Python on the Web

Put the Fun Back into Development

Armin Ronacher

@mitsuhiko

lucumr.pocoo.org

What is it?

fun

object oriented

dynamic

.py

community focused

functional

strongly typed

widely used

Who uses it?


EVE
ONLINE


QUAKE LIVE


reddit

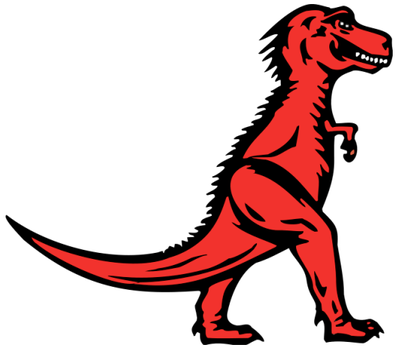

Disney


Quora


INST


DISQUS


EVE
ONLINE




YouTube
Broadcast Yourself™


EveryBlock


Google™


BATTLEFIELD 3
BATTLELOG


QUAKE LIVE

And those are just the websites …

Why?

Platforms?

all*

*Win32, OS X, Linux, BSD, Symbian, Android

License?

BSD-ish

Price?

Free

Is it fast?

fast enough

Is it reliable?

NASA / banks use it

Why not $lang?

Parsing an Apache Logfile

with Python

The Logfile Format

```
 -  - [20/Nov/2011:01:10:35 +0100] "GET /feed.atom HTTP/1.0" 200 2
47 -  - [20/Nov/2011:01:10:49 +0100] "GET /feed.atom HTTP/1.1" 304
5 -  - [20/Nov/2011:01:10:50 +0100] "GET /2008/1/23/no HTTP/1.0" 4
11 -  - [20/Nov/2011:01:10:50 +0100] "GET /feed.atom?_qt=data HTTP/
```

```
with open('/var/log/apache2/access.log') as f:
    for line in f:
        ...
```

```
import re



with open('/var/log/apache2/access.log') as f:
    for line in f:
        match = re.search(r' "\w+ (.*?) HTTP/', line)
        if match is None:
            continue
        print match.group(1).split('?')[0]
```

```
import re
from collections import defaultdict


counts = defaultdict(int)

with open('/var/log/apache2/access.log') as f:
    for line in f:
        match = re.search(r' "\w+ (.*?) HTTP/', line)
        if match is None:
            continue
        counts[match.group(1).split('?')[0]] += 1
```

```
import re
from collections import defaultdict


counts = defaultdict(int)

with open('/var/log/apache2/access.log') as f:
    for line in f:
        match = re.search(r' "\w+ (.*?) HTTP/', line)
        if match is None:
            continue
        counts[match.group(1).split('?')[0]] += 1


for url, count in counts.items():
    print '%s (%d times)' % (url, count)
```

```
import re
from collections import defaultdict
from heapq import nlargest

counts = defaultdict(int)

with open('/var/log/apache2/access.log') as f:
    for line in f:
        match = re.search(r' "\w+ (.*?) HTTP/', line)
        if match is None:
            continue
        counts[match.group(1).split('?')[0]] += 1

most_common = nlargest(5, counts.items(), key=lambda x: x[1])
for url, count in most_common:
    print '%s (%d times)' % (url, count)
```

```
import re
from collections import defaultdict
from heapq import nlargest

counts = defaultdict(int)

with open('/var/log/apache2/access.log') as f:
    for line in f:
        match = re.search(r' "\w+ (.*?) HTTP/', line)
        if match is None:
            continue
        counts[match.group(1).split('?')[0]] += 1

most_common = nlargest(5, counts.items(), key=lambda x: x[1])
for url, count in most_common:
    print '%s (%d times)' % (url, count)
```

Parsing an Apache Logfile

with Java

```
import java.io.FileInputStream;
import java.io.DataInputStream;
import java.io.BufferedReader;
import java.util.Map;
import java.util.HashMap;
import java.util.regex.Pattern;


class LogParser {
  public static void main(String[] args) {
    try {
      String filename = "/var/log/apache2/access.log";
      FileInputStream fstream = new FileInputStream(filename);
      DataInputStream in = new DataInputStream(fstream);
      BufferedReader br = new BufferedReader(new InputStreamReader(in));
      Map<String, Integer> counts = new HashMap<String, Integer>();
      try {
        Pattern pattern = Pattern.compile(" \"\\w+ (.*?) HTTP/");
        String line;

        while ((line = br.readLine()) != null) {
          Matcher m = p.matcher(line);
          if (!m.find())
            continue;
          String url = m.group(0).split("\\?")[0];
          if (counts.containsKey(url))
            counts.put(url, counts.get(url) + 1);
          else
            counts.put(url, 1);
        }
      } finally {
        fstream.close();
      }

      Map.Entry<String, Integer> items[] = counts.entrySet().toArray();
      items.sort(new Comparator() {
        int compareTo(Map.Entry<String, Integer> a,
                      Map.Entry<String, Integer> b) {
          return b.getValue().compareTo(a.getValue());
        }
      });

      for (int i = 0; i < Math.min(5, items.length); i++) {
        Map.Entry<String, Integer> item = items[i];
        System.out.println(item.getKey() + " (" + item.getValue() + " times)");
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

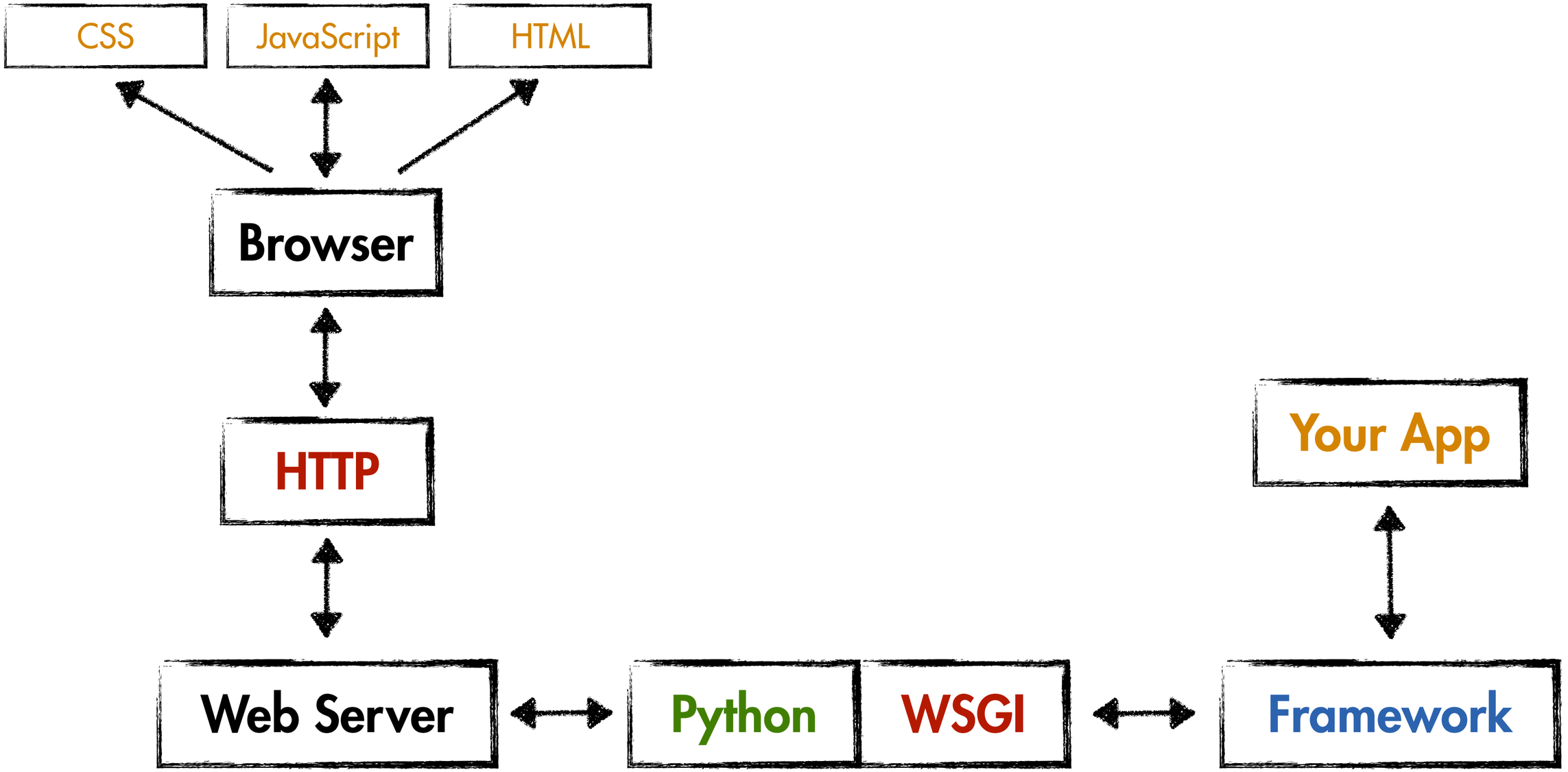
Productivity++

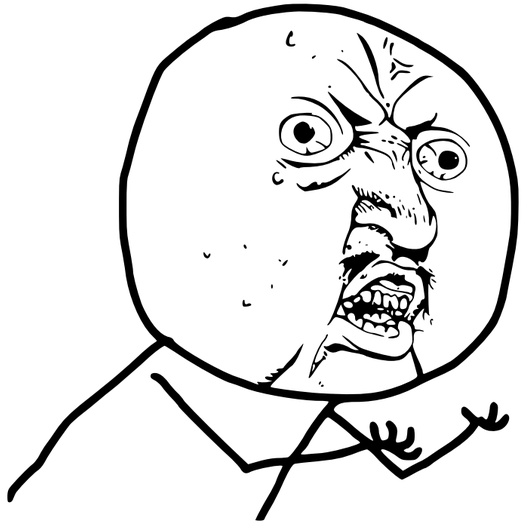
To The Web

The Frameworks

Django
Flask
Pyramid
Zope/Plone

The Stack


CSS
JavaScript
HTML
Browser
HTTP
Web Server
Python
WSGI
Framework
Your App



Y U NO SIMPLE?

Flask

http://flask.pocoo.org/

Step 0 | Install virtualenv

