

PYTHON

1. Method Overloading in Python (refer [practice.py](#))

- Method overloading allows multiple methods in the same class to have the same name but different parameters.

2. Method Overriding in Python (refer [practice.py](#))

- Method overriding occurs when a **subclass provides a specific implementation** of a method that is already defined in its parent class.
- >Can also use **super()** to **Call the Parent Method**

3.Shallow Copy ([copy.copy](#)) (refer [practice_copy.py](#))

- Creates a new object but does not duplicate nested objects.
- Any changes to a nested object will reflect in both copies.

4.Deep Copy ([copy.deepcopy](#)) (refer [practice_copy.py](#))

- Creates a completely new object with new nested objects.
- Changes in one do not affect the other.

5.Try-Except-Finally in Python (refer [try_except.py](#))

1. **try** Block

- This block contains the code that might raise an exception.
- If no exception occurs, the **except** block is skipped.

2. **except** Block

- Catches and handles specific exceptions.
- Multiple **except** blocks can be used for different exception types.

3. **finally** Block

- The **finally** block **always executes**, regardless of whether an exception was raised or not.
- It's typically used for cleanup actions like closing files or database connections.

6.File Operations(refer [file_handling.py](#))

File operations are **crucial** in almost every application that involves **storing, retrieving, processing, and managing data**

Function	Action
<code>create_file()</code>	Creates and writes to a file (overwrites if exists)
<code>write_to_file()</code>	Writes to a file (erases previous content)
<code>read_from_file()</code>	Reads and prints file content
<code>append_to_file()</code>	Appends new content to a file
<code>delete_file()</code>	Deletes a file
<code>check_if_file_exists()</code>	Checks if a file exists
<code>get_current_working_directory()</code>	Prints the current directory
<code>list_files_in_directory()</code>	Lists files in a directory
<code>get_size_of_file()</code>	Gets file size in bytes
<code>get_last_modified_time_of_file()</code>	Gets last modified time
<code>get_file_type()</code>	Gets (name, extension) of a file
<code>get_file_name()</code>	Gets only the file name
<code>get_file_extension()</code>	Gets only the file extension
<code>get_file_path()</code>	Gets absolute file path
<code>get_file_directory()</code>	Gets the directory path

Jenkins setup script execution:-

[./jenkins-setup-script.sh](#)

```

hafsa_027@Dell: ~/practice_je
=====
JENKINS ZERO TO HERO SETUP SCRIPT
=====
This script will setup Jenkins, create example files, and provide
detailed explanations of Jenkins concepts.
=====

Do you want to proceed with the setup? (y/n) y

==== CHECKING ENVIRONMENT ====

/ Running in WSL: Linux Dell 5.15.167.4-microsoft-standard-WSL2 #1 SMP Tue Nov 5 00:21:55 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
/ Running Ubuntu 24.04

==== CHECKING/INSTALLING JAVA ====

/ Java is already installed (version: 17.0.14)

# Setting up JAVA_HOME
/ JAVA_HOME set to /usr/lib/jvm/java-17-openjdk-amd64

==== CHECKING/INSTALLING JENKINS ====

/ Jenkins is already installed and running (version: )
▲ Skipping Jenkins installation

==== INSTALLING PYTHON AND REQUIRED PACKAGES ====

/ Python is already installed: Python 3.12.3
/ Virtual environment already exists at /home/hafsa_027/my_python_env
▲ Installing required Python packages...
Requirement already satisfied: pip in /home/hafsa_027/my_python_env/lib/python3.12/site-packages (25.0.1)
Requirement already satisfied: python-jenkins in /home/hafsa_027/my_python_env/lib/python3.12/site-packages (1.8.2)
Requirement already satisfied: multi-key-dict in /home/hafsa_027/my_python_env/lib/python3.12/site-packages (from python-jenkins) (2.0.3)
Requirement already satisfied: pbr>=0.8.2 in /home/hafsa_027/my_python_env/lib/python3.12/site-packages (from python-jenkins) (6.1.1)
Requirement already satisfied: requests in /home/hafsa_027/my_python_env/lib/python3.12/site-packages (from python-jenkins) (2.32.3)
Requirement already satisfied: six>=1.3.0 in /home/hafsa_027/my_python_env/lib/python3.12/site-packages (from python-jenkins) (1.17.0)
Requirement already satisfied: setuptools in /home/hafsa_027/my_python_env/lib/python3.12/site-packages (from pbr>=0.8.2->python-jenkins) (75.8.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /home/hafsa_027/my_python_env/lib/python3.12/site-packages (from requests->python-jenkins) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /home/hafsa_027/my_python_env/lib/python3.12/site-packages (from requests->python-jenkins) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /home/hafsa_027/my_python_env/lib/python3.12/site-packages (from requests->python-jenkins) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /home/hafsa_027/my_python_env/lib/python3.12/site-packages (from requests->python-jenkins) (2025.1.31)
/ Python packages installed successfully in virtual environment

==== CREATING SAMPLE PYTHON SCRIPT ====

/ Created Python script at: ~/jenkins-demo/scripts/jenkins_api_example.py
This script demonstrates how to interact with Jenkins using the Python API.
It requires the python-jenkins package. Install it with:
pip install python-jenkins

==== CREATING JENKINSFILE EXAMPLE ====

/ Created Jenkinsfile at: ~/jenkins-demo/pipeline-example/Jenkinsfile
This Jenkinsfile demonstrates a complete CI/CD pipeline with multiple stages.

==== CREATING JENKINS JOB XML EXAMPLE ====

/ Created Jenkins job XML at: ~/jenkins-demo/job-configs/freestyle-job.xml
This XML file demonstrates a complete Freestyle job configuration.

```

```

==== CREATING README ====

✓ Created README at: ~/jenkins-demo/README.md

==== JENKINS SETUP INSTRUCTIONS ====

Accessing Jenkins:
  1. Open your browser and navigate to http://localhost:8080
  2. For the initial setup, use the admin password shown earlier
    (You can retrieve it again with: sudo cat /var/lib/jenkins/secrets/initialAdminPassword)

Initial Jenkins Setup:
  1. Install suggested plugins (or select specific plugins)
  2. Create an admin user when prompted
  3. Configure Jenkins URL (default is fine for local development)

Recommended Plugins:
- Pipeline: Allows creating pipeline jobs (already included in suggested plugins)
- Blue Ocean: Modern UI for Jenkins pipelines
- Job Import Plugin: For importing the XML job config provided
- Git Integration: For source code management (already included in suggested plugins)

Setting up a Pipeline Job:
  1. Click 'New Item' in Jenkins
  2. Enter a name and select 'Pipeline'
  3. In the configuration, scroll down to the Pipeline section
  4. Select 'Pipeline script from SCM' if your Jenkinsfile is in a repository
  5. Or copy the contents of ~/jenkins-demo/pipeline-example/Jenkinsfile into the script area

For More Information:
- See the README.md file in ~/jenkins-demo/ for more instructions
- Explore the example files created in ~/jenkins-demo/

==== SETUP COMPLETE ====

```

Issue:-

VS Code was not opening

In command prompt :-

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```