Python for Beginners

Mentoring and Tips & Trick

Presented at the FayettePy monthly meetup on October 18, 2018
By Dennis Meek

Why Should Anyone Want to be a Great Mentor?

Benefits that studies show[1]:

- Mentors often make more money than non-mentors
- Mentors are more engagement in jobs/promotions
- Mentors receive Joy - Not simple happiness, but joy from their relationships with proteges

Most Important:

Legacy

When you die no one is going to care if you did X,

but people will remember if you invested in them

*Master craftsman/apprentice relationships worked for thousands of years, the notion of going at it alone is a new concept in general

[1] Source: *Power Mentoring: How Successful Mentors and Proteges Get the Most Out of Their Relationships,* Ensher, Ellen & Murphy, Susan

Mentoring Guidelines

Mentors:

- You don’t need to be an expert, you just need to be **one-step ahead** of the person you are mentoring
- If you don’t know something, offer to explore the subject together
- Nothing says you have to be older than the person you mentor
- It’s ok to test someone to see if they are serious and worth the time investment

Mentees:

- A good mentor will not spoon feed you
- The mentor is investing in you, don’t waste the opportunity or their time

Both:

- Learning is a two way process
- The relationship can vary in time and in scope
- Either party can dissolve the relationship at any time necessary, but explain why
- Get to know each other, it’s a relationship, but remember to share at an exchanged level

What happens when an “A” biology student with no experience with computers changes majors to CIS?

- COBOL - C — Thank goodness they dropped the second semester requirement
- Databases/SQL - A
- Networking - A (Bash was ok)
- Visual Basic - C
- Object Oriented Design - A (going into the final)
  - Stupid COBOL program wouldn’t run - C
- Capstone Course - A (exchange student)

So what is one to do?

A: Go into networking and avoid programming forever! Until the game changed.

So for a few years…

All I really needed to do good work was…

and
sometimes

Most large companies have well developed systems and processes for dealing with large volumes of data. With that, most analytics was easily done.

But then I changed jobs in 2016…

How the account was described

What I found when I started

Sometimes Help Shows Up In Unexpected Ways

$15 Humble Bundle

Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?

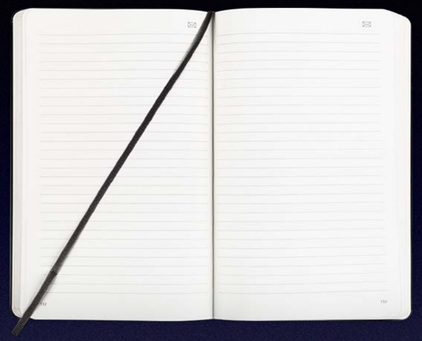
Tips: Getting Started

- Learn with a focus on teaching
  - You will remember more and it’s the first step to being a mentor
  - Tip: Try and teach what you learn each week to someone else within five days
- Learn principles, not syntax
  - Principles can transfer from language-to-language
- Write code every day (at least 30 minutes)
  - No computer, write in a notebook, on a napkin, in sand, or whatever (more on that later)
- Don’t avoid hard topics - RegEx
- Don’t talk ill about the code you write — It’s a self fulfilling thing
- There is no such thing as one and done

Tips: Retention

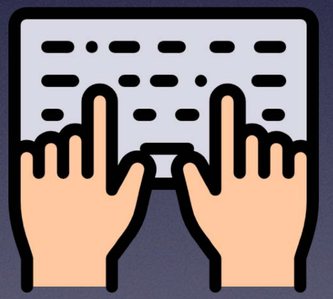
“Did you really spend $300 of your annual budget on notebooks?”

- Yes, hand writing notes as you learn, in pencil, will help you retain more information.
- Plus, it will help you see your progress when you hit a wall — and you will.



Examples & Exercises

- Type every exercise and do not skip any exercises — it will help develop muscle memory
- DO NOT go to Stack Overflow and copy & paste.



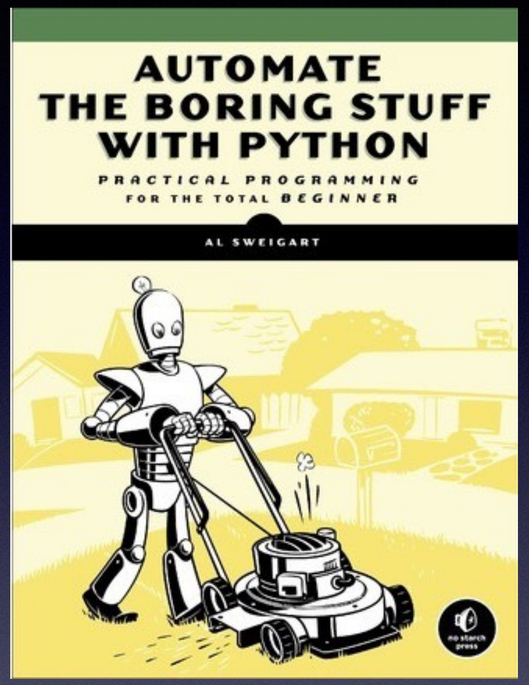
Memory Builders

- Focus on a certain skill — loops, if - elif - else statements, functions, etc.
- Write example code -> delete -> write from memory -> delete -> repeat

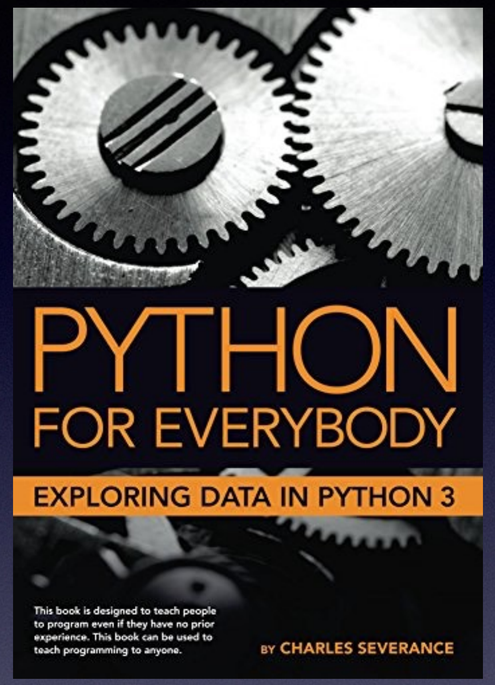
Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?

