



Opening the Flask

How an April Fools' Joke became a
Framework with Good Intentions

About Me

My name is Armin Ronacher

Part of the Poccoo Team

@mitsuhiko on Twitter/github/bb

<http://lucumr.poccoo.org/>

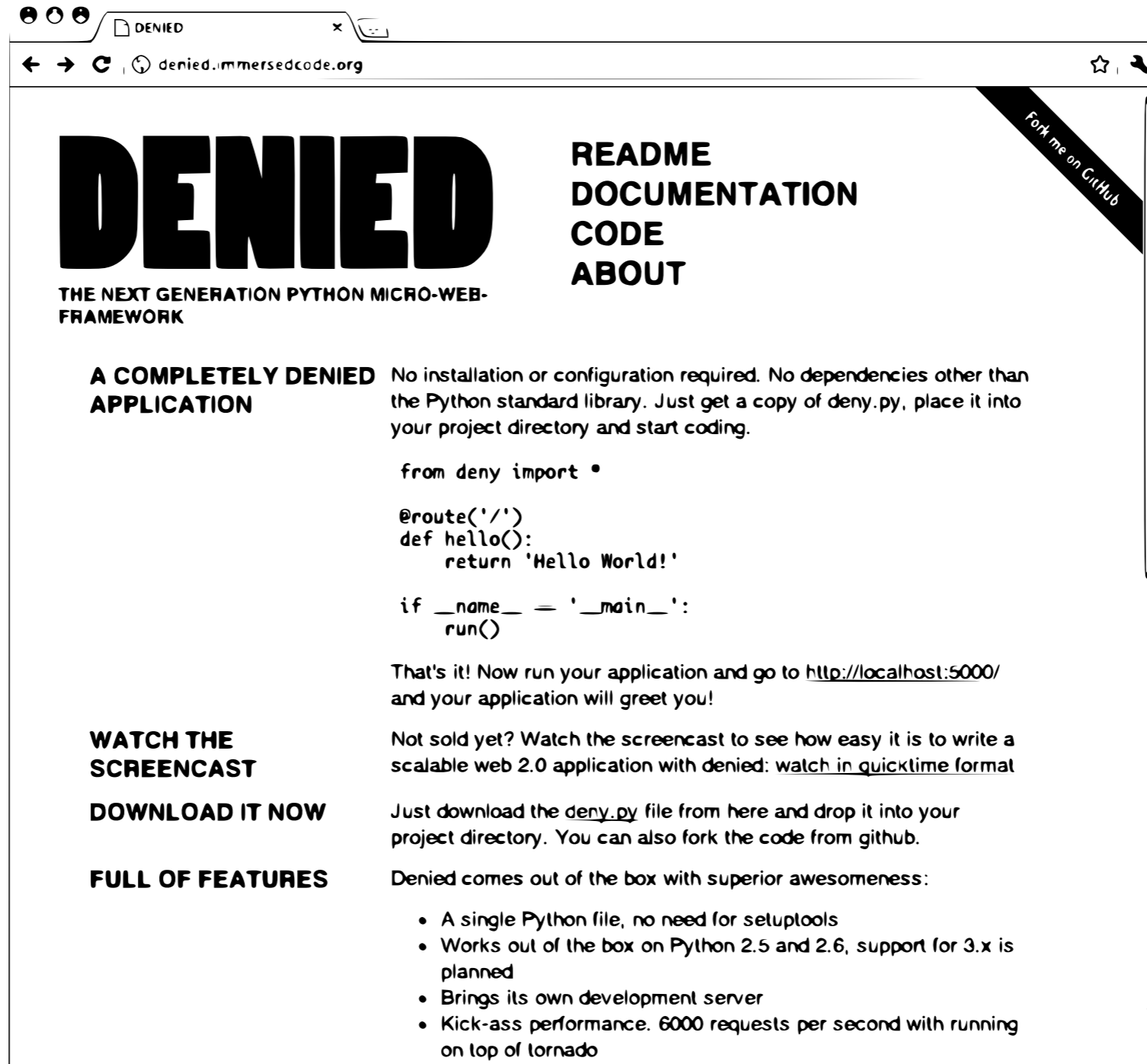


How it came to be

The story starts with an April Fool's Joke ...



It started as a Joke



DENIED
THE NEXT GENERATION PYTHON MICRO-WEB-FRAMEWORK

README
DOCUMENTATION
CODE
ABOUT

A COMPLETELY DENIED APPLICATION No installation or configuration required. No dependencies other than the Python standard library. Just get a copy of `deny.py`, place it into your project directory and start coding.

```
from deny import *
```

```
@route('/')  
def hello():  
    return 'Hello World!'
```

```
if __name__ == '__main__':  
    run()
```

That's it! Now run your application and go to <http://localhost:5000/> and your application will greet you!

WATCH THE SCREENCAST Not sold yet? Watch the screencast to see how easy it is to write a scalable web 2.0 application with denied: [watch in quicktime format](#)

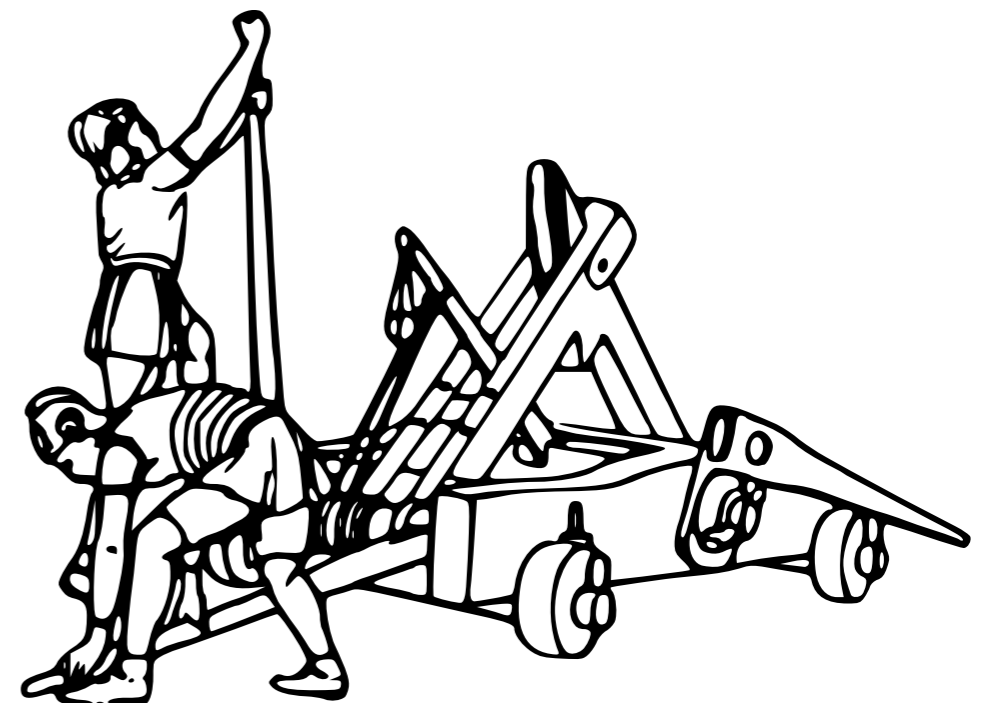
DOWNLOAD IT NOW Just download the [deny.py](#) file from here and drop it into your project directory. You can also fork the code from github.

FULL OF FEATURES Denied comes out of the box with superior awesomeness:

- A single Python file, no need for setuptools
- Works out of the box on Python 2.5 and 2.6, support for 3.x is planned
- Brings its own development server
- Kick-ass performance. 6000 requests per second with running on top of tornado

Motivation

- web2py / bottle / web.py
- “single file framework”
- “web scale”
- NoSQL
- screencast



The Story

- by Eirik Lahavre
- Entirely made up
- Jinja2 + Werkzeug zipped
- “Impressive Scaling Capabilities”
- RESTful



What that taught me

- Nobody has time to properly test the framework and read the code
- Marketing beats Quality
- Features don't matter
- Does not have to be new



Don't be evil™

- Just because nobody looks at tests it does not mean that there shouldn't be tests
- Marketing and good code quality do not have to be mutually exclusive

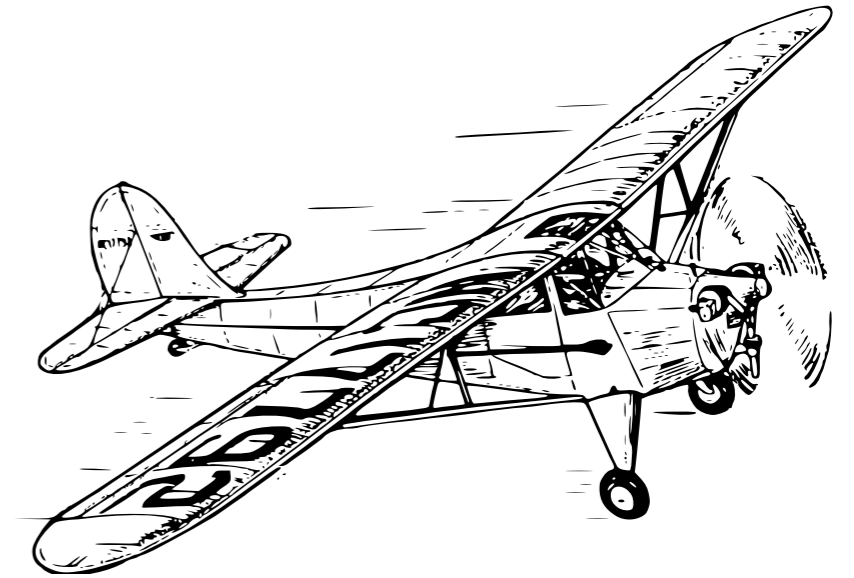
Inspiration

Why Flask looks the way it looks



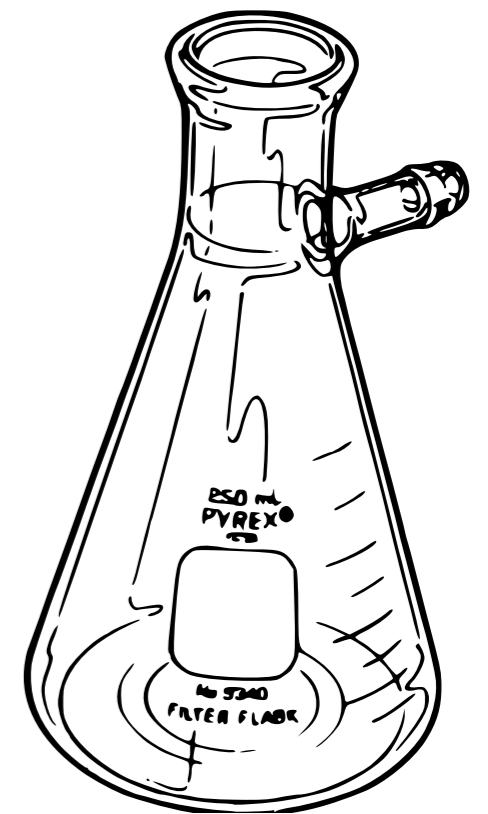
Good Intentions

- Be Honest
- Don't reinvent things
- Stay in Touch with Others



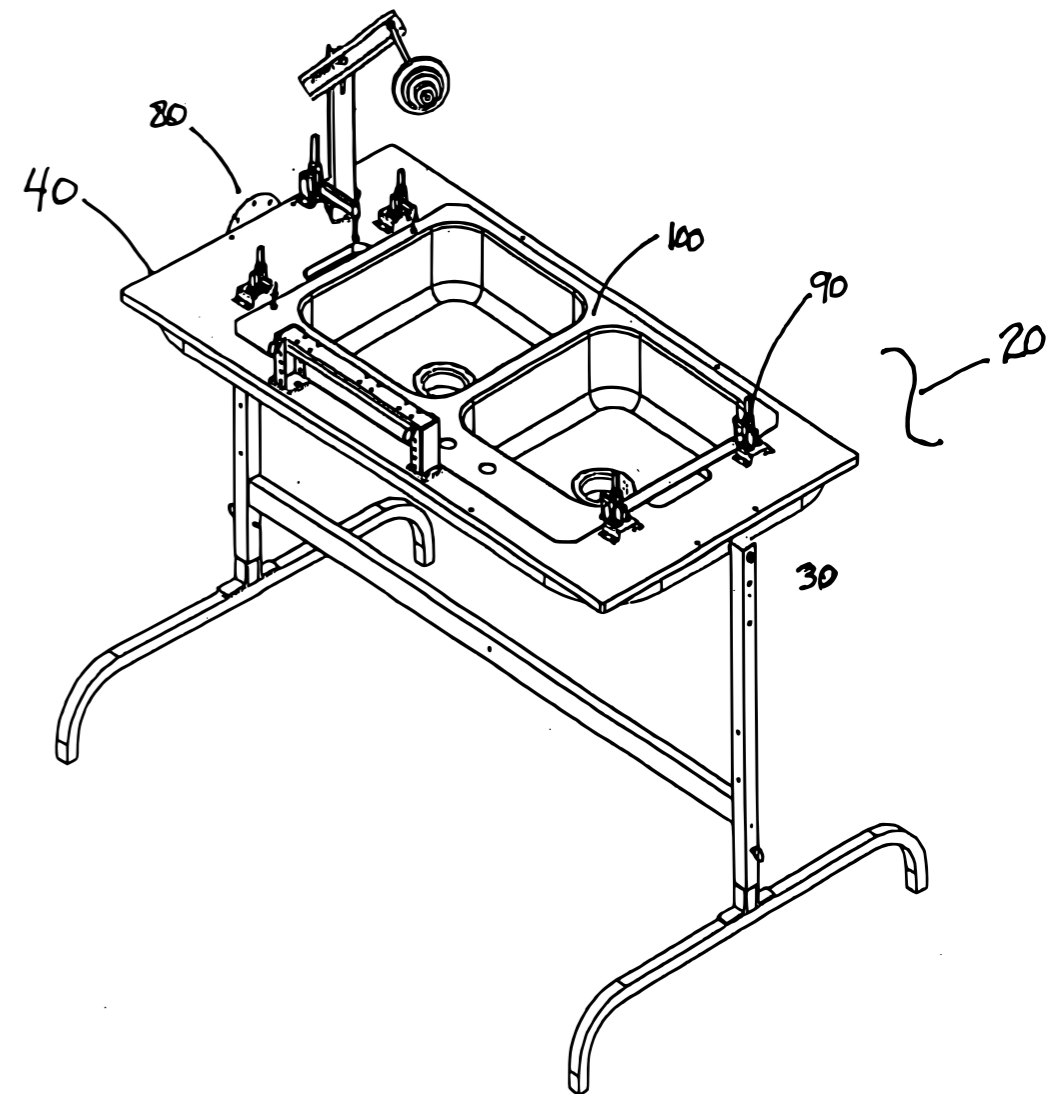
Enter Flask

- Wordplay on Bottle, *probably a mistake*
- based on Jinja2 and Werkzeug
- tons of documentation
- “best of breed” code



$\mu?$

- Flask depends on Werkzeug, Jinja2 and optionally Blinker
- There is also a Kitchensink release that includes Flask and deps to drop next to your Project.



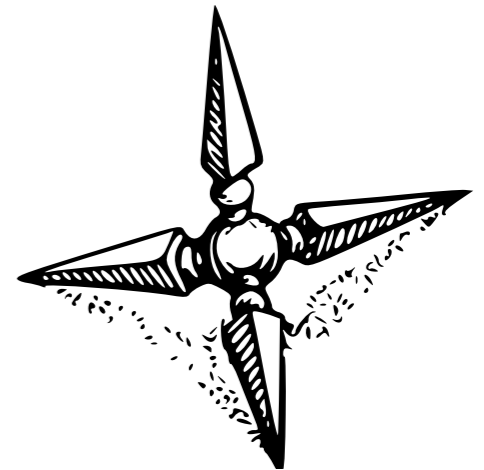
Results and Numbers

Where we are now



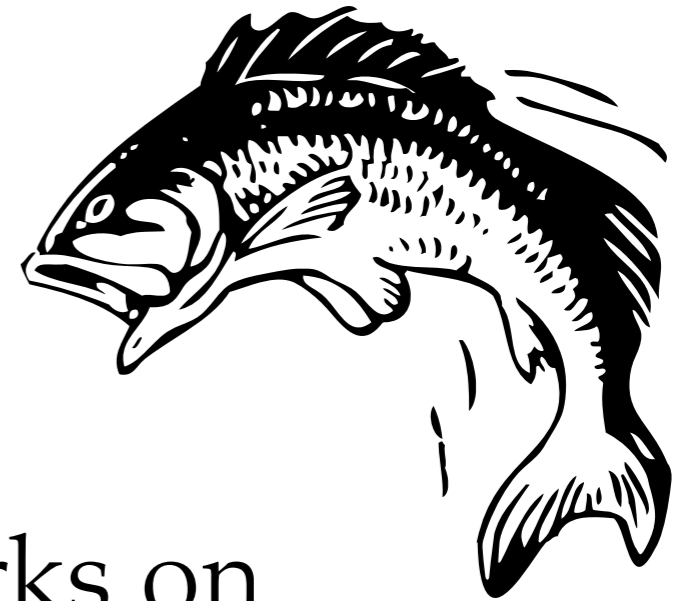
Some Numbers

- 800 LOC Code
- 1500 LOC Tests
- 200 A4 Pages of Documentation



Ecosystem

- over 30 extensions
- very active mailinglist
- over 700 followers and 100 forks on github – *yay*



Hello Flask

A minimal application in Flask



hello.py

```
from flask import Flask

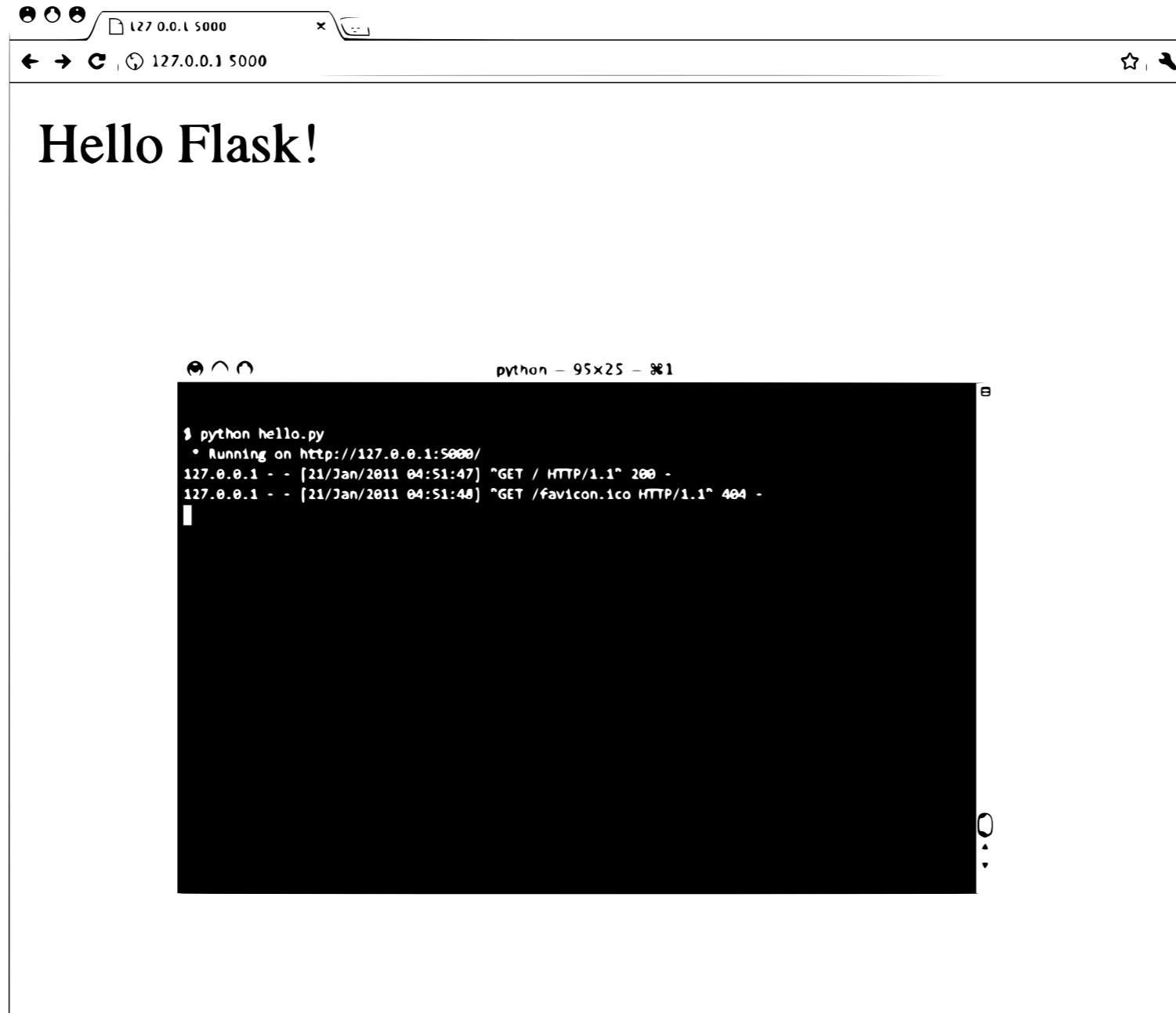
app = Flask(__name__)

@app.route('/')
def index():
    return 'Hello Flask!'

if __name__ == '__main__':
    app.run()
```



What it looks like



hello.py

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/', defaults={'name': 'Flask'})
@app.route('/<name>')
def index(name):
    return render_template('hello.html',
                           name=name)

if __name__ == '__main__':
    app.run()
```



hello.html

```
{% extends 'layout.html' %}
{% block title %}Greetings{% endblock %}
{% block body %}
    <h1>Hello {{ name }}!</h1>
{% endblock %}
```



layout.html

```
<!doctype html>  
<head>  
  <title>{% block title %}{% endblock %}</title>  
</head>  
<body>  
  {% block body %}{% endblock %}  
</body>
```



Flask's Design

Why things work the way they work



Context Locals

- either you have them everywhere or nowhere
- some things really need them or it becomes ugly (ORMs for instance)
- so we chose to embrace them



Fighting the Python

- No import time side effects
- Explicit application setup
- Circular imports
- Cached Imports



Why not like this?

```
from flask import route, run

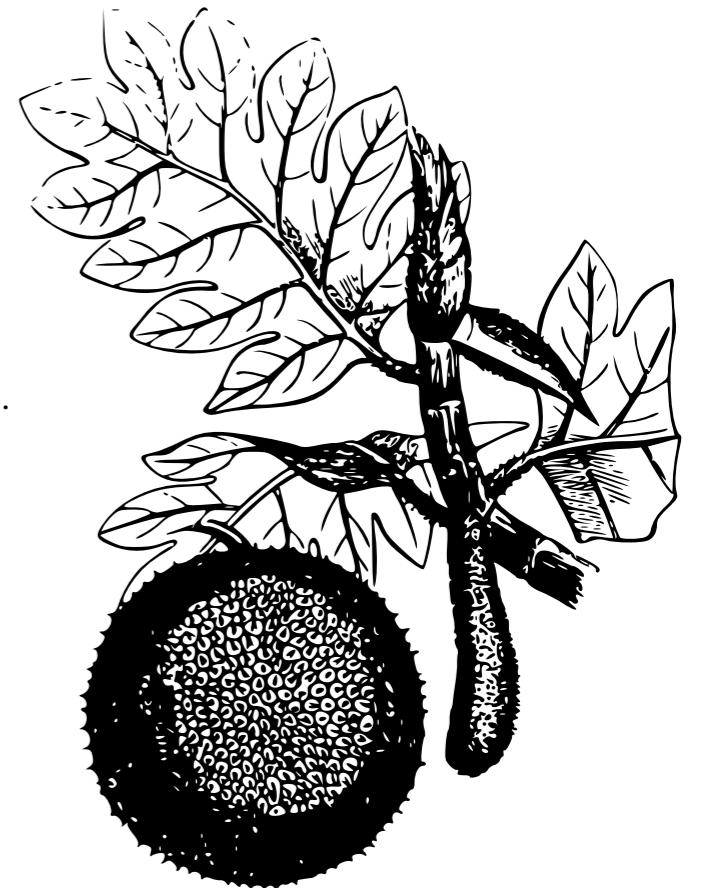
@route('/')
def index():
    return 'Hello Flask!'

if __name__ == '__main__':
    run()
```



Explicit Setup

- Applying WSGI middlewares
- More than one app
- Testing
- Create app in function



Import Order

- Larger projects: module seem to import in arbitrary order
- URL rules are attached to functions
- Routing system has to reorder them intelligently



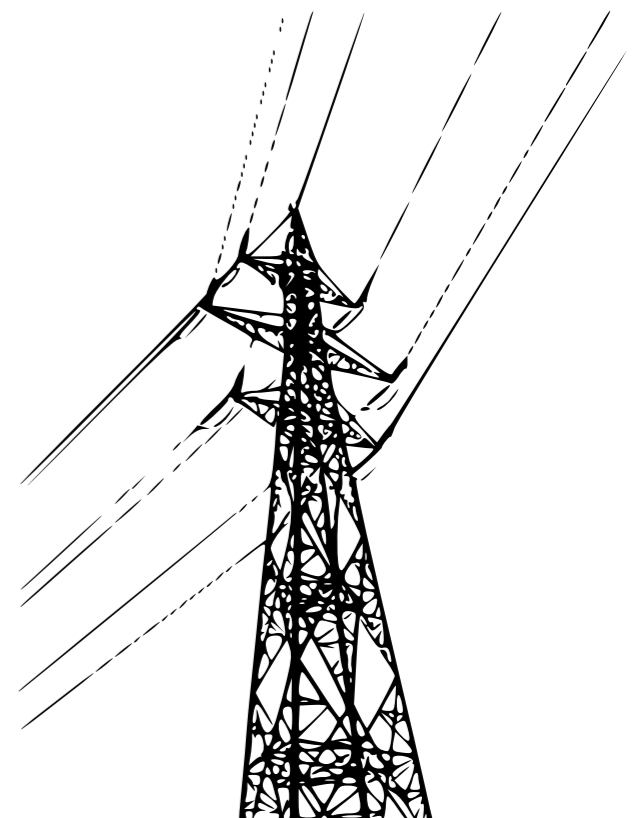
Aspects

How the design affects users and patterns



Power and Simplicity

```
def wsgi_app(self, environ, start_response):  
    with self.request_context(environ):  
        rv = self.preprocess_request()  
        if rv is None:  
            rv = self.dispatch_request()  
        response = self.make_response(rv)  
        return response(  
            environ, start_response)
```



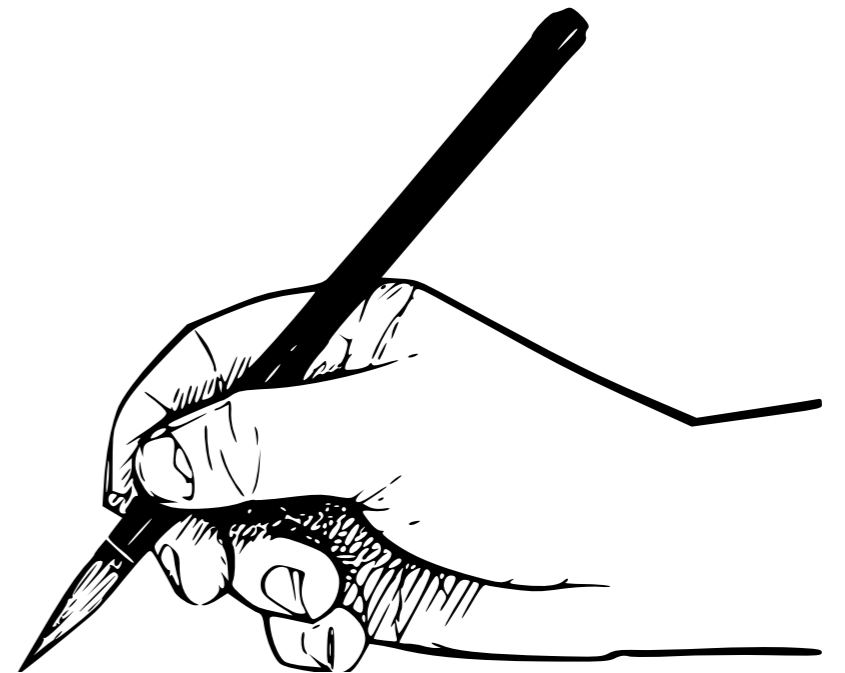
Simple Things Simple

```
import sqlite3
from flask import g

DATABASE = '/path/to/database.db'

@app.before_request
def before_request():
    g.db = sqlite3.connect(DATABASE)

@app.after_request
def after_request(response):
    g.db.close()
    return response
```



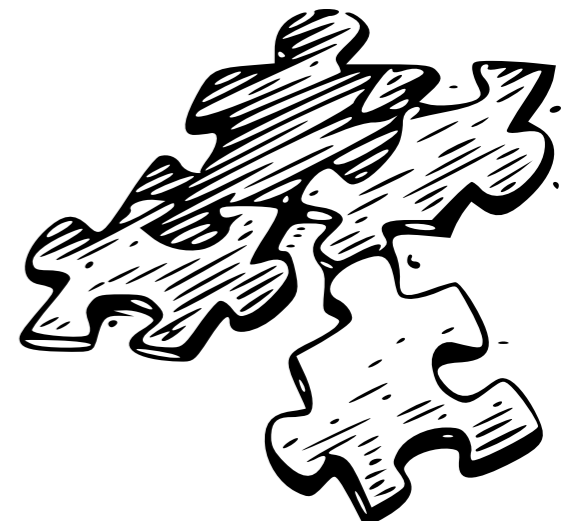
More

“but I want a pony”



Extensions

- Addons for Flask go into dedicated extensions.
- Core stays small
- SQLAlchemy, Babel, Genshi, CouchDB, MongoDB, etc.



Flask-SQLAlchemy

```
from flask import Flask
from flaskext.sqlalchemy import SQLAlchemy
```

```
app = Flask(__name__)
db = SQLAlchemy(app)
```

```
class User(db.Model):
    name = db.Column(db.String(40), primary_key=True)
    email = db.Column(db.String(100))
```

```
@app.route('/user/<name>')
```

```
def show_user(name):
```

```
    user = User.query.filter_by(name=name).first_or_404()
    return render_template('user.html', user=user)
```



Lessons Learned

because at the end of the day we're always wiser



The Important Ones

- Documentation matters
- Communication matters
- Heartbeat signals
- Consistency



Not a Mistake

- Nice documentation design makes you actually write documentation
- Documentation style for extensions
- Simple visual design is easy to adapt for extension developers



Thank you for listening and
your interest in Flask. Feel
free to ask *questions!*

Slides available at <http://lucumr.pocoo.org/talks/>

Contact me on twitter @mitsuhiko

or via mail: armin.ronacher@active-4.com

Legal

© Copyright 2011 by Armin Ronacher
<http://lucumr.pocoo.org/> — @mitsuhiko

Content licensed under the Creative Commons attribution-noncommercial-sharealike License. Images vectorized from images from Wiki Commons (<http://commons.wikimedia.org/>) and public domain sources. Individual copyrights apply.

Talk available for download at <http://lucumr.pocoo.org/talks/>