```
$ sudo apt-get install python-virtualenv
$ sudo easy_install virtualenv
```

For windows: http://bit.ly/easy-install-windows

Step 1 | Create Environment

```
$ virtualenv my-app
$ . my-app/bin/activate

> my-app\Scripts\activate.bat
```

Step 2 | Install Flask

```
$ pip install Flask
```

Step 3 | Hello World

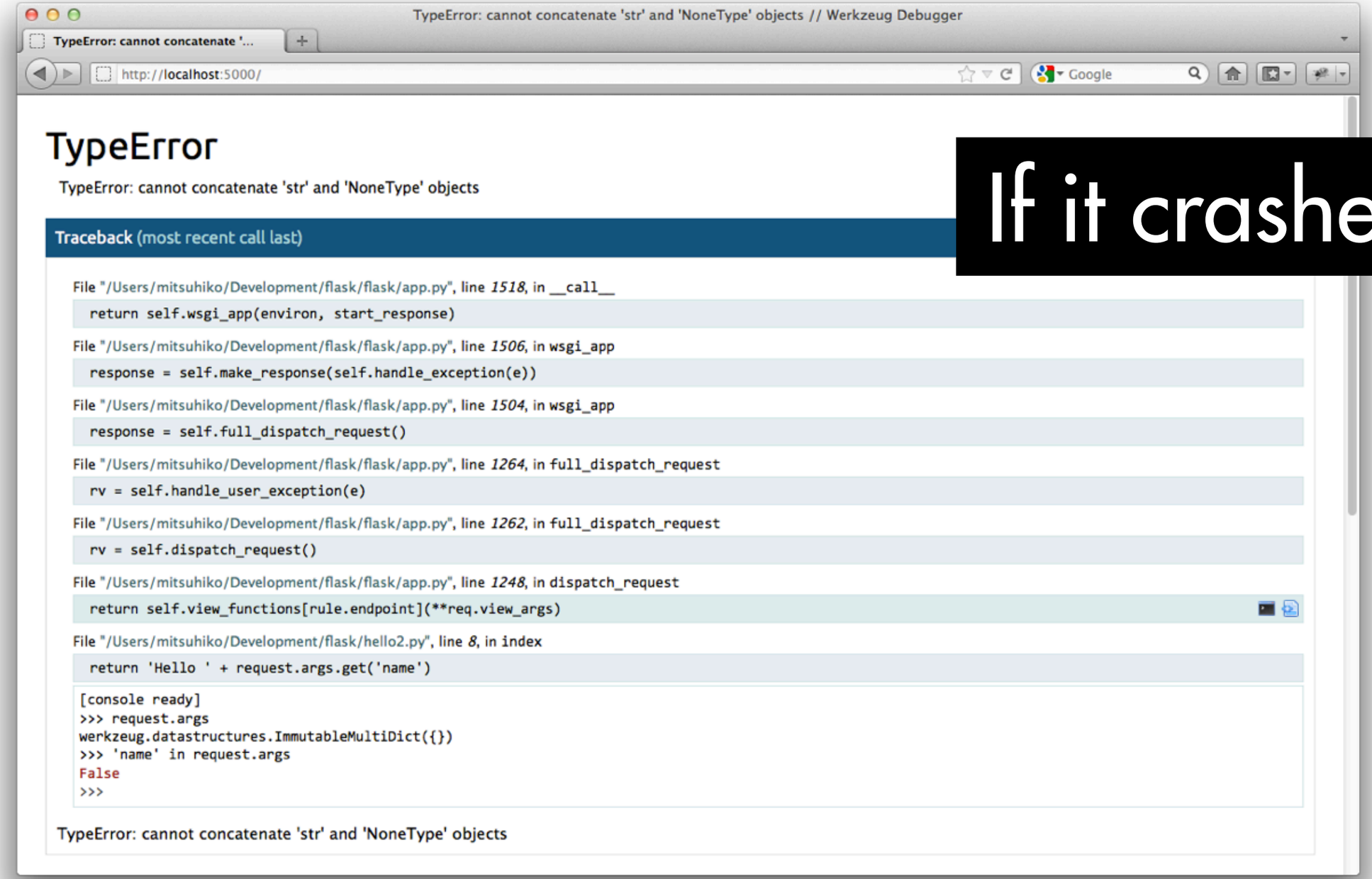
```
from flask import Flask
app = Flask(__name__)


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


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

Step 4 | Run!

```
$ python hello.py
 * Running on http://127.0.0.1:5000/
```

What we like:

- Develop locally
- Isolated environments
- Persistent execution
- Automatic code reloading
- Kick-ass debugging
- Logic / UI separation



If it crashes

Pastebin

Step 0 | Overview

- General Purpose Pastebin
- Users can sign in with Facebook
- Authenticated users can delete their entries
- Authenticated users can list their entries
  - *Flask, Flask-OAuth, Flask-SQLAlchemy*

Step 0 | Overview

```
/pastebin                       Project Folder
  /static                         Static Files
    /style.css
  /templates                      Templates
    /layout.html
    /new_paste.html
    /delete_paste.html
    /my_pastes.html
  /pastebin.py                    The Application
```

Step 1 | Install Extensions

```
$ pip install Flask-OAuth Flask-SQLAlchemy
```

Step 2 | Imports

```
from datetime import datetime
from flask import Flask, request, url_for, redirect, g, session, \
     abort, render_template
from flask.ext.sqlalchemy import SQLAlchemy
from flask.ext.oauth import OAuth
```

Step 2 | Imports

```
from datetime import datetime
from flask import Flask, request, url_for, redirect, g, session, \
     abort, render_template
from flask.ext.sqlalchemy import SQLAlchemy
from flask.ext.oauth import OAuth
```

Step 2 | Imports

```
from datetime import datetime
from flask import Flask, request, url_for, redirect, g, session, \
     abort, render_template
from flask.ext.sqlalchemy import SQLAlchemy
from flask.ext.oauth import OAuth
```

Step 2 | Imports

```
from datetime import datetime
from flask import Flask, request, url_for, redirect, g, session, \
     abort, render_template
from flask.ext.sqlalchemy import SQLAlchemy
from flask.ext.oauth import OAuth
```

Step 3 | Setup

```
app = Flask(__name__)
app.config.update(
    SQLALCHEMY_DATABASE_URI='sqlite:///pastebin.db',
    SECRET_KEY='development-key'
)
db = SQLAlchemy(app)
oauth = OAuth()
facebook = oauth.remote_app('facebook',
    base_url='https://graph.facebook.com/',
    request_token_url=None,
    access_token_url='/oauth/access_token',
    authorize_url='https://www.facebook.com/dialog/oauth',
    consumer_key='<consumer key here>',
    consumer_secret='<consumer secret here>',
    request_token_params={'scope': 'email'}
)
```

Step 3 | Setup

```
app = Flask(__name__)
app.config.update(
    SQLALCHEMY_DATABASE_URI='sqlite:///pastebin.db',
    SECRET_KEY='development-key'
)
db = SQLAlchemy(app)
oauth = OAuth()
facebook = oauth.remote_app('facebook',
    base_url='https://graph.facebook.com/',
    request_token_url=None,
    access_token_url='/oauth/access_token',
    authorize_url='https://www.facebook.com/dialog/oauth',
    consumer_key='<consumer key here>',
    consumer_secret='<consumer secret here>',
    request_token_params={'scope': 'email'}
)
```

Step 3 | Setup

```
app = Flask(__name__)
app.config.update(
    SQLALCHEMY_DATABASE_URI='sqlite:///pastebin.db',
    SECRET_KEY='development-key'
)
db = SQLAlchemy(app)
oauth = OAuth()
facebook = oauth.remote_app('facebook',
    base_url='https://graph.facebook.com/',
    request_token_url=None,
    access_token_url='/oauth/access_token',
    authorize_url='https://www.facebook.com/dialog/oauth',
    consumer_key='<consumer key here>',
    consumer_secret='<consumer secret here>',
    request_token_params={'scope': 'email'}
)
```

Step 3 | Setup

```
app = Flask(__name__)
app.config.update(
    SQLALCHEMY_DATABASE_URI='sqlite:///pastebin.db',
    SECRET_KEY='development-key'
)
db = SQLAlchemy(app)
oauth = OAuth()
facebook = oauth.remote_app('facebook',
    base_url='https://graph.facebook.com/',
    request_token_url=None,
    access_token_url='/oauth/access_token',
    authorize_url='https://www.facebook.com/dialog/oauth',
    consumer_key='<consumer key here>',
    consumer_secret='<consumer secret here>',
    request_token_params={'scope': 'email'}
)
```

Step 3 | Setup

```
app = Flask(__name__)
app.config.update(
    SQLALCHEMY_DATABASE_URI='sqlite:///pastebin.db',
    SECRET_KEY='development-key'
)
db = SQLAlchemy(app)
oauth = OAuth()
facebook = oauth.remote_app('facebook',
    base_url='https://graph.facebook.com/',
    request_token_url=None,
    access_token_url='/oauth/access_token',
    authorize_url='https://www.facebook.com/dialog/oauth',
    consumer_key='<consumer key here>',
    consumer_secret='<consumer secret here>',
    request_token_params={'scope': 'email'}
)
```

Step 4 | Database Schema

```
class Paste(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    code = db.Column(db.Text)
    pub_date = db.Column(db.DateTime)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

    def __init__(self, user, code):
        self.user = user
        self.code = code
        self.pub_date = datetime.utcnow()

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    display_name = db.Column(db.String(120))
    fb_id = db.Column(db.String(30), unique=True)
    pastes = db.relationship(Paste, lazy='dynamic', backref='user')
```

Step 4 | Database Schema

```
class Paste(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    code = db.Column(db.Text)
    pub_date = db.Column(db.DateTime)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

    def __init__(self, user, code):
        self.user = user
        self.code = code
        self.pub_date = datetime.utcnow()

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    display_name = db.Column(db.String(120))
    fb_id = db.Column(db.String(30), unique=True)
    pastes = db.relationship(Paste, lazy='dynamic', backref='user')
```

Step 4 | Database Schema

```
class Paste(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    code = db.Column(db.Text)
    pub_date = db.Column(db.DateTime)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

    def __init__(self, user, code):
        self.user = user
        self.code = code
        self.pub_date = datetime.utcnow()

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    display_name = db.Column(db.String(120))
    fb_id = db.Column(db.String(30), unique=True)
    pastes = db.relationship(Paste, lazy='dynamic', backref='user')
```

Step 4 | Database Schema

```
class Paste(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    code = db.Column(db.Text)
    pub_date = db.Column(db.DateTime)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

    def __init__(self, user, code):
        self.user = user
        self.code = code
        self.pub_date = datetime.utcnow()

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    display_name = db.Column(db.String(120))
    fb_id = db.Column(db.String(30), unique=True)
    pastes = db.relationship(Paste, lazy='dynamic', backref='user')
```