Absolute Beginners & Career Augmenters



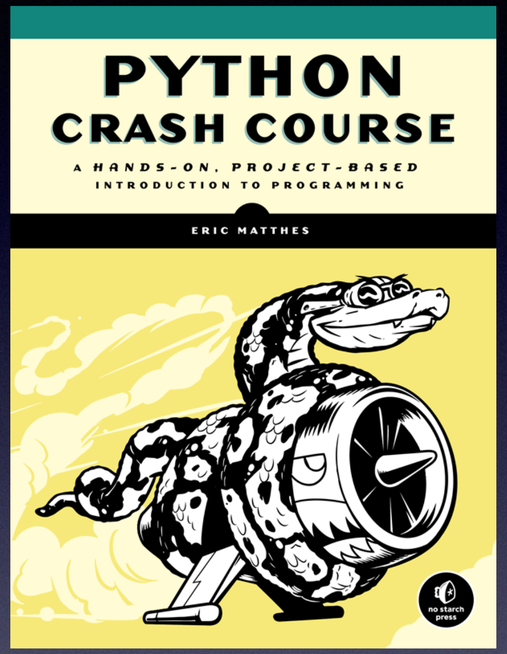
- Progressive lessons
- Integrated exercises
- End of book projects
- Great for repetitive tasks



- PDF version of the book is free
- Video lessons for each chapter
- Computer graded exercises
- Key lesson: database keys
- More of a program

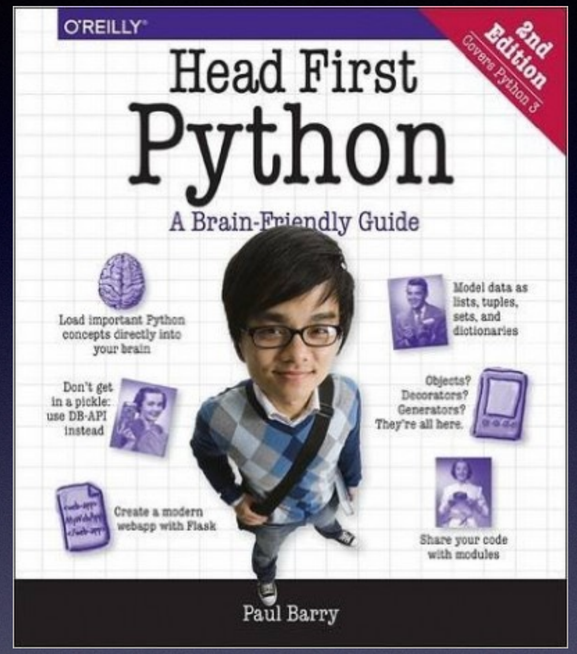
Some Programming Experience

Good in General



- Packed with a lot of information
- Has three project appendixes
  - PyGame
  - Data Visualization
  - Django (skip till later)

Better for Visual Learners



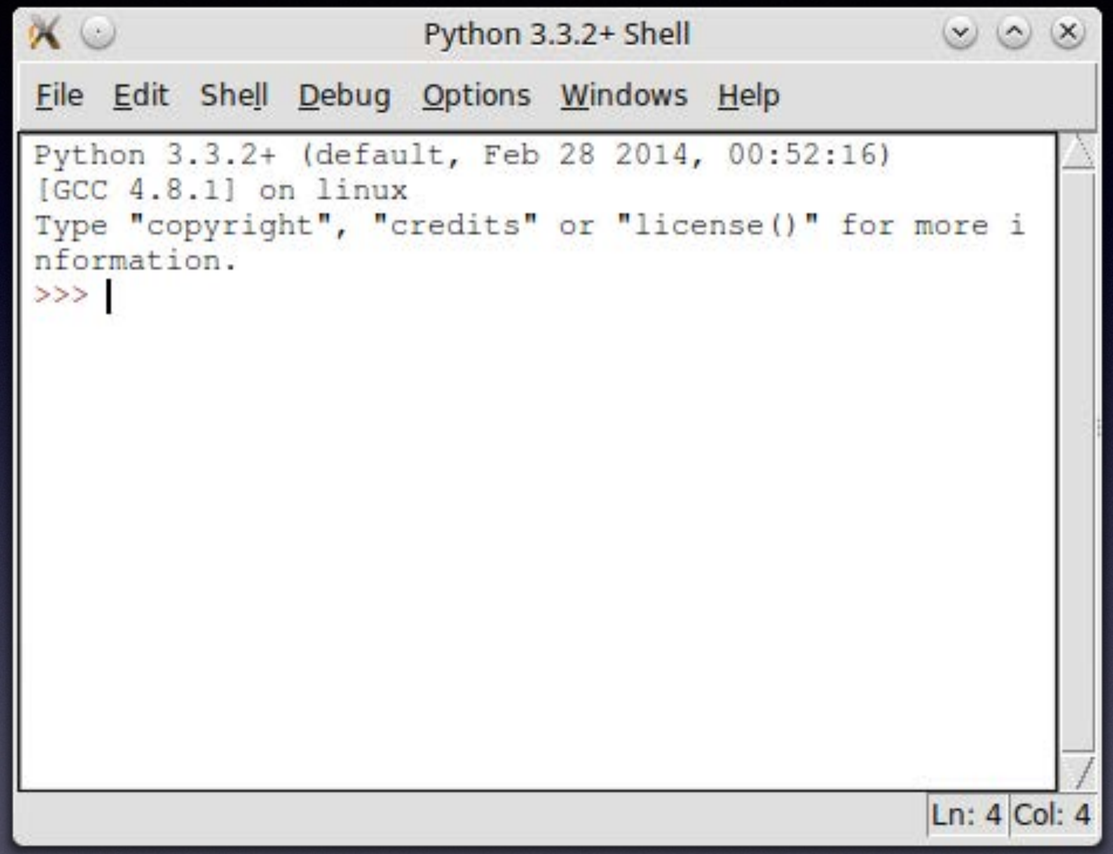
- Hits all the senses
- Not meant to be a reference guide
- Designed for a ten day read
- Inviting conversational approach

Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?

IDEs: Use IDLE First

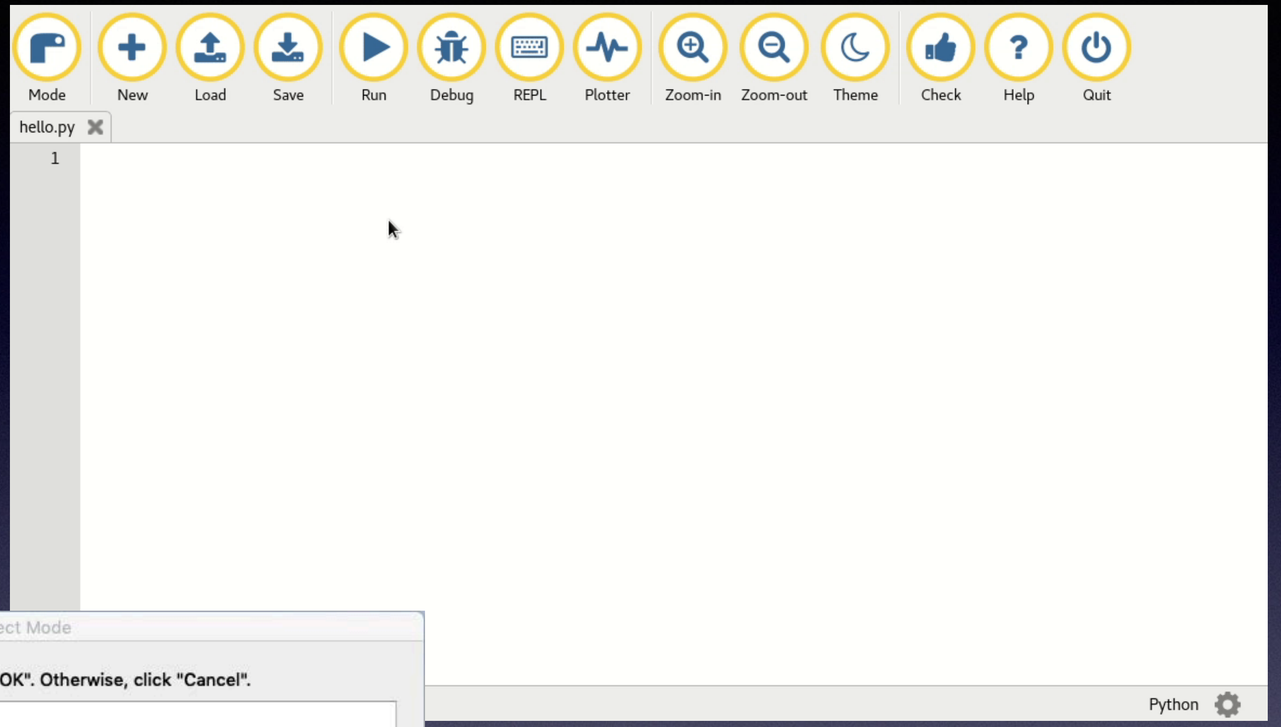
- The Python console forces you think more about how Python works
- Most full featured IDEs have a console window implementation
- Documentation info can be pulled up in the console
- Good troubleshooting tool
- For better visibility install iPython (Interactive Python)

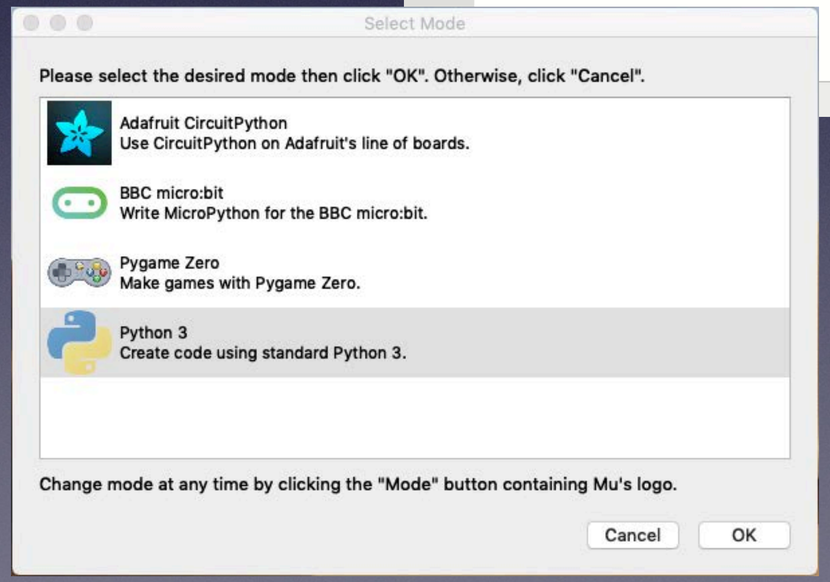


IDEs: Simple is Good

Start with a simple IDE that can grow with you — Mu Editor

- https://codewith.mu
- Great tutorials
- MicroPython mode





IDEs: Feature Rich

Graduate to feature heavy IDE

- Sublime can be difficult to setup
  - Corey Schafer's YouTube channel has a good tutorial
- PyCharm can be confusing, but there are good tutorials
  - Supports PyTest, Flask & Django, Pipenv, and Docker
- WingIDE
  - Base functionality is great
  - Tutorial documents are confusing
  - No subscription

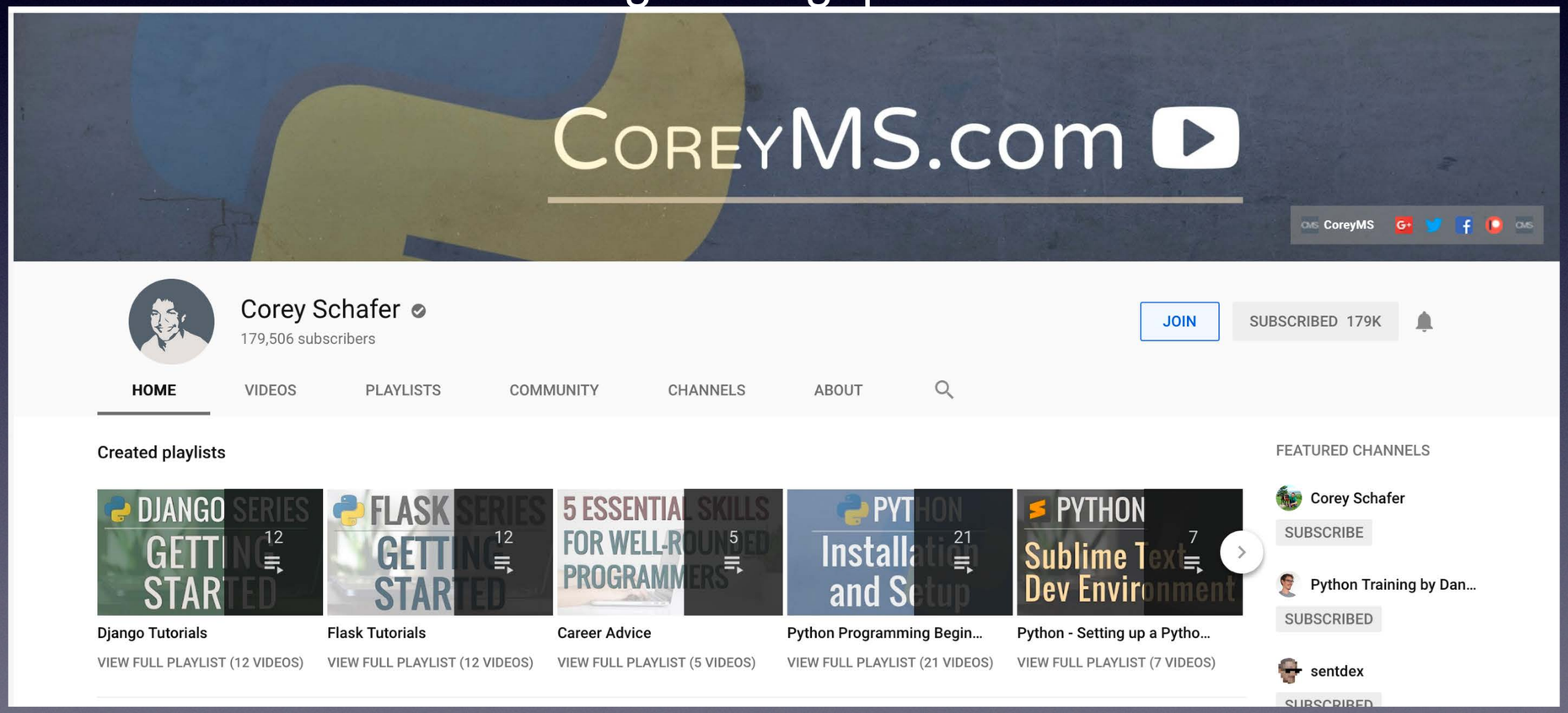
Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?

Tutorial Omissions

- All beginning tutorials leave some little points that will make you crazy
  - For example, what's a PYTHONPATH and how do I set it in the IDE I'm using

Best source I found for filling in the gaps



Mentors: Try to take yourself back to zero

Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?

Helpful Podcasts

- Weekly podcast
- About an hour each
- Discussions on a variety of topics
- Interesting guests

- Weekly library discussions
- Less than an hour long
- Multiple short topics
- The show description always have a full transcript with links

Interesting Topics and Helpful Resources

Weekly Podcast

TalkPython['Podcast'] Episodes Python Courses Friends of the show Patreon Merch Contact

Recorded episodes

Subscribe @ iTunes

Show number	Date	Title	Guests
#181	2018-10-12	30 amazing Python projects	Brian Okken
#180	2018-10-02	What's new in Python 3.7 and beyond	Anthony Shaw
#179	2018-09-26	Python Language Summit 2018	Panelists
#178	2018-09-21	Coverage.py	Ned Batchelder
#177	2018-09-15	Flask goes 1.0	David Lord
#176	2018-09-10	The Python Community by the Numbers	Panelists
#175	2018-08-31	Teaching Python to network engineers	Hank Preston III
#174	2018-08-16	Coming into Python from another Industry (part 2)	Panelists
#173	2018-08-07	Coming into Python from another Industry (part 1)	Panelists
#172	2018-08-01	Nuitka: A full Python compiler	Kay Hayen
#171	2018-07-29	1M Jupyter notebooks analyzed	Adam Rule
#170	2018-07-20	Guido van Rossum steps down	Panelists
#169	2018-07-13	Becoming a Python content creator	Corey Schafer
#168	2018-07-06	10 Python security holes and how to plug them	Panelists
#167	2018-06-29	Simplifying Python's Async with Trio	Nathaniel Smith
#166	2018-06-14	Continuous delivery with Python	Cris Medina
#165	2018-06-08	Python and the blockchain	Stuart Farmer
#164	2018-06-01	Python in Brain Research at the Allen Institute	Panelists

Show Links

PythonBytes['headlines'] Episodes Submit news Courses

Sponsored by DataDog -- **pythonbytes.fm/datadog**

Brian #1: dataset: databases for lazy people

- **dataset** provides a simple abstraction layer removes most direct SQL statements without the necessity for a full ORM model - essentially, databases can be used like a JSON file or NoSQL store.
- A simple data loading script using **dataset** might look like this:

```
import dataset

db = dataset.connect('sqlite:///:memory:')

table = db['sometable']
table.insert(dict(name='John Doe', age=37))
table.insert(dict(name='Jane Doe', age=34, gender='female'))

john = table.find_one(name='John Doe')
```

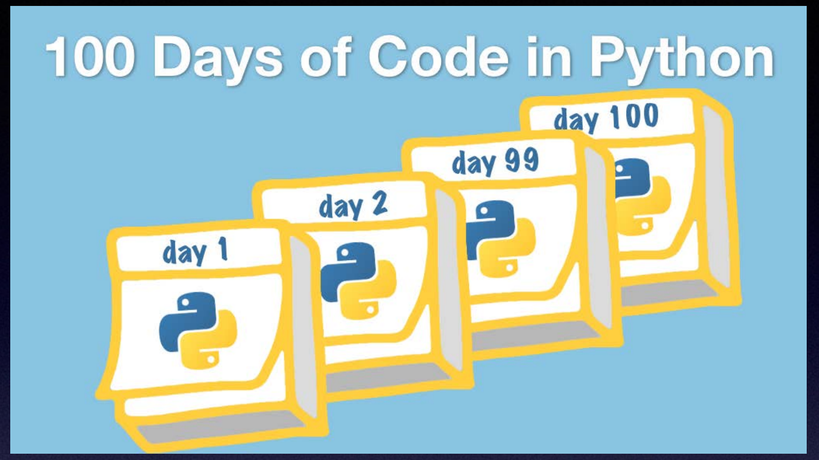
Michael #2: CuPy GPU NumPy

- A NumPy-compatible matrix library accelerated by CUDA
- How many cores does a modern GPU have?
- CuPy's interface is highly compatible with NumPy; in most cases it can be used as a drop-in replacement.
- You can easily make a custom CUDA kernel if you want to make your code run faster, requiring only a small code snippet of C++. CuPy automatically wraps and compiles it to make a CUDA binary
- PyCon 2018 presentation: Shohei Hido - CuPy: A NumPy-compatible Library for GPU
- Code example

Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?

Jump Start & Daily Coding Topics

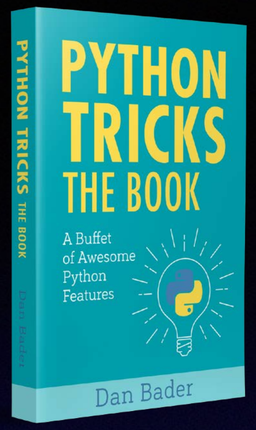


- Jump Start Class:
  - Starts at the beginning
  - Teaches how to use PyCharm through out the Jump Start course
- Hundred days of code class:
  - Covers a large area of topics
  - Helps keep you on track
- Work on things that are important to you

  ✳ There will be a drawing at the end of the night for three access keys for a free Jump Start class donated by Micheal Kennedy at Talk Python

Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?



Intermediate Skills

PYCODER'S WEEKLY

Weekly News Letter

Dan Bader

Improve Your Python Skills

Real Python

Tutorials & Interviews

Python Tricks

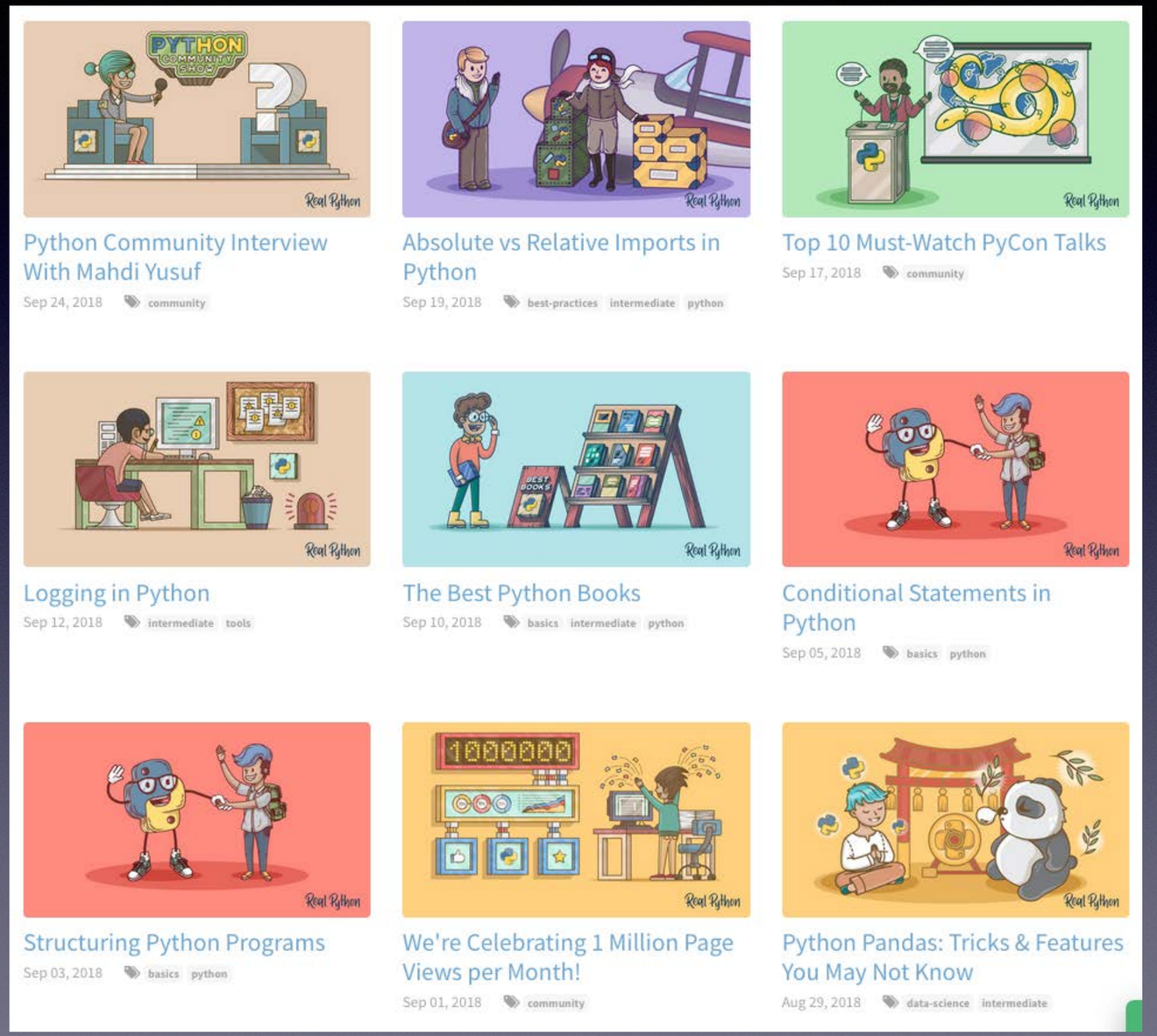
Get a short & sweet **Python Trick** delivered to your inbox every couple of days. No spam ever. Unsubscribe any time. Curated by yours truly.

```
# How to merge two dicts
# in Python 3.5+

>>> x = {'a': 1, 'b': 2}
>>> y = {'b': 3, 'c': 4}

>>> z = {**x, **y}

>>> z
{'c': 4, 'a': 1, 'b': 3}
```

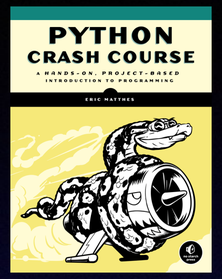
Tips E-mail

Tips: My Favorite Tutorial Source

- There are a lot of bad habits learned through web tutorials
- Make sure the source is reputable and has solid practices
- The podcasts mentioned earlier can help identify sources of good tutorials and classes



Multi-Sensory Learning



lynda.com

See

Relationship

Hear

Do

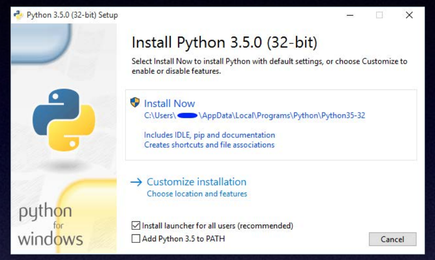
The way to learn is by using all your senses and through relationships (Mentoring)

Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?

Oh Crap! I Broke My App, Now What?

Good



Better

Homebrew Formulae

python

```
pip install image
```

```
Shell
# Python 2:
$ virtualenv env

# Python 3
$ python3 -m venv env
```

Freezing & Requirements files

```
$ pip freeze > requirements.txt
$ pip -r requirements.txt
```

Best

Homebrew Formulae

pyenv-virtualenv

```
pip install image
```

Freezing & Requirements files

```
$ pip freeze > requirements.txt
$ pip -r requirements.txt
```

Best - Best

Homebrew Formulae

pyenv-virtualenv

Homebrew Formulae

pipenv

Python dependency management tool

```
1. Development (bash)
(04_journal-2lhPJxuA) bash-3.2$ cat Pipfile
[[source]]
url = "https://pypi.org/simple"
verify_ssl = true
name = "pypi"

[packages]
pandas = "*"

[dev-packages]
pytest = "*"

[requires]
python_version = "3.7"
(04_journal-2lhPJxuA) bash-3.2$
```

No matter what, you need to learn how to use virtual environments at a minimum

Pyenv & Pipenv

If possible install Pyenv and Pipenv to a clean install of the OS via homebrew

Pyenv:

- Carefully read the advanced configuration section with regards to **shims**
  - If you don't follow the instructions pyenv will not work properly

Pipenv:

- Is available for macOS, Windows, and Linux
- Is the recommended method for installing packages per the PSF
- The are several good tutorials available
  - Written tutorial at 

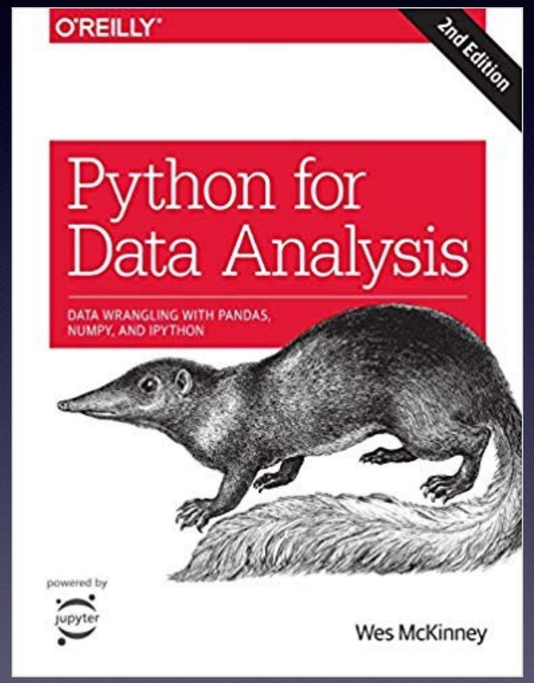
  - Video tutorial at 

Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?

