

Python For

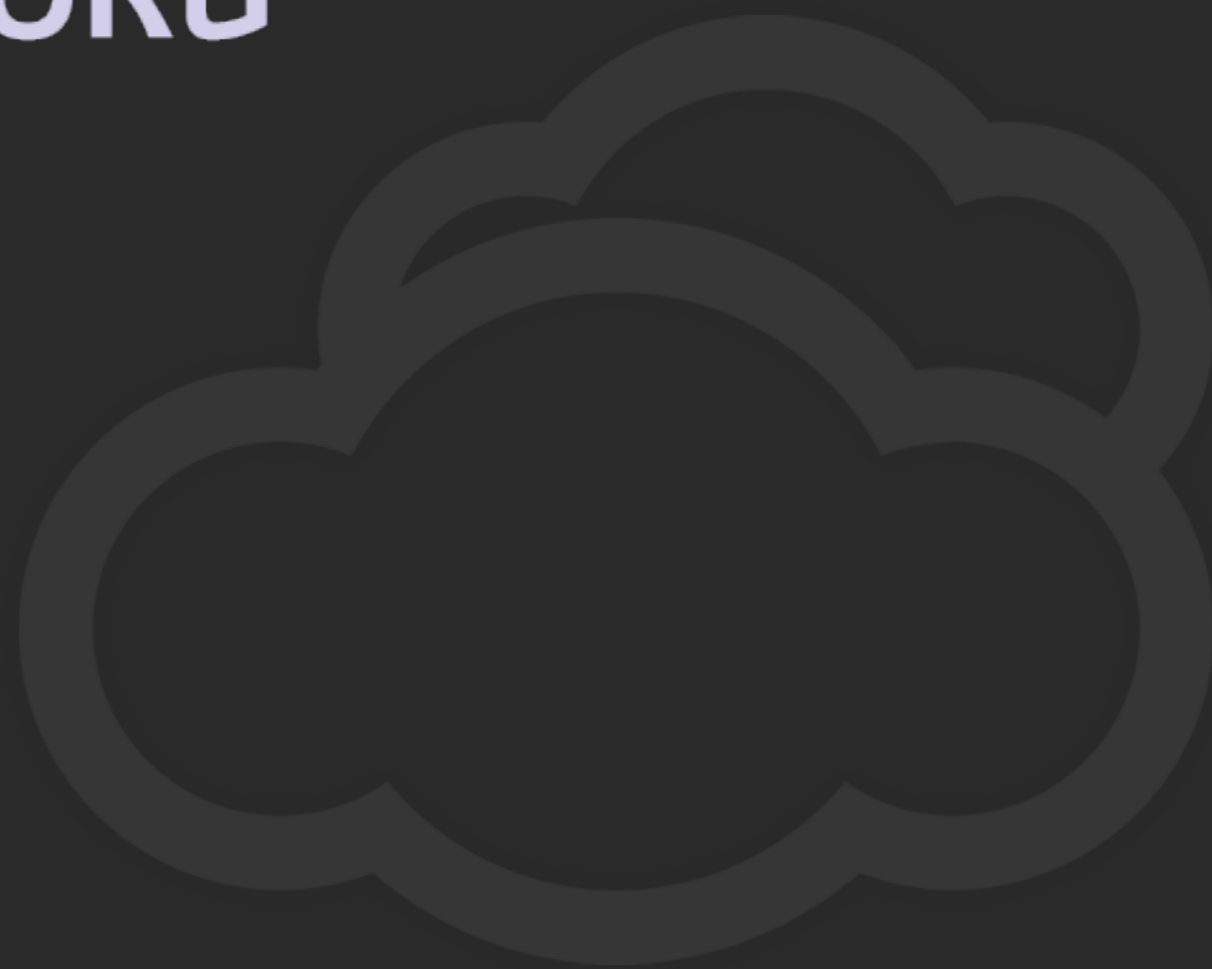


Kenneth Reitz

Hi.

@kennethreitz





Python Software



A decorative, symmetrical frame in a light gray color. It features ornate, swirling lines and small circular accents, resembling a stylized floral or scrollwork design. The frame is centered on the page and encloses the text.

Open Source

Requests

HTTP for Humans

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf-8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type": "User" ...}'
>>> r.json
{u'private_gists': 419, u'total_private_repos': 77, ...}
```



Httpbin.org

```
$ curl http://httpbin.org/get?test=1
```

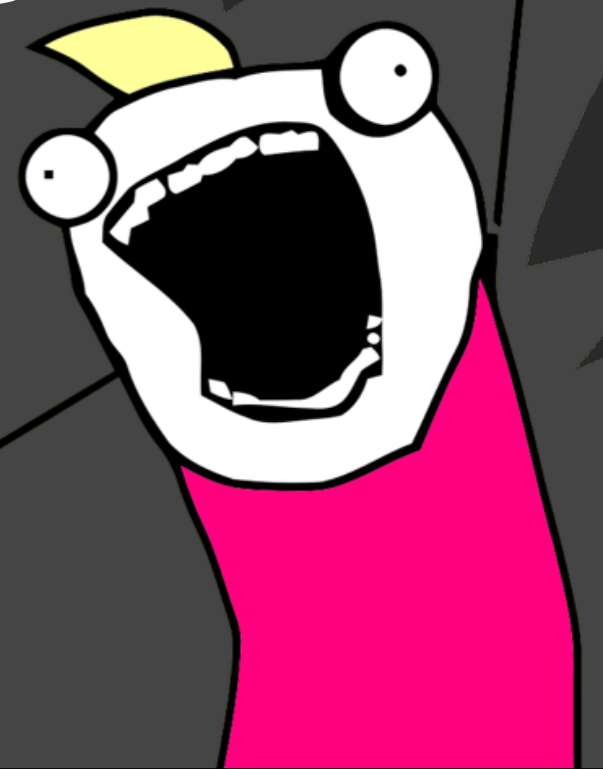
```
{  
  "url": "http://httpbin.org/get",  
  "headers": {  
    "Content-Length": "",  
    "Connection": "keep-alive",  
    "Accept": "*/*",  
    "User-Agent": "curl/7.21.4 ...",  
    "Host": "httpbin.org",  
    "Content-Type": ""  
  },  
  "args": {  
    "test": "1"  
  },  
  "origin": "67.163.102.42"  
}
```

Et Cetera

- Legit: Git Workflow for Humans
- Envoy: Subprocess for Humans
- Tablib: Tabular Data for Humans
- Clint: CLI App Toolkit
- Autoenv: Magic Shell Environments
- OSX-GCC-Installer: Provokes Lawyers

275+ More

Open Source
All The Things!



Build for Open Source

- Components become concise & decoupled.
- Concerns separate themselves.
- Best practices emerge (e.g. no creds in code).
- Documentation and tests become crucial.

A decorative, symmetrical frame in a light gray color. It features ornate, swirling lines and small circular accents, resembling a stylized floral or scrollwork design. The frame is centered on the page and encloses the word "Philosophy".

Philosophy

We share a dark

past.

Perl, Java, PHP,

ColdFusion,

Classic ASP, &c.

The Zen of Python

```
>>> import this
```

Beautiful is better
than ugly.

Explicit is better
than implicit.

Simple is better
than complex.

Complex is better
than complicated.

If the implementation is
hard to explain, it's a bad

(except PyPy)

There should be one—and
preferably only one—
obvious way to do it.

A decorative, ornate frame in a light gray color, featuring intricate scrollwork and floral motifs, surrounding the central text.

Welcome to
Paradise

Lies!

We know Ruby...

```
require 'net/http'
```

```
require 'uri'
```

```
uri = URI.parse('https://api.github.com/user')
```

```
http = Net::HTTP.new(uri.host, uri.port)
```

```
http.use_ssl = true
```

```
req = Net::HTTP::Get.new(uri.request_uri)
```

```
req.basic_auth('username', 'password')
```

```
r = http.request(req)
```

```
puts r
```

Python's net/http?

<http://url/lib/2>

Several hours later...

```
import urllib2
```

```
gh_url = 'https://api.github.com/user'
```

```
req = urllib2.Request(gh_url)
```

```
password_manager = urllib2.HTTPPasswordMgrWithDefaultRealm()  
password_manager.add_password(None, gh_url, 'user', 'pass')
```

```
auth_manager = urllib2.HTTPBasicAuthHandler(password_manager)  
opener = urllib2.build_opener(auth_manager)
```

```
urllib2.install_opener(opener)
```

```
handler = urllib2.urlopen(req)
```

```
print handler.read()
```

```
import re
```

```
class HTTPForcedBasicAuthHandler(HTTPBasicAuthHandler):
```

```
    auth_header = 'Authorization'
```

```
    rx = re.compile('(?:.*,)*[ \t]*([^\t]+)[ \t]+'  
                    'realm=([\"\\\'])(.*?)\\2', re.I)
```

```
    def __init__(self, *args, **kwargs):  
        HTTPBasicAuthHandler.__init__(self, *args, **kwargs)
```

```
    def http_error_401(self, req, fp, code, msg, headers):  
        url = req.get_full_url()  
        response = self._http_error_auth_reqed(  
            'www-authenticate', url, req, headers)  
        self.reset_retry_count()  
        return response
```

```
    http_error_404 = http_error_401
```

Admit it.
You'd leave and never come
back.

The Problem.

- Unclear which module to use in the first place.
- Prognosis seems to be urllib2, but the docs are useless.
- Worst API ever.

This is a serious
problem.
HTTP should be as
simple as a print
statement.

The Solution is Simple.

Build elegant tools to
perform these tasks.

Python needs more
Pragmatic Packages.

pra•gmat•ic |prag'matɪk|, *adj*:

Dealing with things sensibly and realistically in a way that is based on practical rather than theoretical considerations

Python For

Let's Break it Down.

What is HTTP at its core?

- A small set of methods with consistent parameters.
- HEAD, GET, POST, PUSH, PUT, PATCH, DELETE, &c.
- They all accept Headers, URL Parameters, Body/Form Data.

Urllib2 is Toxic.

- Heavily over-engineered.
- Abolishes most of PEP20.
- Docs are impossible to read.
- HTTP is simple. Urllib2 is not.
- Scares people away from Python.

Enter Requests.

HTTP for Humans.

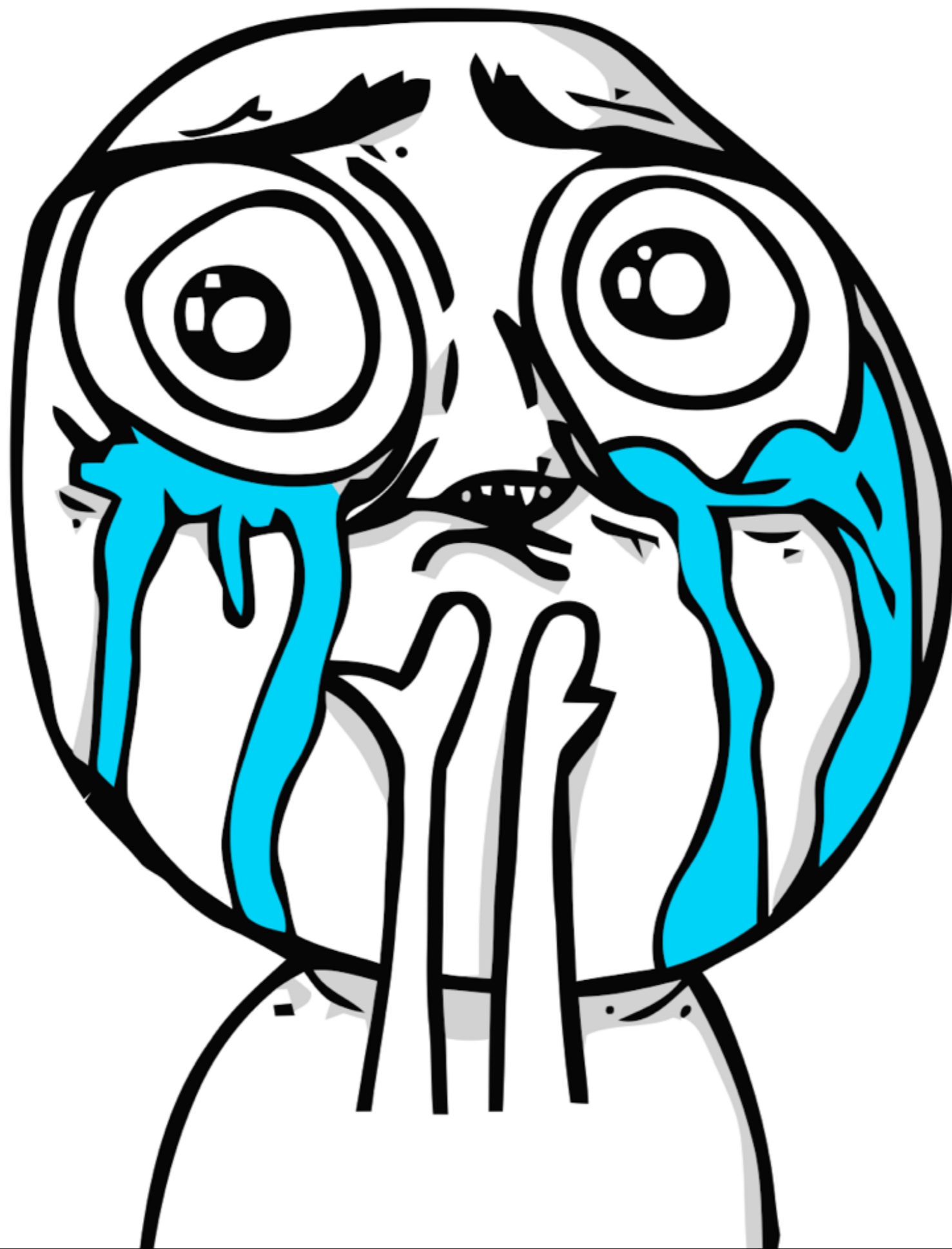
```
import requests
```

```
url = 'https://api.github.com/user'
```

```
auth = ('username', 'password')
```

```
r = requests.get(url, auth=auth)
```

```
print r.content
```



Achievement Unlocked!

- A small set of methods with consistent parameters.
- HEAD, GET, POST, PUSH, PUT, PATCH, DELETE, &c.
- They all accept Headers, URL Parameters, Body/Form Data.

Do this.

The Litmus Test

If you have to refer to the documentation every time you use a module, find (or build) a new module.

Fit the 90% Use

The API is all that matters.

Everything else is secondary.

I Mean Everything.

- Features.
- Efficiency.
- Performance.
- Corner-cases.
- Everything.

Write the README.

Pivot!

- At first, Requests was far from powerful.
- But, it deeply resonated with people.
- Features grew over time, but the API was never compromised.

Requests Today

- Cookies, sessions, content-iteration, decompression, file uploads, async i/o, keep-alive, connection pooling, callback hooks, proxies, OAuth, &c
- ~90,000,000+ downloads from PyPi.
- Google, Amazon, Microsoft, PayPal, Twilio, Mozilla, NPR, Twitter, &c.

Cool Story, Bro.

- We need better APIs.
- We want better APIs.
- It's worth your time as a developer.
- It's worth everyone's time as users.

A decorative, symmetrical frame in a light gray color. It features ornate scrollwork, floral motifs, and small circular accents. The frame is wider at the top and bottom, tapering slightly towards the sides, and it encloses the central text.

Barriers to Entry

File and System Operations

- `sys` | `shutils` | `os` | `os.path` | `io` modules
- Really difficult to run external commands.
- Blocks dev+ops folks from adopting Python.

Installing Python

- Just use the system Python?
- Python 2 or Python 3?
- Installer from Python.org?
- 32bit or 64bit?
- Build from source?
- Unix or Framework build?

There should be one—and
preferably only one—
obvious way to do it.

XML Hell

- etree annoys people.
- lxml is awesome, but can be difficult to install.

Date[time]s.

- Which module to use? Datetime? Date? Time? Calendar? Dateutil? 1.5?
- Timezones.
- The stdlib can generate but not parse ISO8601 dates.

Unicode.

Testing.

Installing Dependencies.

- Python-mysql (if you remember the name)
- Python Imaging Library.
- Mod_WSGI.
- lxml

Packaging & Dependencies

- Pip or easy_install?
- No easy_uninstall?
- Distribute vs Setuptools?
- Setuptools appears to be built into Python.
- Broken setup.py files.
- “Released” packages not in the Cheeseshop.

Packaging & Dependencies

- Pip or easy_install?
- No easy_uninstall?
- Distribute vs Setuptools?

Setuptools appears to be built into Python.

- Broken setup.py files.

- “Released” packages not in the CheeseShop.

Fixed!

Packaging & Dependencies

- Pip or easy_install?
- No easy_uninstall?
- Distribute vs Setuptools?

Setuptools appears to be built into Python.

- Broken setup.py files.

- “Released” packages not in the CheeseShop.

Fixed!
(mostly)

The Hitchhiker's Guide to Python.

<http://python-guide.org>

The Hitchhiker's Guide to Python

This opinionated guide exists to provide both novice and expert Python developers a best practice handbook to the installation, configuration, and usage of Python on a daily basis.

Stay Informed

Receive updates on new releases and upcoming projects.

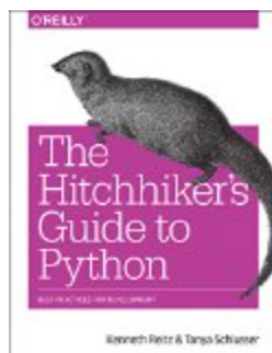
 Follow @kennethreitz

 Follow @kennethreitz

[Join Mailing List.](#)

O'Reilly Book

This guide is now available for pre-order in tangible book form!



All proceeds are being directly

The Hitchhiker's Guide to Python!

Greetings, Earthling! Welcome to The Hitchhiker's Guide to Python.

This is a living, breathing guide. If you'd like to contribute, [fork us on GitHub!](#)

This handcrafted guide exists to provide both novice and expert Python developers a best practice handbook to the installation, configuration, and usage of Python on a daily basis.

This guide is **opinionated** in a way that is almost, but not quite, entirely *unlike* Python's official documentation. You won't find a list of every Python web framework available here. Rather, you'll find a nice concise list of highly recommended options.

Let's get started! But first, let's make sure you know where your towel is.

Getting Started with Python

New to Python? Let's properly setup up your Python environment.

- [Picking an Interpreter](#)
 - [The State of Python \(2 vs 3\)](#)
 - [Recommendations](#)
 - [So... 3?](#)
 - [Implementations](#)
- Properly Install Python
 - [Properly Installing Python](#)
 - [Installing Python on Mac OS X](#)
 - [Installing Python on Windows](#)
 - [Installing Python on Linux](#)

Writing Great Python Code

This part of the guide focuses on the best-practices for writing Python code.

- [Structuring Your Project](#)

Python-Guide.org

The Hitchhiker's Guide to Python

- Documented best practices.
- Guidebook for newcomers.
- Reference for seasoned veterans.
- Don't panic & always carry a towel.

Best Practices

- Recommends distribute, pip, and virtualenv out of the box. Explicit installation directions for every OS.
- Instills a resistance to doctest.
- Teaches the use of `datetime.utcnow()`

Please Contribute!

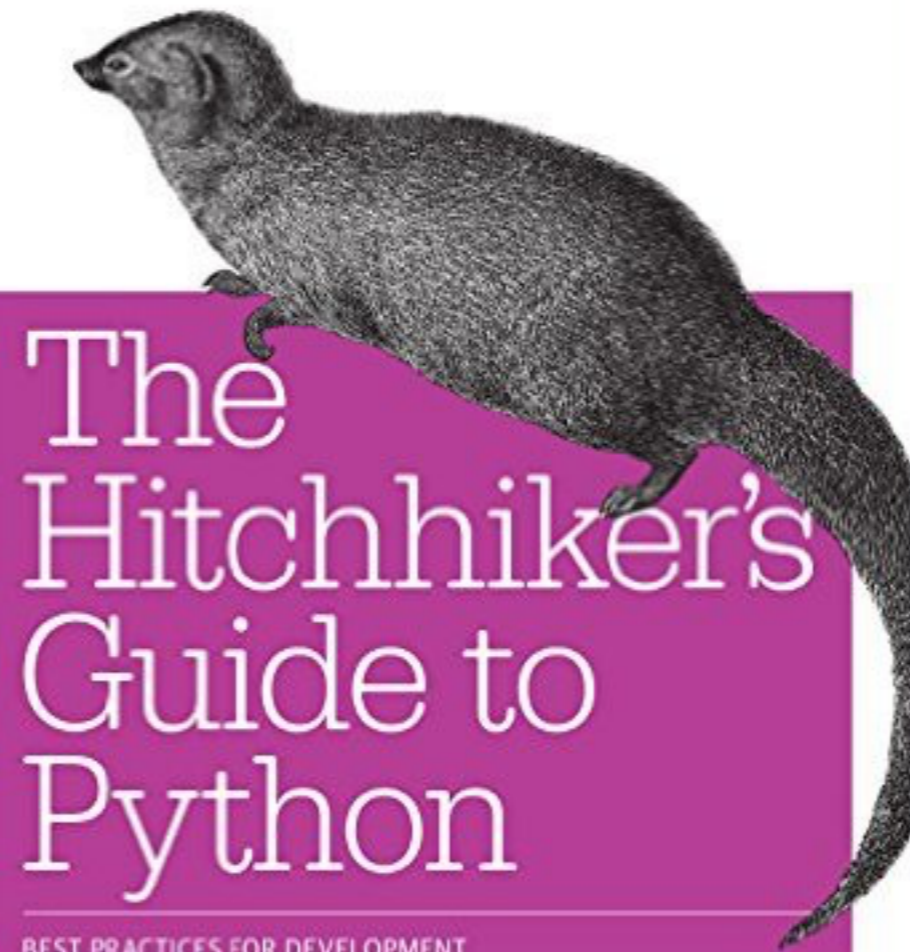
There's only one
rule...

There should be one—and
preferably only one—
obvious way to do it.

This Fixes...

- Makes Python more accessible, lowering the barrier to entry.
- Sets developers on the right path.
- Great reference guide for seasoned veterans.
- Practice what we preach.

O'REILLY®



The Hitchhiker's Guide to Python

BEST PRACTICES FOR DEVELOPMENT

Kenneth Reitz & Tanya Schlusser



django girls

MAINTHEFFE

STO

Simplify terrible APIs.

Document our
best-practices.

Questions?

github.com/kennethreitz

