Managing Python Dependencies – Workbook

Welcome to *Managing Python Dependencies*!

Use this printable workbook to track your learning progress as you move through the course.

Each page in the workbook is dedicated to a single lesson in the course. You can mark lessons and modules as completed (□ → ☑) and take your learning notes in the blank space on each page. This will help you build up your mental "knowledge map" of the topic and reinforce your learning.

I would encourage you to take *handwritten* notes and to even add little mind maps and scribblings. Research shows that this helps retain more information and for longer periods of time.

By completing this workbook you'll get a personalized "cheat sheet" covering your unique perspective on the topic. A personalized cheat sheet is a priceless learning resource. Highly recommended!

By the way, some of the topics covered in the course are quite "fractal." The more you drill down and learn about them, the more complex they get. If this is your first time working with Python's dependency management techniques it is completely normal to feel a little overwhelmed.

These challenges are *a normal part of the learning process*. When you feel like you're banging your head against the wall—that's often when real progress happens and you make a big leap forward. So don't be too hard on yourself if you feel a little out of your depth. I know you'll be able to see through it!

Good luck and:

Happy Pythoning!

— Dan Bader (dbader.org)

□ Module 1: Welcome & Course Overview

□ Lesson 1.1 Welcome & Course Overview

Notes:

□ Module 2: Managing Third-Party Dependencies With pip

- **Dependency management** enables modern software development by making well-packaged building blocks available for use in your own programs.
- Key tool: **pip**—Python's recommended **package manager**
- Python packages are hosted on package repositories (**PyPI**)
- pip has powerful **version management features**

Notes:

□ Lesson 2.1 Introduction to Dependency Management

Notes:

□ Lesson 2.2 pip: The Python Package Manager

Notes:

□ Lesson 2.3 Installing & Updating pip

Notes:

□ Lesson 2.4 Python Package Repositories

Notes:

□ Lesson 2.5 Installing Packagages With pip

Notes:

□ Lesson 2.6 Identifying & Updating Outdated Packages

Notes:

□ Lesson 2.7 Uninstalling Packages

Notes:

□ Module 3: Isolating Dependencies With Virtual Environments

- Virtual environments keep your project dependencies **isolated**.
- They help you **avoid version conflicts** between packages and different versions of the Python runtime.
- As a **best practice**, all of your Python projects should use virtual environments to store their dependencies.

Notes:

□ 3.1 Introduction to Virtual Environments

Notes:

□ 3.2 Creating and Activating a Virtual Environment

Notes:

□ 3.3 Installing Packages Into a Virtual Environment

Notes:

□ 3.4 Deactivating Virtual Environments

Notes:

□ 3.5 Destroying Virtual Environments

Notes:

□ 3.6 My Virtual Environment Workflow

Notes:

□ Module 4: Finding Quality Python Packages

- Python has a **rich third-party library ecosystem**. Using it effectively is the key to becoming **more productive**.
- Using a third-party package always has a **maintenance cost**.
- Stick to **high-quality packages** to keep maintenance costs under control.

Notes:

□ 4.1 How Third-Party Packages Can Help You

Notes:

□ 4.2 Finding Popular Packages on Curated Lists

Notes:

□ 4.3 "Rules of Thumb" for Selecting a Great Package

Notes:

□ Module 5: Setting Up Reproducible Environments & Application Deploys

- Requirements files allow you to specify the **third-party dependencies** of a Python program.
- This makes dependency installs and application deployments **repeatable**.
- Dependencies can be **captured** (`pip freeze`) and **restored** (`pip install -r`) with **pip**.

Notes:

□ 5.1 Introduction to Requirements Files

Notes:

□ 5.2 Capturing Project Dependencies

Notes:

□ 5.3 Restoring Captured Dependencies

Notes:

□ 5.4 Separating Development and Production Dependencies

Notes:

□ 5.5 Requirements files best practices

Notes:

□ Module 6: Course Conclusion

You now know how to:

- Manage Python project dependencies with pip
- Isolate project dependencies with virtual environments to avoid version conflicts
- Find and identify quality third-party packages to use in your own Python projects
- Set up repeatable development environments and application deployments

Notes: