Learn Python

Here's a page about **Learning Python** if you are new to it.

Learning Python

The first key is **Don't Panic.** You can learn most by hacking.
There's so much out there: do not fret. This is a curated list of resources that we think will help you with Python at Veracode. We are taking care not to put everything and the kitchen sink. Even so, you can pick and choose from this page! You're going to do just fine 🙂

If you haven't installed Python yet, go to:

> HOWTO - install Python and use virtualenv

It's also a good idea to debunk the myths you've heard about Python.

> PayPal Engineering Blog: 10 Myths of Enterprise Python (5-10 minute read)

Python community at Veracode

We love you and we do not bite.

- Slack: join **#python**
- Python Distribution list: email Michael Floering or dl-python@veracode.com to get on it

Cheat sheets/quickstarts

Keep these handy.

- The official docs! https://docs.python.org/2/
- Learn X in Y Minutes: Python can get you up to speed on basics *very fast*. Can also be a cheat sheet.
- Flow-chart to help you get past common Python error messages

Python 2 or 3?

I think most Python developers here at Veracode dream about being able to use Python 3. For various reasons, most teams are stuck on Python 2.7. (Further reading if you want to know why the fissure between 2.7 and 3.) So, most of the links below assume Python 2.7 rather than Python 3. It's up to you, how you want to learn. But your Veracode projects will probably use Python 2.7.

Courses and books

If you like **taking online courses with a group**, Unknown User (sgray) says this one is good:

- **Coursera: An Introduction to Interactive Programming in Python** (free)

If you like **self-guided** courses, try one of these:

- RealPython: "Real Python teaches programming and web development through hands-on, interesting examples that are useful and fun!" (paid)
- or **Dive into Python (the classic).** (Note that Dive into Python 3 only covers new & advanced features in Python 3 - not the language as a whole. Don't start there.) (free)
- or Google's Python Introduction course (free)
- or How to Think Like a Computer Scientist: Learning with Python, Interactive Edition (free)

A good IDE will help you be more productive!

- Most of us prefer JetBrains PyCharm. PyCharm does incredible things with your Python code. You may not even miss the compiler, once you get to know it 😉
  - There's **a free Community Edition you can use immediately.** We have Professional Edition licenses, so get your hands on one of those as soon as you can!
  - Veracode Python Guide: PyCharm Setup Notes

Interactive tools for understanding Python better

Please try at least one of these out. Interactive tools can really deepen your understanding and accelerate your learning. A huge strength of Python is its interactivity ... Use it!

- **pythontutor.com** (thanks Abhishek Raju!)
  - "Python Tutor, created by Philip Guo, helps people overcome a fundamental barrier to learning programming: understanding what happens as the computer executes each line of a program's source code."
  - It's amazing.
- Cool Tools for Python - ipython
- Use PyCharm's debugger, it is excellent. JetBrains PyCharm docs: Debugging

Scenario guide

You're probably going to use libraries to do a lot of things. When you have a new scenario and you are thinking *what should I use...* You should:

1. ask #python. We're here to help!
2. check Hitchhiker's Guide to Python: Scenario Guide for Python Applications
3. check Full Stack Python, also pretty comprehensive and full of useful context, advice, etc.

Testing

One of Python's biggest strengths is its testability. Test-Driven Development and Python go together happily.

1. If you're new to this topic ...
   1. start with this concise post, http://docs.python-guide.org/en/latest/writing/tests/#the-basics
   2. then try this comprehensive post. It's about unit testing with Python's Standard Library unit test package, but leads you up to introducing pytest (what we use widely at Veracode)
      https://www.jeffknupp.com/blog/2013/12/09/improve-your-python-understanding-unit-testing/
2. **Veracode wiki, pytest aka py.test resources and notes (Python testing)**

For intermediate and advanced Pythonistas

Are you intermediate or advanced? If you're not sure, one way to check:

- Ask yourself Python Interview Questions and see if you know them. If you don't, try and learn more 😀

Python at Veracode

- **HOWTO - Start a new Python project ready to build, test, and release here at Veracode**
- **Learn Python - Security.** Security is Priority 1 at Veracode, Python can help you learn more about it.
- **Style matters.** In a flexible langue like Python, style matters especially.
  - **Our Veracode Python Style Guide is an aggregation of popular Python style guides.**
    Why? When combining these two guides, we had to make some decisions to keep things consistent.
    It's a combination of ...

    1. Google Python Style Guide – perhaps the most widely-used Python style guide
    2. The Hitchhiker's Guide to Python – opinionated guide by the legendary Kenneth Reitz of requests fame

Fast and scalable Python

Concurrency

For completely out-of-the-box Python, concurrency is slightly complicated (because of the GIL). **However,** it's not some no-man's-land or something. You don't need to rediscover the solution on your own. The best single summary I've seen of (preferred) concurrency options –

https://www.paypal-engineering.com/2014/12/10/10-myths-of-enterprise-python/#python-lacks-concurrency

but also, is concurrency exactly what you want? Maybe it is, but maybe a worker-and-queue option is more like what you want...

Parallelism: distributed task queue

For Python projects that need some horizontal scale-out and fast performance, there's a good chance a worker-and-queue setup might work better for you (better than single-computer concurrency). There's no question: **Celery** is the way to go for worker-and-queue systems in Python --

http://docs.celeryproject.org/en/latest/

Map/Reduce and beyond (Functional Reactive Programming)

For really *really* great horizontal scalability, consider *Functional Reactive Programming*, such as with Apache Spark.

Functional Reactive Programming braindump

More special topics

- Unicode is pretty sad in Python 2.7. Fixed in Python 3, but some of our teams can't upgrade yet.
  Our teams have had to deal with the strangest edge cases (on Discovery, MARS ....).
    - Discovery's onboarding page about Unicode in Python: Unicode, Character Sets, and the Web
    - Unicode in Python, completely demystified (presentation; press spacebar to proceed through slides)
- Functional programming, list comprehensions, etc.
    - **Python List Comprehensions: Explained Visually**
    - Decorators & functional Python
- "Good to Great Python Reads" is a nice index of great Python reading (good to Ctrl+F for topics) (thanks Abhishek Raju)

Functions-as-a-Service (aka 'Serverless')

- AWS Lambda
- Flask-on-Lambda: Zappa
- Flask-like-on-Lambda (but AWS official): Chalice

Beautiful is better than ugly. **Explicit** is better than implicit. **Simple** is better than complex. **Complex** is better than complicated. **Flat** is better than nested. **Sparse** is better than dense. **Readability** counts. *Special cases* aren't special enough to break the rules. Although **practicality** beats purity. *Errors* should never pass silently. Unless **explicitly** silenced. In the face of *ambiguity*, **refuse** the temptation to guess. There should be **one** — and preferably only one — obvious way to do it. Although that way may not be obvious at first *unless you're Dutch*. **Now** is better than never. Although never is **often** better than *right* now. If the implementation is *hard* to explain, it's a **bad** idea. If the implementation is *easy* to explain, it *may* be a **good** idea. **Namespaces** are one *honking great* idea — let's do more of those!

Automatic list of related pages, labelled "python" or "pytest"

- Page:Writing tests with pytest
- Page:How-to run ios comparator tool
- Page:pytest aka py.test resources and notes (Python testing)
- Page:Discovery Development Environment - Some Useful Tools
- Page:[procedure] get a standard component included in the Job Manager

Start a Python project at Veracode!

HOWTO - Start a new Python project ready to build, test, and release