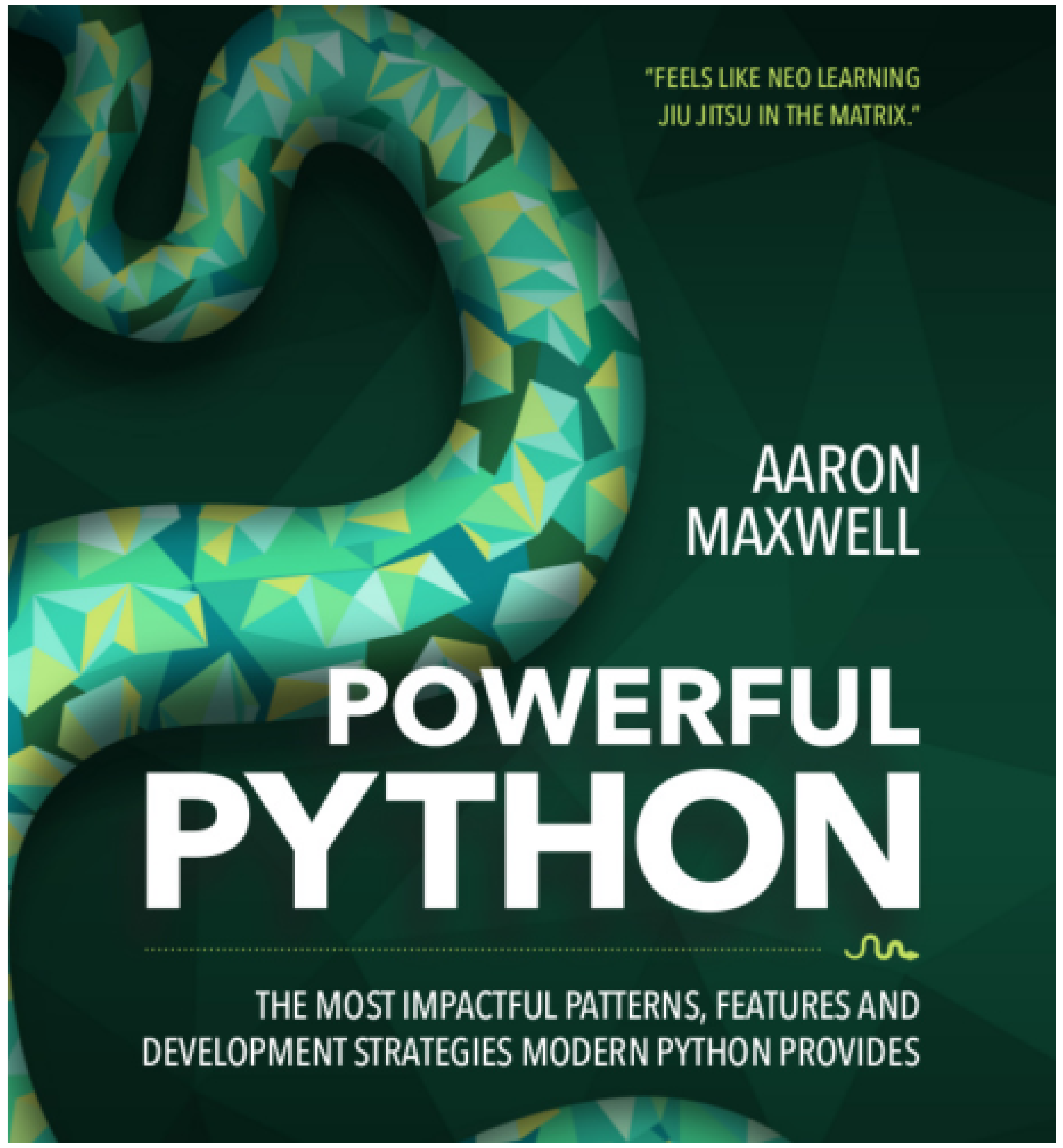
Beyond Python Scripts: Logging, Modules, And Dependency Management

Welcome

I’m your host, Aaron Maxwell.

- Author of **Powerful Python**
- aaron@powerfulpython.com
- On twitter:
  - @powerfulpython (professional)
  - @redsymbol (personal)


"FEELS LIKE NEO LEARNING JIU JITSU IN THE MATRIX."
AARON MAXWELL
POWERFUL PYTHON
THE MOST IMPACTFUL PATTERNS, FEATURES AND DEVELOPMENT STRATEGIES MODERN PYTHON PROVIDES

Our focus: Python’s features for **larger applications**. Today is **part one**, covering logging, modules, and dependency management. Part two covers exceptions, error handling and command-line interfaces.

Python Scripts

Many people use Python for **scripts**.

- Small enough to easily fit in one file
- Often written and maintained by one developer
- Frequently written to solve a specific problem
- If in version control, work is done in mainline branch
- Often does NOT have unit tests
- Often is not code reviewed

Python Applications

For larger **applications**, it’s a whole different game.

- Split across many different files. Too many LOC for one
- Worked on concurrently by a team of developers
- All work done in feature branches, merged into mainline
- Version control an **absolute must**
- Extensive test coverage
- Likely to be code reviewed
- Static code analysis

At this scale, sheer complexity becomes a barrier to progress.

We **explode** through that barrier by leveraging certain features of Python.

How we will proceed

Download courseware ZIP:

- Click on the green "Resource List" icon
- Or fetch from: https://powerfulpython.com/courseware-app1.zip

What's included:

- PDF course book
- Slides
- README.txt with pointers
- Labs (i.e., programming exercises - more on that later)

Give you a break every hour (10 minutes).

Give me a thumbs up. (Let's try it now)

Ask questions anytime.

Python versions

Most code I show you will run in both Python 2 and 3.

Where it's different, I'll code in Python 3, and point out the differences. (There won't be many.)

You can do the programming exercises in either 3, or 2.7.

What makes perfect?

Practice, practice, practice.

- Practice syntax (typing things in)
- Practice programming (higher-level labs)

I expect you to do your part!

You **exponentially** get out of this what you put into it.

GO FOR IT.

Running the labs

Labs are the main programming exercises. You are given a failing automated test; your job is to write Python code to make it pass.

Simply run it as a Python program, any way you like. (For example, "python3 helloworld.py")

Run unmodified first, so you can see the failure report.

When done, click the thumb’s up, and find someone to high-five.

Then: Move on to the extra credit.

Lab: helloworld.py

Let's do our first lab now: 'helloworld.py`

- In `labs/py3` for Python 3.x, or `labs/py2` for 2.7

When you finish:

- Give me a HIGH FIVE! in the chat room, and click Thumbs up, so I know you're done.
- Proceed to `helloworld_extra.py`

You'll know the tests pass when you see:

```
*** ALL TESTS PASS ***
Give someone a HIGH FIVE!
```

When that's done: Open and skim through **PythonLogging.pdf** - just notice what topics interest you.

Getting the most

We’ll take some class time for each lab. You may not finish, but it’s **critically important** that you at least start when I tell you to.

After we’re done for the day, find time to finish all the main labs before tomorrow.

Solutions are provided. Use them wisely, not foolishly:

- After you get the lab passing, compare your code to the official solution.
- Other than that, don’t look at them if you can avoid it.
- The more work you can do on your own, the more you will learn. Peek at the solution to get a hint when you really need it.

Optional (**only** for future master Pythonistas): Do all the extra labs as well, as soon as you can manage.