Creating new environments for a session

Create a new Conda environment and install the required packages using the **conda** command. For example:

```
conda create -n myenv
conda activate myenv
conda install package_name
```

Explanation:

In the command `conda create -n myenv`, `myenv` is a name given to specify the specific environment, not a directory on the local system. It represents the name of the virtual environment that will be created by conda.

When you execute the `conda create` command with the `-n` or `--name` option, followed by the desired name, Conda creates a new virtual environment with that name. Conda manages the virtual environments internally and stores them in its own environment directory, which is typically located within the Conda installation.

The exact location of the environment directory depends on your operating system and the Conda configuration. By default, Conda creates the environment directory inside the Conda installation directory or in a dedicated Conda environment directory. It does not directly correspond to an existing directory on your local system.

For example, on Linux or macOS, the environment directory for Conda might be located at `~/anaconda3/envs/myenv` or `~/miniconda3/envs/myenv`. On Windows, it could be at `C:\Anaconda3\envs\myenv` or `C:\Miniconda3\envs\myenv`.

Once you create the virtual environment using the specified name (`myenv` in this case), you can activate it using the command `conda activate myenv` and start installing packages or working within that environment.

Remember to replace `myenv` with your desired name when creating the virtual environment using Conda.

How many environments can be active on a local system at a given time?

On a local system, the number of active environments at a given time depends on the specific tools you're using to manage virtual environments. Here's an overview of some common tools and their capabilities regarding the number of active environments:

1. `**venv**` (built-in Python module): With the `venv` module, which is included in the Python standard library since Python 3.3, you can create and activate multiple virtual environments simultaneously. There is generally no enforced limit on the number of active `venv` environments on a local system.

2. `**conda**` (Conda package and environment manager): Conda allows you to create and manage virtual environments as well. However, Conda operates on a slightly different model. By default, Conda restricts the activation of multiple environments simultaneously within the same shell session. So, only one Conda environment can be active at a given time within a specific shell. However, you can open multiple shell sessions and activate different environments in each session concurrently.

It's worth noting that although only one environment can be active within a specific shell session, you can switch between different environments by activating and deactivating them as needed.

In summary, the number of active environments at a given time depends on the specific tools you're using. With `venv`, you can have multiple active environments within a single shell session. With `conda`, you can activate one environment per shell session but can have multiple shell sessions with different active environments.

How to activate an environment for Jupyter Notebook?

When running a Jupyter Notebook, you can activate a virtual environment by following these steps:

1. **Install Jupyter Notebook**: Make sure you have Jupyter Notebook installed in your base environment or the global Python environment. If it's not already installed, you can install it using the following command:

```
pip install jupyter
```

2. **Activate the virtual environment**: Activate the desired virtual environment using the activation command specific to the virtual environment tool you are using. For example, with `venv` or `conda`, you can use the following commands, respectively:

```
source myenv/bin/activate  # Activate a venv environment
conda activate myenv       # Activate a conda environment
```

3. **Start Jupyter Notebook**: With the virtual environment activated, you can start Jupyter Notebook from the command line by running the following command:

```
jupyter notebook
```

4. **Access Jupyter Notebook in a web browser**: After executing the `jupyter notebook` command, it will launch a Jupyter Notebook server and provide a URL (usually http://localhost:8888/) in the terminal output. Copy the URL and paste it into a web browser. This will open the Jupyter Notebook interface in the browser.

By activating the virtual environment before starting Jupyter Notebook, you ensure that the notebook runs within the context of the selected environment. Any packages or dependencies required by your notebook will be sourced from the activated environment.

In the Jupyter Notebook interface, you can create a new notebook or open an existing one. From there, you can write and execute Python code within the notebook, utilizing the packages and dependencies of the activated environment.

Remember that the virtual environment needs to be activated each time you start a new Jupyter Notebook session. If you open multiple Jupyter Notebook instances, you'll need to activate the environment separately for each instance.

IMPLEMENTATION

- **Create a new environment:**

  Notice that when in global environment – the left side mentions "**base**"

```
(base) harshit@Harshits-MacBook-Pro ~ % conda create -n env1
```

  The installer also shows where the environment gets saved:

```
## Package Plan ##

  environment location: /Users/harshit/opt/anaconda3/envs/env1
```

  # To activate this environment, use
  #
  # $ conda activate env1
  #
  # To deactivate an active environment, use
  #
  # $ conda deactivate

- **Activate the environment:**

```
(base) harshit@Harshits-MacBook-Pro ~ % conda activate env1
(env1) harshit@Harshits-MacBook-Pro ~ %
```

  Note how the left side changed from "**base**" to "**env1**"

- **Install "pip" in the new environment:**

  If your `.zshrc` file cannot find `pip` in your new environment, it's likely because the virtual environment's Python executable and associated packages are not in the system's PATH by default. When you create a new virtual environment, it is isolated from the system's Python installation and has its own separate set of packages.

  To ensure that the `pip` command is accessible within your virtual environment, you can take the following steps:

  1. **Activate the virtual environment**: Activate your virtual environment using the appropriate activation command. For example, if you are using `venv`, you would use the following command:
  ```
  source myenv/bin/activate
  ```
  Replace `myenv` with the name of your virtual environment.

2. **Install or verify `pip` installation**: Once the virtual environment is activated, check if `pip` is installed by running the following command:
```
pip --version
```
If `pip` is not installed, you can install it within the virtual environment using the `ensurepip` module. Run the following command:
```
python -m ensurepip --upgrade
```
This will ensure that `pip` is installed and up to date within the virtual environment.

3. **Use virtual environment-specific `pip`**: With the virtual environment activated and `pip` installed, you can now use the `pip` command specific to that environment. For example, running `pip install package_name` will install the package into the virtual environment rather than the system Python.

It's important to note that the `pip` command within the virtual environment is separate from the system's `pip` command. Each virtual environment has its own isolated set of packages and its own version of `pip`.

By activating the virtual environment and using the `pip` command within that environment, you can ensure that the correct version of `pip` and the associated packages are used.

```
(env1) harshit@Harshits-MacBook-Pro ~ % sudo python -m ensurepip --upgrade
```

- **To upgrade pip:**

```
(env1) harshit@Harshits-MacBook-Pro ~ % sudo pip install --upgrade pip
```

- **Install the required packages using “pip”**

Note that the packages installed in the environment will only be effective when the specific environment is active. By default, the global environment is active, but we might not want to change the global settings every time to suit the needs of a project and hence we have made a new environment which will have specific libraries and versions needed for the project.

- **Update “jupyter” and “notebook”**

Update your Jupyter Notebook installation and its dependencies by running the following commands:

```
pip install --upgrade jupyter
pip install --upgrade notebook
```

(For some reason jupyter didn’t install in my environment after the previous step so I had to install jupyter in next step)

- **Install Jupyter Notebook**: If Jupyter Notebook is not installed, you can install it using the following command:

  pip install jupyter

- **Start Jupyter Notebook**: With the virtual environment activated, you can start Jupyter Notebook from the command line by running the following command:

  jupyter notebook

- **To install a package:**

  pip install <package-name>

- **To upgrade a package**:

  pip install --upgrade <package-name>

- **To install a specific version of a package:**

  pip install <package-name>==<desired_version>