Rahul Fere
Date: 06/02/2025

CMD (command prompt) Basics and most Used Commands :-

- cmd is a command line interpreter for windows that performs tasks without a graphical interface.

How CMD works ?

① User Input : You type a command
② Execution : CMD interprete and execute the command
③ Output : CMD displays the result or perform the action

To open CMD :
Press 'Win + R', type 'cmd" and hit "Enter".

Most Used CMD Commands :-

Command	Description
'dir'	Lists files and directories in the current folder.
'cd'	Changes the directory
'mkdir'	Creates a new folder
'rmdir'	Deletes an empty folder.
'del'	Deletes a file.
'copy'	Copies files.
'move'	Moves a file
'cls'	Clears the command prompt screen.
'echo'	Displays a massage
'ipconfig'	Shows network details.
'ping'	Checks internet connectivity
'netstat'	Displays network connections and ports.
'chkdsk'	Checks Disk for errors.
'format'	Formats a Drive.
'systeminfo'	Shows system details.

How CMD commands Works ?

- Each command consist of ;

- Command Name : The function to execute.
- Options / flags : Modifies how the command runs.
- Arguments : the target of the command.

Examples :

```
''' cmd
  Copy file.txt  D:\Backup
'''
```

- 'Copy' → command
- 'file.txt' → source file
- 'D:\Backup' → Destination folder.

Python Environment :

① What is a Python Environment ?

- A python environment is a setup where python runs with its dependencies, libraries, and settings. It ensures that projects use the correct versions of the libraries without conflict.

② Why use a python environment ?

① Avoid conflicts
② Organized Development
③ Portability
④ Control over dependencies.

Types of Python Environments :-

1. System Python :-
- The default python installation
- Not recommended for project-specific dependencies

2. Virtual Environment :
- Creates isolated environment for each project
- Example : 'venv'

3. Conda Environment :
- Used in data science and machine learning
- Manage dependencies efficiently.

How to use a Python Virtual environment ?

1) Create a virtual environment :

```sh
python -m venv myenv
```

- 'myenv' is the environment name.

2) Activate the environment :

- Windows :

```sh
myenv\Scripts\activate
```

3) Install Dependencies :

```sh
pip install requests pandas
```

- installs required libraries inside the virtual environment.

4) Deactivate the environment:

```sh
deactivate
```

5) Delete an Environment:

```sh
rm -rf myenv
```

When to use a Python Environment?

- Project developement
- Testing Different Versions
- Deployement
- Collaboration

Conclusion:

Python environment helps manage dependencies avoid conflicts, and mantain organized projects. Use 'venv' for regular projects and 'conda' for data science and machine learning.

Do we have to create a python environment again & Again?

→ No, you dont have to create a python environment repeatedly for every session. However, whether you need to operate it again or an existing one depends on how you manage your projects.

① Creating an environment : Once or Multiple times?

- If you are working on one project → create the environment once and use it every time.
- If you are working on multiple projects with different dependencies → create a seperate for each project to avoid conflicts.

② When do we need to create a New Environment?

1) Starting a New project
2) When different python versions are required
3) Avoding Dependency Conflicts
4) Experimenting with the new libraries
5) Deploying to Production.

③ When can you reuse the existing environment?

1) When working on the same project.
2) All projects use the same dependencies
3) Short term testing.

④ How to reuse the Existing Environment?

- Each time you want to use an existing environment, activate it instead of creating new one.

- Windows:

```
" sh
myenv\ scripts\ activate "
```

- mac/Unux:

```
" sh
source \myenv/ bin / activate "
```

⑥ Do we need to install packages Again?

- If you delete the environment, you will need to install packages again.

- If you reuse the same environment, all previously used libraries remain available.

- To check installed libraries:

```sh
pip list
```

⑦ Can we share or move the environment?

Yes! you can export dependencies and recreate the same environment on another system.

Saving installed packages:

```sh
pip freeze > requirements.txt
```

Recreating the same environment later

```sh
python -m venv myenv
source myenv/bin/activate
pip install -r requirements.txt.
```

classmate
Date
Page

Conclusion :-

- You do not need to create an environment every time.
- Use one environment per project and activate it when needed.
- Create a new environment only when working on different projects with different dependencies.
- You can save the environment setup and reinstall it later if needed.