Data Science / Analytics

Best Book



Dev Environment

jupyter

Notable Libraries

pandas

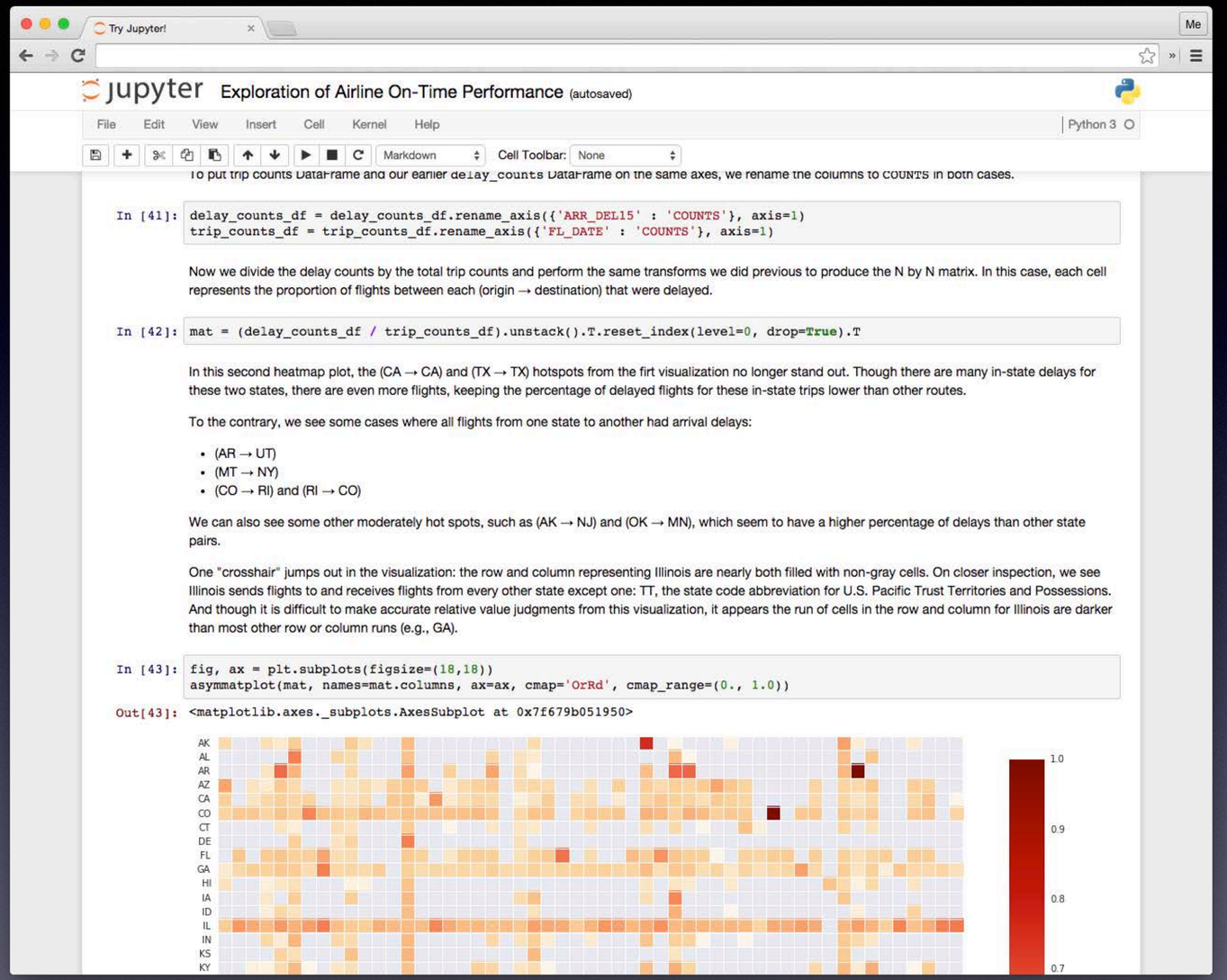
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

NumPy

matplotlib

Version 3.0.0

- Much faster than re-running datasets in an IDE
- Markdown capabilities allow for detailed notes
- The Jupyter cell only need to be ran a second time when changes are made to the code
- In-line visualization allow for proofing
- Can be used as part of a work flow to improve development time



Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?

Web Frameworks

Start with Flask

- It forces you to learn more about what goes on under the hood

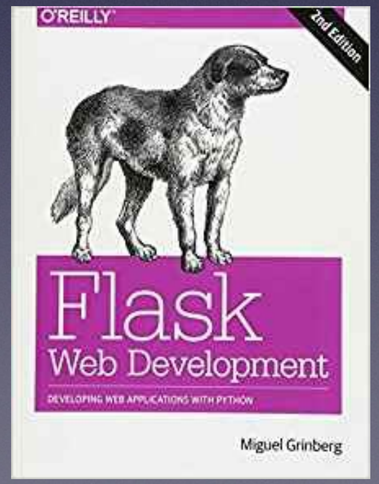
Then Learn Django

- Faster development, but does a lot under the hood for you

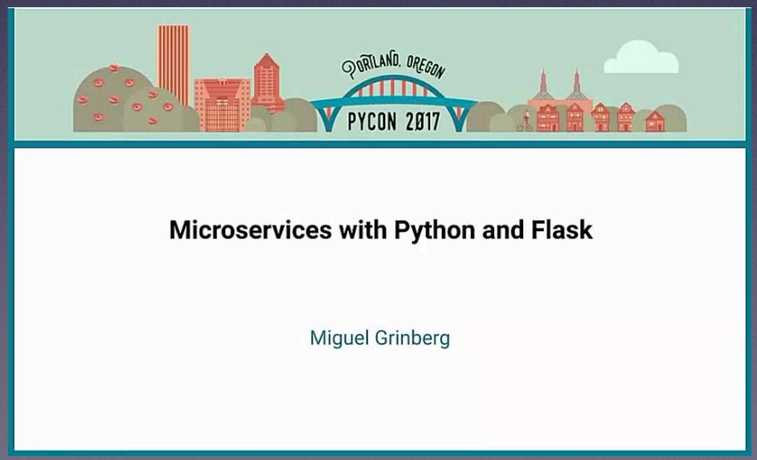
There are other frameworks available, such as, pyramid, but Flask and Django are the two most popular at the moment