Step 4 | Database Schema

```
class Paste(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    code = db.Column(db.Text)
    pub_date = db.Column(db.DateTime)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

    def __init__(self, user, code):
        self.user = user
        self.code = code
        self.pub_date = datetime.utcnow()

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    display_name = db.Column(db.String(120))
    fb_id = db.Column(db.String(30), unique=True)
    pastes = db.relationship(Paste, lazy='dynamic', backref='user')
```

Step 4 | Database Schema

```
class Paste(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    code = db.Column(db.Text)
    pub_date = db.Column(db.DateTime)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

    def __init__(self, user, code):
        self.user = user
        self.code = code
        self.pub_date = datetime.utcnow()

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    display_name = db.Column(db.String(120))
    fb_id = db.Column(db.String(30), unique=True)
    pastes = db.relationship(Paste, lazy='dynamic', backref='user')
```

Step 4 | Database Schema

```
class Paste(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    code = db.Column(db.Text)
    pub_date = db.Column(db.DateTime)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

    def __init__(self, user, code):
        self.user = user
        self.code = code
        self.pub_date = datetime.utcnow()

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    display_name = db.Column(db.String(120))
    fb_id = db.Column(db.String(30), unique=True)
    pastes = db.relationship(Paste, lazy='dynamic', backref='user')
```

Step 5 | Authentication

```
@app.before_request
def check_user_status():
    g.user = None
    if 'user_id' in session:
        g.user = User.query.get(session['user_id'])


@facebook.tokengetter
def get_facebook_oauth_token():
    return session.get('fb_access_token')
```

Step 5 | Authentication

```
@app.before_request
def check_user_status():
    g.user = None
    if 'user_id' in session:
        g.user = User.query.get(session['user_id'])


@facebook.tokengetter
def get_facebook_oauth_token():
    return session.get('fb_access_token')
```

Step 5 | Authentication

```
@app.before_request
def check_user_status():
    g.user = None
    if 'user_id' in session:
        g.user = User.query.get(session['user_id'])
```

```
@facebook.tokengetter
def get_facebook_oauth_token():
    return session.get('fb_access_token')
```

Step 5 | Authentication

```
@app.route('/login')
def login():
    return facebook.authorize(callback=url_for('facebook_authorized',
        next=request.args.get('next') or request.referrer or None,
        _external=True))


@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('new_paste'))
```

Step 5 | Authentication

```
@app.route('/login')
def login():
    return facebook.authorize(callback=url_for('facebook_authorized',
        next=request.args.get('next') or request.referrer or None,
        _external=True))


@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('new_paste'))
```

Step 5 | Authentication

```
@app.route('/login')
def login():
    return facebook.authorize(callback=url_for('facebook_authorized',
        next=request.args.get('next') or request.referrer or None,
        _external=True))
```

```
@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('new_paste'))
```

Step 5 | Authentication

```
@app.route('/login')
def login():
    return facebook.authorize(callback=url_for('facebook_authorized',
        next=request.args.get('next') or request.referrer or None,
        _external=True))


@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('new_paste'))
```

Step 5 | Authentication

```
@app.route('/login/authorized')
@facebook.authorized_handler
def facebook_authorized(resp):
    next_url = request.args.get('next') or url_for('new_paste')
    if resp is None:
        return redirect(next_url)
    session['fb_access_token'] = (resp['access_token'], '')
    me = facebook.get('/me')
    user = User.query.filter_by(fb_id=me.data['id']).first()
    if user is None:
        user = User()
        user.fb_id = me.data['id']
        db.session.add(user)
    user.display_name = me.data['name']
    db.session.commit()
    session['user_id'] = user.id
    return redirect(next_url)
```

Step 5 | Authentication

```
@app.route('/login/authorized')
@facebook.authorized_handler
def facebook_authorized(resp):
    next_url = request.args.get('next') or url_for('new_paste')
    if resp is None:
        return redirect(next_url)
    session['fb_access_token'] = (resp['access_token'], '')
    me = facebook.get('/me')
    user = User.query.filter_by(fb_id=me.data['id']).first()
    if user is None:
        user = User()
        user.fb_id = me.data['id']
        db.session.add(user)
    user.display_name = me.data['name']
    db.session.commit()
    session['user_id'] = user.id
    return redirect(next_url)
```

Step 5 | Authentication

```
@app.route('/login/authorized')
@facebook.authorized_handler
def facebook_authorized(resp):
    next_url = request.args.get('next') or url_for('new_paste')
    if resp is None:
        return redirect(next_url)
    session['fb_access_token'] = (resp['access_token'], '')
    me = facebook.get('/me')
    user = User.query.filter_by(fb_id=me.data['id']).first()
    if user is None:
        user = User()
        user.fb_id = me.data['id']
        db.session.add(user)
    user.display_name = me.data['name']
    db.session.commit()
    session['user_id'] = user.id
    return redirect(next_url)
```

Step 5 | Authentication

```
@app.route('/login/authorized')
@facebook.authorized_handler
def facebook_authorized(resp):
    next_url = request.args.get('next') or url_for('new_paste')
    if resp is None:
        return redirect(next_url)
    session['fb_access_token'] = (resp['access_token'], '')
    me = facebook.get('/me')
    user = User.query.filter_by(fb_id=me.data['id']).first()
    if user is None:
        user = User()
        user.fb_id = me.data['id']
        db.session.add(user)
    user.display_name = me.data['name']
    db.session.commit()
    session['user_id'] = user.id
    return redirect(next_url)
```

Step 5 | Authentication

```
@app.route('/login/authorized')
@facebook.authorized_handler
def facebook_authorized(resp):
    next_url = request.args.get('next') or url_for('new_paste')
    if resp is None:
        return redirect(next_url)
    session['fb_access_token'] = (resp['access_token'], '')
    me = facebook.get('/me')
    user = User.query.filter_by(fb_id=me.data['id']).first()
    if user is None:
        user = User()
        user.fb_id = me.data['id']
        db.session.add(user)
    user.display_name = me.data['name']
    db.session.commit()
    session['user_id'] = user.id
    return redirect(next_url)
```

Step 5 | Authentication

```
@app.route('/login/authorized')
@facebook.authorized_handler
def facebook_authorized(resp):
    next_url = request.args.get('next') or url_for('new_paste')
    if resp is None:
        return redirect(next_url)
    session['fb_access_token'] = (resp['access_token'], '')
    me = facebook.get('/me')
    user = User.query.filter_by(fb_id=me.data['id']).first()
    if user is None:
        user = User()
        user.fb_id = me.data['id']
        db.session.add(user)
    user.display_name = me.data['name']
    db.session.commit()
    session['user_id'] = user.id
    return redirect(next_url)
```

Step 5 | Authentication

```
@app.route('/login/authorized')
@facebook.authorized_handler
def facebook_authorized(resp):
    next_url = request.args.get('next') or url_for('new_paste')
    if resp is None:
        return redirect(next_url)
    session['fb_access_token'] = (resp['access_token'], '')
    me = facebook.get('/me')
    user = User.query.filter_by(fb_id=me.data['id']).first()
    if user is None:
        user = User()
        user.fb_id = me.data['id']
        db.session.add(user)
    user.display_name = me.data['name']
    db.session.commit()
    session['user_id'] = user.id
    return redirect(next_url)
```

