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.

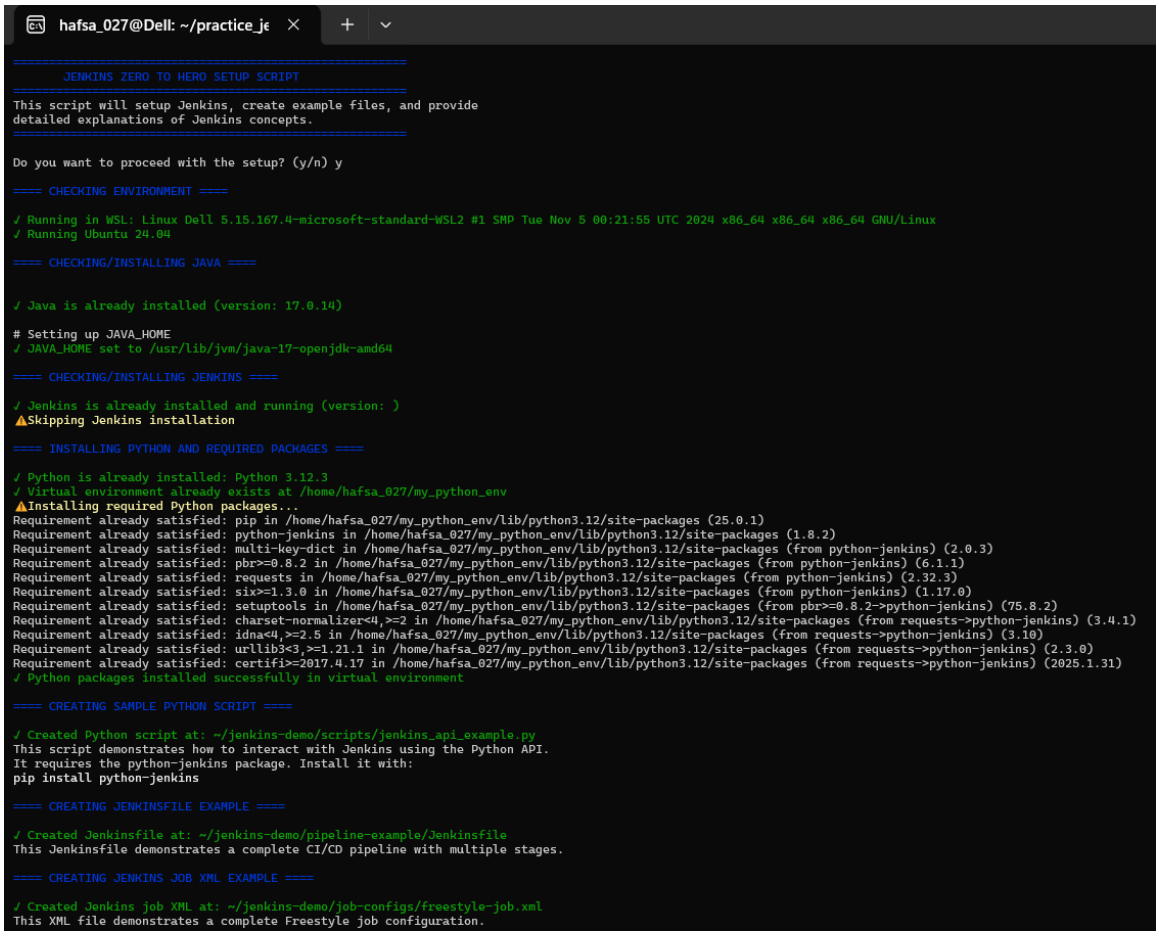
6.File Operations(refer file_handling.py)

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

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

Jenkins setup script execution:-

./jenkins-setup-script.sh




```
==== 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