P1.1.1 - Python Environment Setup (Part 1)

Welcome to the Python Environment Setup module! This guide will help you set up a proper Python development environment.

Topics Covered in Part 1:

1. Python Installation (3.12)
2. Virtual Environments (venv)
3. Installing Packages with pip (including Jupyter Notebook)

---

1. Python Installation (3.12)

What is Python?

Python is a high-level, interpreted programming language known for its simplicity and readability. Python 3.12 is one of the latest stable versions with improved performance and new features.

Installation Steps

Windows:

1. Visit https://www.python.org/downloads/
2. Download Python 3.12.x installer
3. Run the installer
4. **Important**: Check "Add Python to PATH"
5. Click "Install Now"

macOS:

```
# Using Homebrew
brew install python@3.12
```

Linux:

```
# Ubuntu/Debian
sudo apt update
sudo apt install python3.12

# Fedora
sudo dnf install python3.12
```

Verify Installation

After installation, open a terminal or command prompt and verify Python is installed correctly:

```
# Check Python version
python --version
# or on some systems
python3 --version

# Check if Python is in PATH
python
# This should open Python interactive shell
# Type exit() to quit

# Check pip version
pip --version
# or
pip3 --version
```

Expected output:

```
Python 3.12.x
```

If you see the version number, Python is successfully installed!

Understanding Python Versions

- **Python 2.x**: Legacy version (deprecated since 2020)
- **Python 3.x**: Current version with modern features
- **Python 3.12**: Latest stable release with performance improvements

---

2. Virtual Environments (venv)

Why Virtual Environments?

Virtual environments allow you to:

- Isolate project dependencies
- Avoid version conflicts between projects
- Maintain clean Python installations
- Share reproducible environments

Creating a Virtual Environment

Windows:

```
# Create virtual environment
python -m venv myenv
```

```
# Activate
myenv\Scripts\activate

# Deactivate
deactivate
```

macOS/Linux:

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

# Activate
source myenv/bin/activate

# Deactivate
deactivate
```

Best Practices

1. Create one virtual environment per project
2. Name it `venv` or `.venv` for consistency
3. Add virtual environment folder to `.gitignore`
4. Document dependencies in `requirements.txt`

Verify Virtual Environment

After activating your virtual environment, you can verify it's active:

```
# Windows
where python

# macOS/Linux
which python
```

You should see the path pointing to your virtual environment folder.

---

3. Installing Packages with pip

What is pip?

`pip` is Python's package installer. It allows you to install and manage third-party packages from the Python Package Index (PyPI).

Basic pip Commands

```
# Install a package
pip install package_name

# Install specific version
pip install package_name==1.2.3

# Install multiple packages
pip install package1 package2 package3

# Upgrade a package
pip install --upgrade package_name

# Uninstall a package
pip uninstall package_name

# List installed packages
pip list

# Show package information
pip show package_name

# Search for packages
pip search package_name
```

Installing Jupyter Notebook

Now it's time to install Jupyter Notebook so you can work with interactive Python notebooks!

```
# Install Jupyter Notebook
pip install notebook

# Or install JupyterLab (modern interface)
pip install jupyterlab

# Verify installation
jupyter --version
```

Recommended: Install both notebook and commonly used packages:

```
pip install notebook jupyterlab numpy pandas matplotlib
```

Starting Jupyter Notebook

Once installed, you can start Jupyter Notebook:

```
```

```
# Launch Jupyter Notebook
jupyter notebook

# Launch JupyterLab
jupyter lab

# Launch on specific port
jupyter notebook --port=8889

# Launch in a specific directory
jupyter notebook /path/to/directory
```

This will open Jupyter in your web browser at `http://localhost:8888`

Managing Dependencies

Creating requirements.txt

```
# Export current environment
pip freeze > requirements.txt
```

Example `requirements.txt`:

```
notebook==7.0.6
jupyterlab==4.0.9
numpy==1.26.2
pandas==2.1.4
matplotlib==3.8.2
```

Installing from requirements.txt

```
pip install -r requirements.txt
```

This ensures everyone on your team uses the same package versions.

Next Step

Once you have Jupyter Notebook installed and running, open **Part 2** (`P1_1_1_1_Part2_Jupyter.ipynb`) to learn how to use Jupyter Notebooks interactively!