Step 5 | Authentication

```
@app.route('/login/authorized')
@facebook.authorized_handler
def facebook_authorized(resp):
    next_url = request.args.get('next') or url_for('new_paste')
    if resp is None:
        return redirect(next_url)
    session['fb_access_token'] = (resp['access_token'], '')
    me = facebook.get('/me')
    user = User.query.filter_by(fb_id=me.data['id']).first()
    if user is None:
        user = User()
        user.fb_id = me.data['id']
        db.session.add(user)
    user.display_name = me.data['name']
    db.session.commit()
    session['user_id'] = user.id
    return redirect(next_url)
```

Step 5 | Authentication

```
@app.route('/login/authorized')
@facebook.authorized_handler
def facebook_authorized(resp):
    next_url = request.args.get('next') or url_for('new_paste')
    if resp is None:
        return redirect(next_url)
    session['fb_access_token'] = (resp['access_token'], '')
    me = facebook.get('/me')
    user = User.query.filter_by(fb_id=me.data['id']).first()
    if user is None:
        user = User()
        user.fb_id = me.data['id']
        db.session.add(user)
    user.display_name = me.data['name']
    db.session.commit()
    session['user_id'] = user.id
    return redirect(next_url)
```

Step 5 | Authentication

```
@app.route('/login/authorized')
@facebook.authorized_handler
def facebook_authorized(resp):
    next_url = request.args.get('next') or url_for('new_paste')
    if resp is None:
        return redirect(next_url)
    session['fb_access_token'] = (resp['access_token'], '')
    me = facebook.get('/me')
    user = User.query.filter_by(fb_id=me.data['id']).first()
    if user is None:
        user = User()
        user.fb_id = me.data['id']
        db.session.add(user)
    user.display_name = me.data['name']
    db.session.commit()
    session['user_id'] = user.id
    return redirect(next_url)
```

Step 6 | View Functions

```
@app.route('/', methods=['GET', 'POST'])
def new_paste():
    if request.method == 'POST' and request.form['code']:
        paste = Paste(g.user, request.form['code'])
        db.session.add(paste)
        db.session.commit()
        return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('new_paste.html')


@app.route('/<int:paste_id>')
def show_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    return render_template('show_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/', methods=['GET', 'POST'])
def new_paste():
    if request.method == 'POST' and request.form['code']:
        paste = Paste(g.user, request.form['code'])
        db.session.add(paste)
        db.session.commit()
        return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('new_paste.html')


@app.route('/<int:paste_id>')
def show_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    return render_template('show_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/', methods=['GET', 'POST'])
def new_paste():
    if request.method == 'POST' and request.form['code']:
        paste = Paste(g.user, request.form['code'])
        db.session.add(paste)
        db.session.commit()
        return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('new_paste.html')


@app.route('/<int:paste_id>')
def show_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    return render_template('show_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/', methods=['GET', 'POST'])
def new_paste():
    if request.method == 'POST' and request.form['code']:
        paste = Paste(g.user, request.form['code'])
        db.session.add(paste)
        db.session.commit()
        return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('new_paste.html')


@app.route('/<int:paste_id>')
def show_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    return render_template('show_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/', methods=['GET', 'POST'])
def new_paste():
    if request.method == 'POST' and request.form['code']:
        paste = Paste(g.user, request.form['code'])
        db.session.add(paste)
        db.session.commit()
        return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('new_paste.html')


@app.route('/<int:paste_id>')
def show_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    return render_template('show_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/', methods=['GET', 'POST'])
def new_paste():
    if request.method == 'POST' and request.form['code']:
        paste = Paste(g.user, request.form['code'])
        db.session.add(paste)
        db.session.commit()
        return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('new_paste.html')
```

```
@app.route('/<int:paste_id>')
def show_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    return render_template('show_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/', methods=['GET', 'POST'])
def new_paste():
    if request.method == 'POST' and request.form['code']:
        paste = Paste(g.user, request.form['code'])
        db.session.add(paste)
        db.session.commit()
        return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('new_paste.html')
```

```
@app.route('/<int:paste_id>')
def show_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    return render_template('show_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/', methods=['GET', 'POST'])
def new_paste():
    if request.method == 'POST' and request.form['code']:
        paste = Paste(g.user, request.form['code'])
        db.session.add(paste)
        db.session.commit()
        return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('new_paste.html')


@app.route('/<int:paste_id>')
def show_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    return render_template('show_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/', methods=['GET', 'POST'])
def new_paste():
    if request.method == 'POST' and request.form['code']:
        paste = Paste(g.user, request.form['code'])
        db.session.add(paste)
        db.session.commit()
        return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('new_paste.html')


@app.route('/<int:paste_id>')
def show_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    return render_template('show_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/<int:paste_id>/delete', methods=['GET', 'POST'])
def delete_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    if g.user is None or g.user != paste.user:
        abort(401)
    if request.method == 'POST':
        if 'yes' in request.form:
            db.session.delete(paste)
            db.session.commit()
            return redirect(url_for('new_paste'))
        else:
            return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('delete_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/<int:paste_id>/delete', methods=['GET', 'POST'])
def delete_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    if g.user is None or g.user != paste.user:
        abort(401)
    if request.method == 'POST':
        if 'yes' in request.form:
            db.session.delete(paste)
            db.session.commit()
            return redirect(url_for('new_paste'))
        else:
            return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('delete_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/<int:paste_id>/delete', methods=['GET', 'POST'])
def delete_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    if g.user is None or g.user != paste.user:
        abort(401)
    if request.method == 'POST':
        if 'yes' in request.form:
            db.session.delete(paste)
            db.session.commit()
            return redirect(url_for('new_paste'))
        else:
            return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('delete_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/<int:paste_id>/delete', methods=['GET', 'POST'])
def delete_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    if g.user is None or g.user != paste.user:
        abort(401)
    if request.method == 'POST':
        if 'yes' in request.form:
            db.session.delete(paste)
            db.session.commit()
            return redirect(url_for('new_paste'))
        else:
            return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('delete_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/<int:paste_id>/delete', methods=['GET', 'POST'])
def delete_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    if g.user is None or g.user != paste.user:
        abort(401)
    if request.method == 'POST':
        if 'yes' in request.form:
            db.session.delete(paste)
            db.session.commit()
            return redirect(url_for('new_paste'))
        else:
            return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('delete_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/<int:paste_id>/delete', methods=['GET', 'POST'])
def delete_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    if g.user is None or g.user != paste.user:
        abort(401)
    if request.method == 'POST':
        if 'yes' in request.form:
            db.session.delete(paste)
            db.session.commit()
            return redirect(url_for('new_paste'))
        else:
            return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('delete_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/<int:paste_id>/delete', methods=['GET', 'POST'])
def delete_paste(paste_id):
    paste = Paste.query.get_or_404(paste_id)
    if g.user is None or g.user != paste.user:
        abort(401)
    if request.method == 'POST':
        if 'yes' in request.form:
            db.session.delete(paste)
            db.session.commit()
            return redirect(url_for('new_paste'))
        else:
            return redirect(url_for('show_paste', paste_id=paste.id))
    return render_template('delete_paste.html', paste=paste)
```

