = fetch\_api\_data(key, url)

if response:

logger.info(f"Data fetched from {key}: {response}")

# Process Web Pages

if "WEB\_PAGES" in config:

for key, url in config["WEB\_PAGES"].items():

parsed\_data = fetch\_and\_parse\_html(key, url)

if parsed\_data:

logger.info(f"Parsed HTML data from {key}: {parsed\_data}")

**6. How to Run**

Navigate to the my\_project folder and run:

bash

CopyEdit

python main.py

**7. Logging Example (logs/app.log)**

pgsql

CopyEdit

2025-03-17 12:00:00 - INFO - API Response for request1: {'userId': 1, 'id': 1, 'title': 'Post Title', 'body': 'Post content...'}

2025-03-17 12:00:01 - INFO - Web Page (page1) Title: Example Domain

2025-03-17 12:00:02 - INFO - Found 5 links on page2

**✅ Key Features of this Approach**

✔ **Modular Code** – API and HTML processing are in separate modules  
✔ **Logging** – Stores logs instead of using print()  
✔ **Extensible** – Can add more features (e.g., database storage)  
✔ **Works Across Folders** – Modules can be reused in different projects

Would you like me to add more functionality, like **storing results in a database or CSV file**? 🚀

**You said:**

Project Structure

give a scenario

My\*project -> Common ->propertyReader.py MyProject ->common->\*excelMethods.py  MyProject -> Modules -> UnixAcess ->unixAccess.py MyProject -> Modules -> UnixAcess ->unixAccess.py

MyProject -> Modules -> DBAccess->DBAccess.py

MyProject -> Modules ->BitBucketAccess->BitBucketAccess.py

How to create project with above Structure and import commnMethod Function file to execute common functions'

give a scenario My\*project -> Common ->propertyReader.py MyProject ->common->\*excelMethods.py  MyProject -> Modules -> UnixAcess ->unixAccess.py MyProject -> Modules -> UnixAcess ->unixAccess.py MyProject -> Modules -> DBAccess->DBAccess.py MyProject -> Modules ->BitBucketAccess->BitBucketAccess.py How to create project with above Structure and import commnMethod Function file to execute common functions'

**ChatGPT said:**

Here’s how you can create a Python project with the given structure and properly import common functions across modules.

**📁 Project Structure**

bash

CopyEdit

MyProject/

│── main.py # Main entry point

│

├── Common/

│ ├── \_\_init\_\_.py # Makes it a package

│ ├── propertyReader.py # Reads properties/config files

│ ├── excelMethods.py # Handles Excel file operations

│

├── Modules/

│ ├── UnixAccess/

│ │ ├── \_\_init\_\_.py

│ │ ├── unixAccess.py # Unix-related functions

│ │

│ ├── DBAccess/

│ │ ├── \_\_init\_\_.py

│ │ ├── DBAccess.py # Database connection logic

│ │

│ ├── BitBucketAccess/

│ ├── \_\_init\_\_.py

│ ├── BitBucketAccess.py # BitBucket API functions

**🔹 1. How to Import Common Functions into Other Modules**

To use functions from Common/propertyReader.py or Common/excelMethods.py inside your modules (unixAccess.py, DBAccess.py, etc.), **ensure the Common folder is recognized as a package** by adding \_\_init\_\_.py inside it.

**✅ Example: Common/propertyReader.py**

python

CopyEdit

# propertyReader.py

import configparser

def read\_property(file\_path, section, key):

"""Reads a specific property from a .ini config file."""

config = configparser.ConfigParser()

config.read(file\_path)

return config.get(section, key) if config.has\_option(section, key) else None

**✅ Example: Modules/UnixAccess/unixAccess.py**

Importing and using read\_property() function from propertyReader.py:

python

CopyEdit

import os

import sys

# Add the Common directory to the sys.path

sys.path.append(os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), "../../Common")))

from propertyReader import read\_property # Import function from Common module

# Example usage

config\_file = "../../config.ini"

username = read\_property(config\_file, "UnixAccess", "username")

print(f"Unix Username: {username}")

**🔹 2. Alternative Import Approach Using PYTHONPATH**

Instead of modifying sys.path, set the environment variable **PYTHONPATH**:

bash

CopyEdit

export PYTHONPATH=$PYTHONPATH:/path/to/MyProject

Then, in any module, you can import like:

python

CopyEdit

from Common.propertyReader import read\_property

**🔹 3. Running the Script**

Run from the project root:

bash

CopyEdit

python Modules/UnixAccess/unixAccess.py

This setup ensures **modularization**, **easy function reuse**, and **maintainability**