Lesson Reflection

Lesson Summary

This lesson provided an overview of setting up Python environments for developing portable data projects. It introduced key concepts like Python packages and modules, the pip installer, virtual environments, and requirements files.

The lesson walked through using pip to install, uninstall, and update third party packages. It also demonstrated creating isolated virtual environments per project using venv, activating environments, installing packages locally, and deactivating environments when done.

Additionally, the lesson showed how to use pip freeze to generate a requirements file documenting all packages for a project. Learners practiced uninstalling packages from a requirements file and using the file to reinstall packages in a new environment.

Key Points

- Python code can be distributed in packages and reusable modules
- pip installs/manages packages from the Python Package Index
- Virtual environments isolate dependencies on a per project basis
- Requirements files record packages needed to rerun projects
- pip helps recreate project environments from requirements files

Reflection Questions

- Why is it useful to create separate virtual environments for Python projects?
  ↳ to prevent on different packages version on each project in the same environment local com. and make it portable
- How can requirements files make Python projects more portable?
  ↳ pip freeze
- What pip commands are helpful for managing virtual environments?
  ↳ py3 -m venv env
- What are some best practices for organizing Python project code and environments?
  ↳ check pip list
- How could virtual environments and requirements files facilitate team collaboration?
  ↳ portable, so you can share a file that you want and don't worry about packages if that file was used that env has already contained them

Challenge Exercises

- Create a virtual environment for a new Python project
- Install a package like Pandas into the virtual environment
- Freeze the environment to generate a requirements file
- Deactivate then reactivate the environment and confirm Pandas still imports
- Share the project folder & requirements file and attempt to recreate the environment on another system
- Discuss challenges faced and best practices discovered

Code Examples

Pandas DataFrame creation and usage

```python
# Pandas in virtual environment

import pandas as pd

# Create DataFrame
data = {'Apples': [30], 'Bananas': [21]}
purchases = pd.DataFrame(data)

print(purchases)
```

```
   Apples  Bananas
0      30       21
```

Virtualenv workflow

```
# Create virtual environment
python3 -m venv my_env

# Activate virtual environment
source my_env/bin/activate

# Install packages
pip install pandas

# List installed packages
pip list

# Generate requirements file
pip freeze > requirements.txt

# Deactivate environment
deactivate
```