Step 6 | View Functions

```
@app.route('/my-pastes')
def my_pastes():
    if g.user is None:
        return redirect(url_for('login', next=request.url))
    pastes = Paste.query.filter_by(user=g.user).all()
    return render_template('my_pastes.html', pastes=pastes)
```

Step 6 | View Functions

```
@app.route('/my-pastes')
def my_pastes():
    if g.user is None:
        return redirect(url_for('login', next=request.url))
    pastes = Paste.query.filter_by(user=g.user).all()
    return render_template('my_pastes.html', pastes=pastes)
```

Step 6 | View Functions

```
@app.route('/my-pastes')
def my_pastes():
    if g.user is None:
        return redirect(url_for('login', next=request.url))
    pastes = Paste.query.filter_by(user=g.user).all()
    return render_template('my_pastes.html', pastes=pastes)
```

Step 6 | View Functions

```
@app.route('/my-pastes')
def my_pastes():
    if g.user is None:
        return redirect(url_for('login', next=request.url))
    pastes = Paste.query.filter_by(user=g.user).all()
    return render_template('my_pastes.html', pastes=pastes)
```

Step 7 | Templates

```
<!doctype html>
<title>{% block title %}{% endblock %} | Flask Pastebin</title>
<link rel=stylesheet type=text/css href="{{ url_for('static',
    filename='style.css') }}">
<div class=page>
  <h1>Flask Pastebin</h1>
  <ul class=nav>
    <li><a href="{{ url_for('new_paste') }}">New Paste</a>
    {% if g.user %}
    <li><a href="{{ url_for('my_pastes') }}">My Pastes</a>
    <li><a href="{{ url_for('logout') }}">Sign out ({{ g.user.display_name }})</a>
    {% else %}
    <li><a href="{{ url_for('login') }}">Sign in with Facebook</a>
    {% endif %}
  </ul>
  {% block body %}{% endblock %}
</div>
```

layout.html

Step 7 | Templates

```
<!doctype html>
<title>{% block title %}{% endblock %} | Flask Pastebin</title>
<link rel=stylesheet type=text/css href="{{ url_for('static',
    filename='style.css') }}">
<div class=page>
  <h1>Flask Pastebin</h1>
  <ul class=nav>
    <li><a href="{{ url_for('new_paste') }}">New Paste</a>
    {% if g.user %}
    <li><a href="{{ url_for('my_pastes') }}">My Pastes</a>
    <li><a href="{{ url_for('logout') }}">Sign out ({{ g.user.display_name }})</a>
    {% else %}
    <li><a href="{{ url_for('login') }}">Sign in with Facebook</a>
    {% endif %}
  </ul>
  {% block body %}{% endblock %}
</div>
```

layout.html

Step 7 | Templates

```
<!doctype html>
<title>{% block title %}{% endblock %} | Flask Pastebin</title>
<link rel=stylesheet type=text/css href="{{ url_for('static',
    filename='style.css') }}">
<div class=page>
  <h1>Flask Pastebin</h1>
  <ul class=nav>
    <li><a href="{{ url_for('new_paste') }}">New Paste</a>
    {% if g.user %}
    <li><a href="{{ url_for('my_pastes') }}">My Pastes</a>
    <li><a href="{{ url_for('logout') }}">Sign out ({{ g.user.display_name }})</a>
    {% else %}
    <li><a href="{{ url_for('login') }}">Sign in with Facebook</a>
    {% endif %}
  </ul>
  {% block body %}{% endblock %}
</div>
```

layout.html

Step 7 | Templates

```
<!doctype html>
<title>{% block title %}{% endblock %} | Flask Pastebin</title>
<link rel=stylesheet type=text/css href="{{ url_for('static',
    filename='style.css') }}">
<div class=page>
  <h1>Flask Pastebin</h1>
  <ul class=nav>
    <li><a href="{{ url_for('new_paste') }}">New Paste</a>
    {% if g.user %}
    <li><a href="{{ url_for('my_pastes') }}">My Pastes</a>
    <li><a href="{{ url_for('logout') }}">Sign out ({{ g.user.display_name }})</a>
    {% else %}
    <li><a href="{{ url_for('login') }}">Sign in with Facebook</a>
    {% endif %}
  </ul>
  {% block body %}{% endblock %}
</div>
```

layout.html

Step 7 | Templates

```
{% extends "layout.html" %}
{% block title %}New Paste{% endblock %}
{% block body %}
  <h2>New Paste</h2>
  <form action="" method=post>
    <div class=code><textarea name=code cols=60 rows=18></textarea></div>
    <p><input type=submit value="New Paste">
  </form>
{% endblock %}
```

new_paste.html

Step 7 | Templates

```
{% extends "layout.html" %}
{% block title %}New Paste{% endblock %}
{% block body %}
  <h2>New Paste</h2>
  <form action="" method=post>
    <div class=code><textarea name=code cols=60 rows=18></textarea></div>
    <p><input type=submit value="New Paste">
  </form>
{% endblock %}
```

new_paste.html

Step 7 | Templates

```
{% extends "layout.html" %}
{% block title %}New Paste{% endblock %}
{% block body %}
  <h2>New Paste</h2>
  <form action="" method=post>
    <div class=code><textarea name=code cols=60 rows=18></textarea></div>
    <p><input type=submit value="New Paste">
  </form>
{% endblock %}
```

new_paste.html

Step 7 | Templates

```
{% extends "layout.html" %}
{% block title %}Delete Paste #{{ paste.id }}{% endblock %}
{% block body %}
  <h2>Delete Paste #{{ paste.id }}</h2>
  <form action="" method=post>
    <p>Are you sure you want to delete the paste?  You cannot undo this.
    <p>
      <input type=submit name=yes value=Yes>
      <input type=submit name=no value=No>
  </form>
{% endblock %}
```

delete_paste.html

Step 7 | Templates

```
{% extends "layout.html" %}
{% block title %}My Pastes{% endblock %}
{% block body %}
  <h2>My Pastes</h2>
  <ul>
  {% for paste in pastes %}
    <li><a href="{{ url_for('show_paste', paste_id=paste.id)
         }}">#{{ paste.id }}</a>
      from {{ paste.pub_date.strftime('%Y-%m-%d @ %H:%M') }}
  {% endfor %}
  </ul>
{% endblock %}
```

my_pastes.html

Step 8 | CSS

```
body           { margin: 0; padding: 0; }
body, input    { font-size: 16px; font-family: 'Helvetica Neue', sans-serif; }
.page          { margin: 50px auto; width: 740px; }
h1             { margin: 0; font-weight: normal; color: #c00; }
a              { color: black; }
a:hover        { color: #c00; }
.nav           { margin: 0 0 20px 0; list-style: none; padding: 0; }
.nav li        { display: inline; }
.nav li + li:before { content: " // "; }
h2             { font-weight: normal; margin: 0; }
dl             { overflow: auto; font-size: 14px; }
dl dt          { font-weight: bold; width: 90px; float: left;
                 clear: left; }
dl dd          { float: left; margin: 0; padding: 0; }
pre, textarea { font-family: 'Consolas', monospace; font-size: 14px;
                background: #eee; padding: 0; margin: 0; }
textarea       { border: none; width: 720px; }
.code, .flash { background: #eee; margin: 10px -30px; padding: 10px 30px; }
```

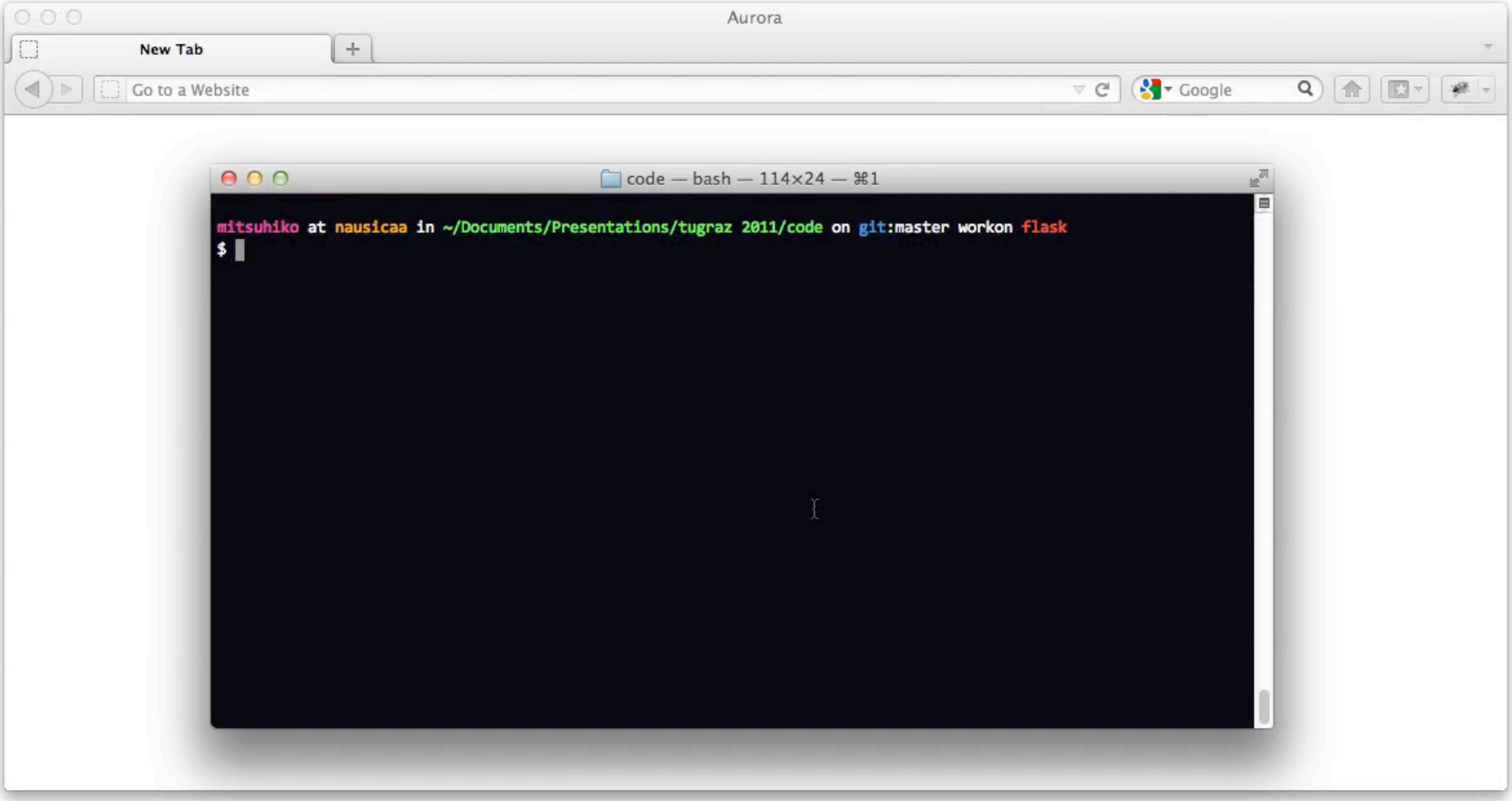
Step 9 | Flashing

```
{% for message in get_flashed_messages() %}
  <p class=flash>{{ message }}
{% endfor %}
```

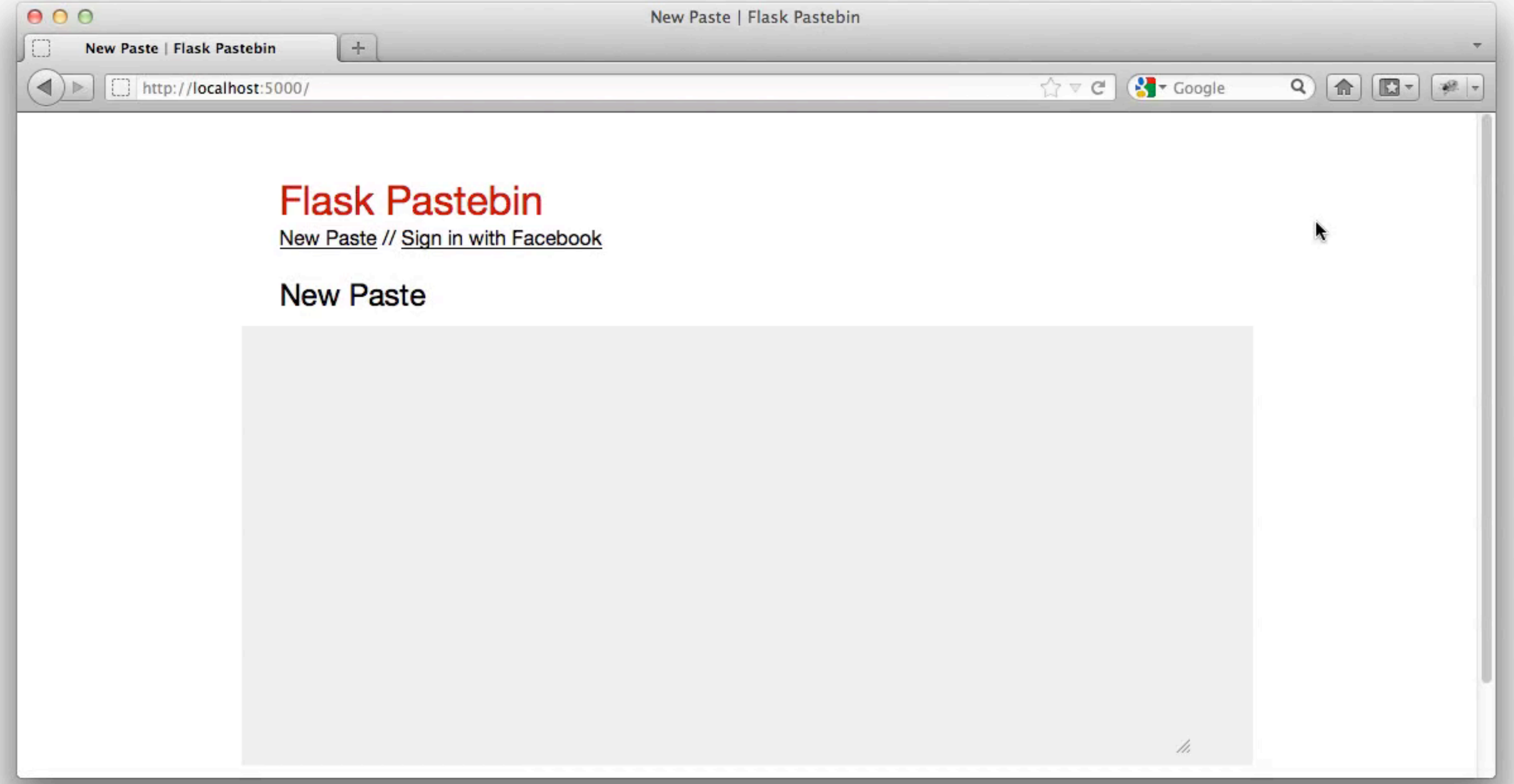
Step 9 | Flashing

```
from flask import flash

@app.route('/logout')
def logout():
    session.clear()
    flash('You were logged out')
    return redirect(url_for('new_paste'))
```