Flask Resources

Books



PyCon



Video

lynda.com

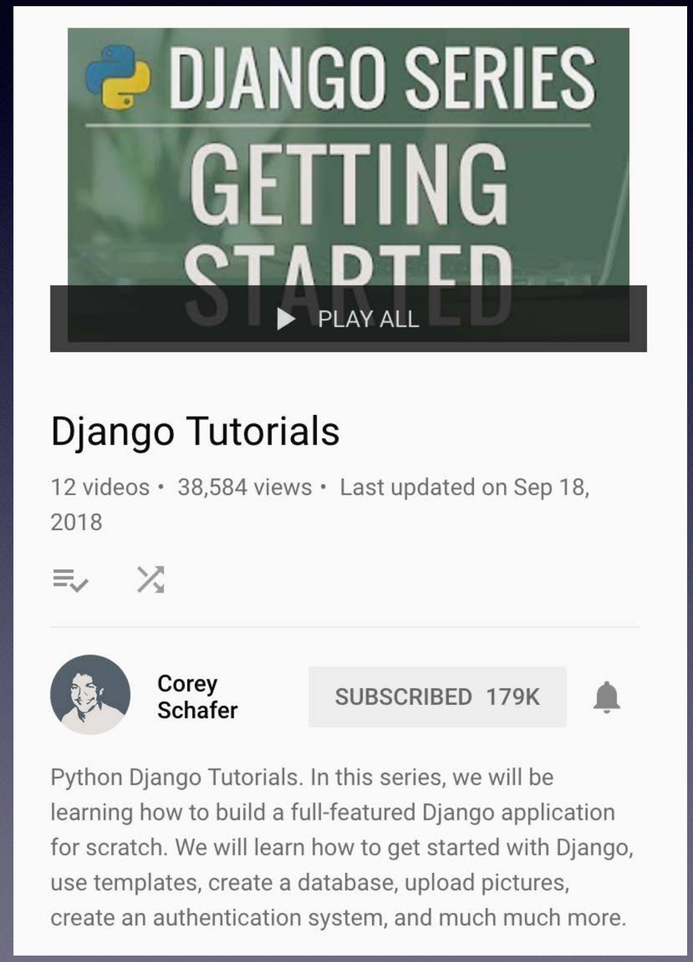
[PACKT]
PUBLISHING
www.PacktPub.com

Django

Books / Written Tutorials

The Official
Django Tutorial

Video



Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?

#2 Data Science

#1 WebApps

DevOps

Software Testing

Hacking/InfoSec

Software Prototyping

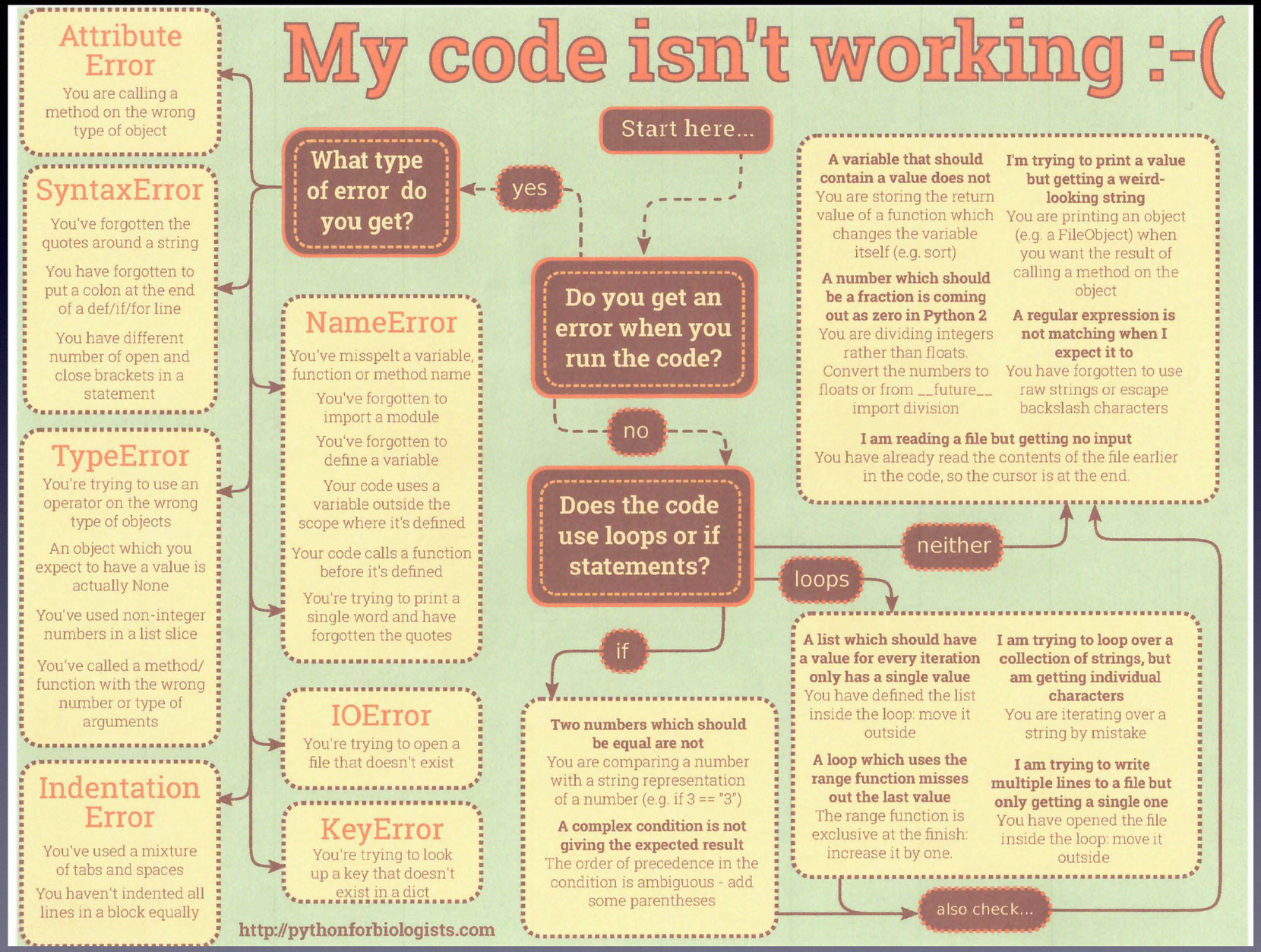
Network Programming

Business

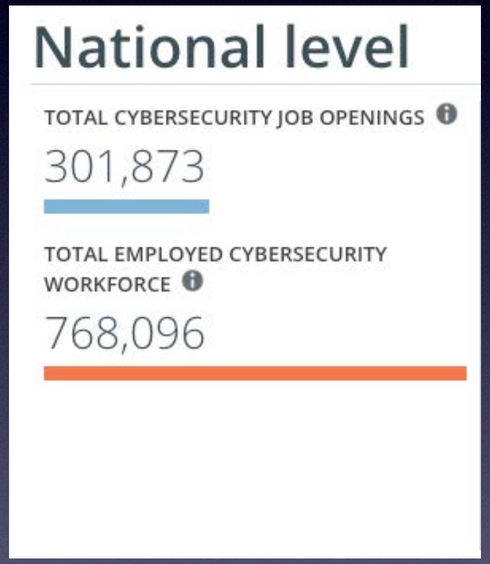
Career Enhancement

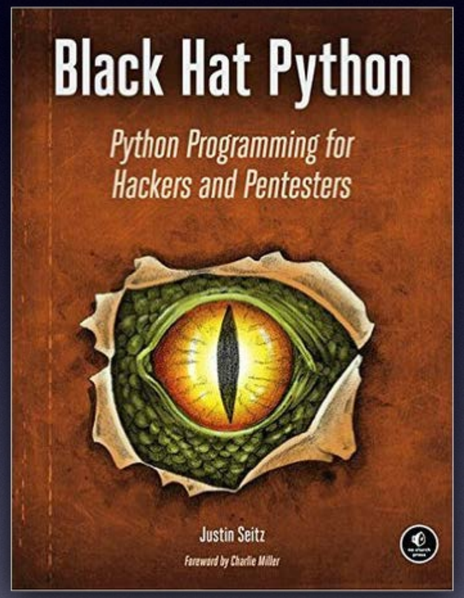
Source: JetBrains 2017 Developer Survey with supplemented Cybrary Data

