Python Package Management with Anaconda

T. Pranavan

ECE, NUS

July 30, 2025

Session Objectives

- Understand the purpose of Anaconda and Conda
- Create and manage virtual environments
- Install, update, and remove Python packages
- Compare Conda and pip
- Learn best practices for reproducibility

What is Anaconda?

- Free, open-source Python distribution for data science and ML
- Comes with Conda, Jupyter, and over 250 packages pre-installed
- Includes a powerful environment and package manager

Anaconda vs Miniconda

- **Anaconda**: Full package distribution, GUI (Navigator), 250+ libraries
- **Miniconda**: Minimal installer, install only what you need
- Use Miniconda for lightweight setups

Installing Anaconda

- Download from `https://www.anaconda.com`
- Follow installation steps for your OS
- Run `conda init` to set up shell integration
- Verify: `conda --version`

What is a Conda Environment?

- Isolated Python environment with specific packages
- Prevents dependency conflicts
- Helps with reproducibility and project separation

Creating and Activating Environments

```
# Create a new environment with Python 3.10
conda create -n myenv python=3.10

# Activate the environment
conda activate myenv

# Deactivate environment
conda deactivate
```

Managing Environments

```
# List environments
conda env list

# Remove an environment
conda remove -n myenv --all

# Export environment to YAML
conda env export > environment.yml

# Create env from YAML
conda env create -f environment.yml
```

Installing Packages with Conda

```
# Search for a package
conda search numpy

# Install package
conda install numpy

# Install specific version
conda install pandas=1.5.0

# Update a package
conda update scikit-learn
```

Removing and Listing Packages

```
# Remove a package
conda remove matplotlib

# List installed packages
conda list
```

Conda Channels

- Channels are package sources (default: `defaults`)
- Popular channel: `conda-forge`

Using Pip Inside Conda

- Use pip when package not in Conda

```
conda install pip
pip install somepackage
```

- Be cautious of mixing Conda and pip extensively

Conda vs Pip

- **Conda**: binary package manager (not just Python)
- **Pip**: Python-only package manager (PyPI)
- Conda resolves dependencies better in many cases

Best Practices

- Use a separate environment per project
- Use YAML files for reproducibility
- Prefer Conda packages when possible
- Keep environments minimal and documented

Common Issues

- Dependency conflicts – use specific versions
- Long install time – try Mamba as a faster alternative
- Breakages – export and rebuild environments

Summary

- Conda simplifies environment and dependency management
- Use environments to isolate and control setups
- Export to YAML for collaboration and reproducibility

Questions?