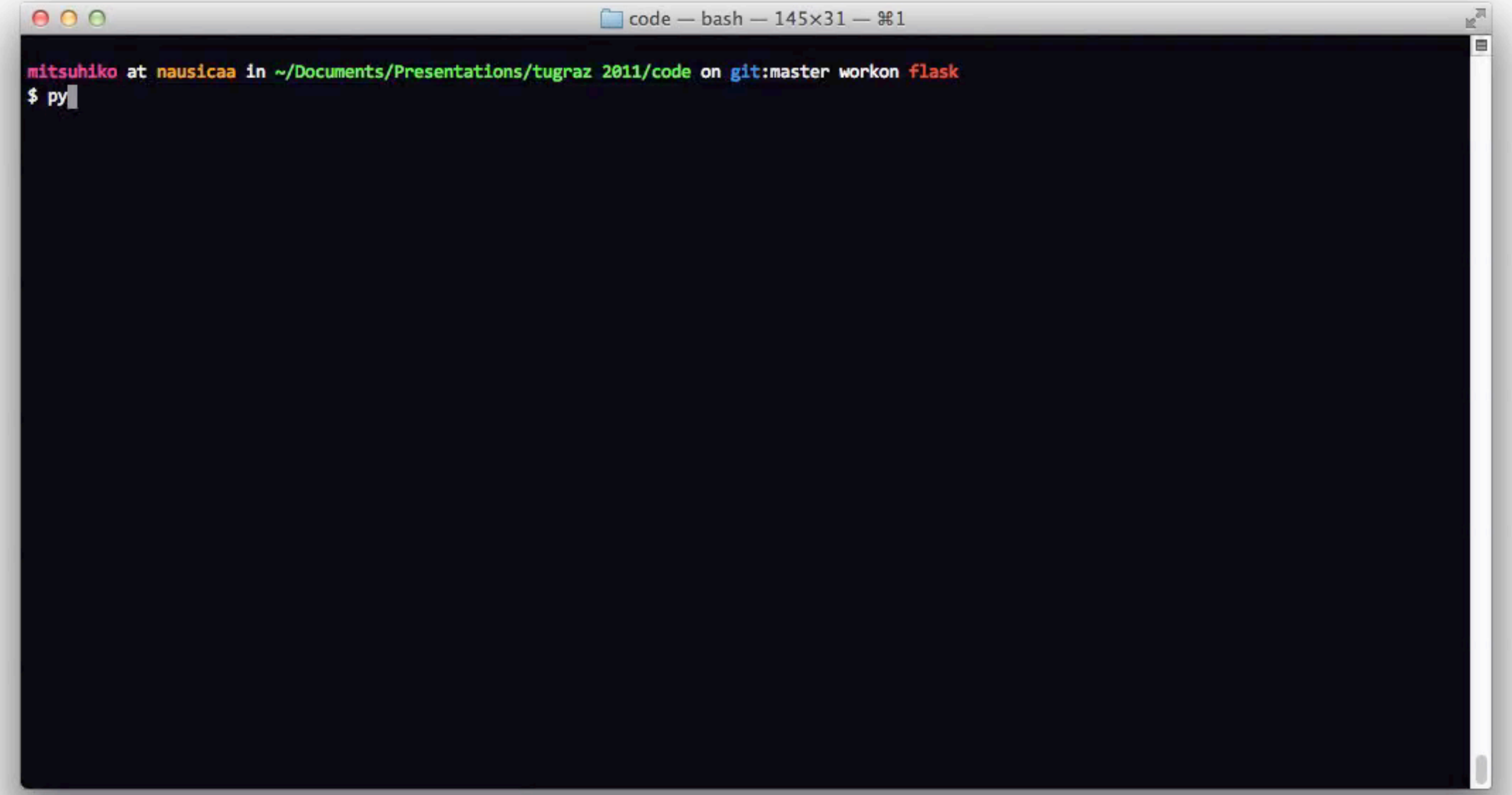
Demo

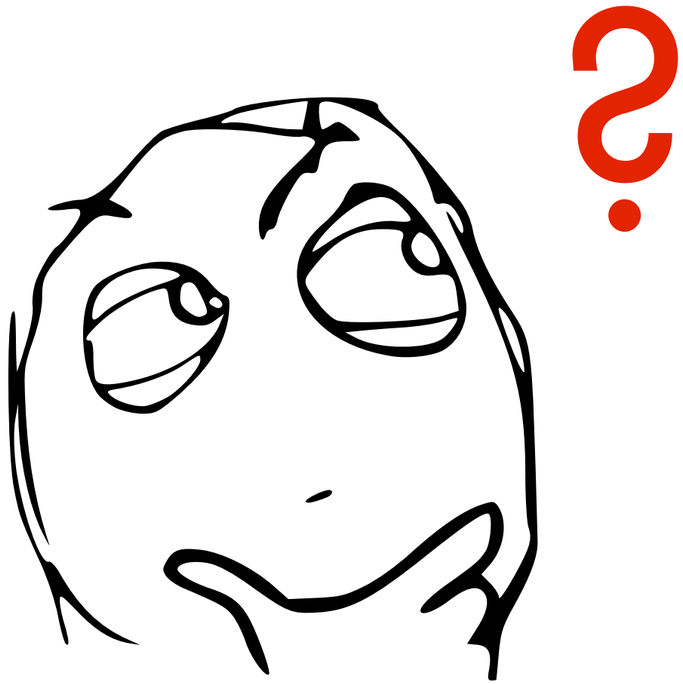

Aurora
New Tab
Go to a Website
Google
code — bash — 114×24 — ⌘1
mitsuhiko at nausicaa in ~/Documents/Presentations/tugraz 2011/code on git:master workon flask
$

Debugging


New Paste | Flask Pastebin
New Paste | Flask Pastebin
http://localhost:5000/
Google
Flask Pastebin
New Paste // Sign in with Facebook
New Paste

Interactive Shell


code — bash — 145×31 — ⌘1
mitsuhiko at nausicaa in ~/Documents/Presentations/tugraz 2011/code on git:master workon flask
$ py



These slides: http://lucumr.pocoo.org/talks/