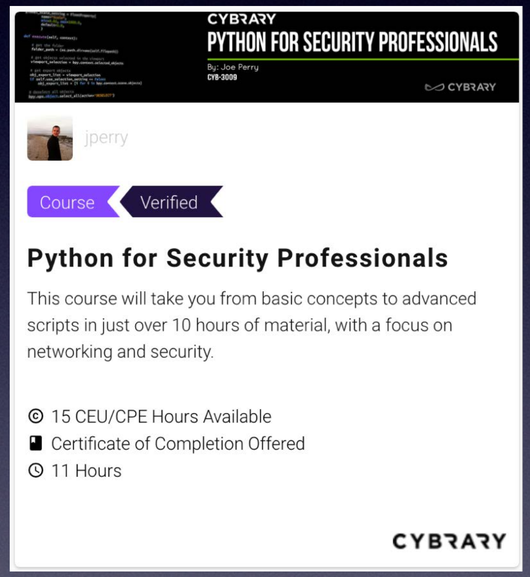
Look to Non-IT Fields for Resources



Security & Networking







Source: https://www.cyberseek.org/heatmap.html, last accessed on 2018-10-18

Tips: What I Wish Someone Had Told Me…

1. What should I know before I get started?
2. Where should I start if I struggled with programming in the past or not sure I want to commit?
3. Do not use a full featured IDE until after I finished the first book or two
4. Where can I find answers for things the tutorials leave out?
5. Where can I learn more about Python while I work through this giant book?
6. Is there a faster way to get up and running and what should I do daily?
7. I'm done with the first book(s), where can I find tutorials that are smaller than a book?
8. Oh crap! I broke my app by updating Python, now what?
9. I work with a lot of data, what can I do with Python?
10. Python GUI's look like the 1980's, what can I do with the web?
11. I'm not a programmer, what else can I do with Python?
12. Any last minute tips?

Tips: Final Thoughts

Databases:

- Use the database you plan on running in production during development
- Fight the urge to use sqlite
- PostgreSQL & the PostgreSQL app on macOS are good options
- Psycopg2 is the best adaptor I have found

Coming from a C based language that uses { }

- If you can stick with Python, that cold feeling down your back about not using { } will go away in about a month

When other people don’t understand how cool your code is, like your boss

- Let it go, you know you did a great job because it looks easy to an outsider

MOOC’s

- Don’t forget classes like those on EdX

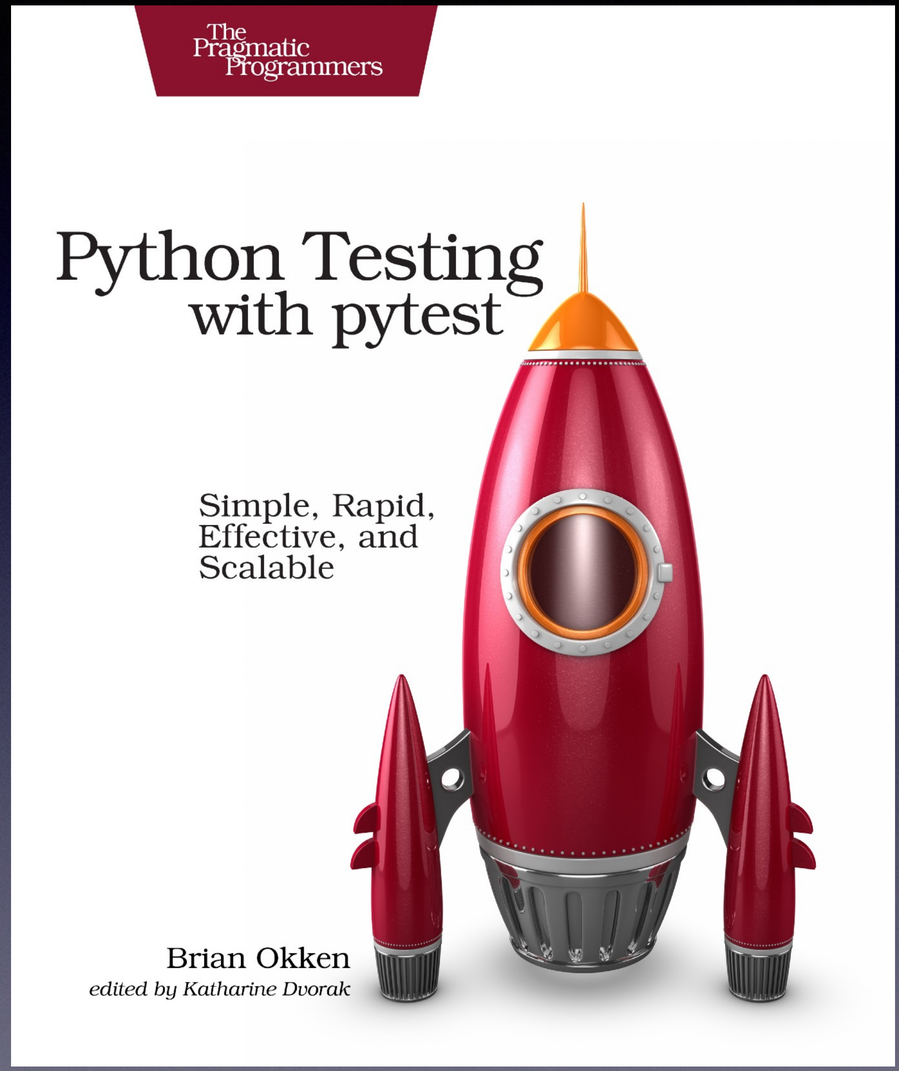
“I have such a stressful job that the only way I can get it out of my mind is by running hard.”

–Alan Turing

Bonus Material

Testing

- Unit Test is fine
- But PyTest seems Better
- Brian Okken's book was easy to understand and short enough to be helpful



Contact Information

Dennis P. Meek

e-mail: dennis.meek@gmail.com
LinkedIn: linkedin.com/in/dennis-meek-360a40
GitHub: github.com/DennisMeek

“Never stop learning, do meaningful work, and never quit.”