PYTHON & FLASK

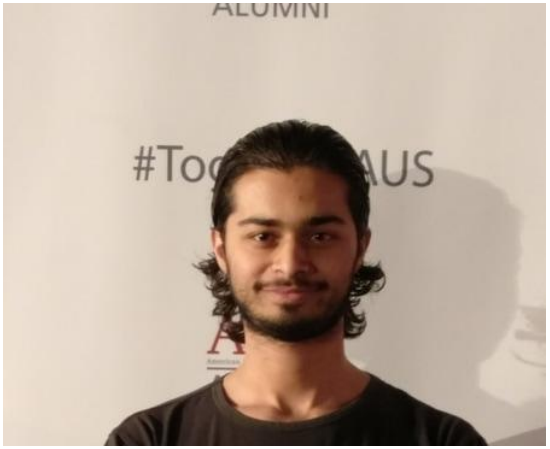
CG INTERACTIVE
Jan 31 - Feb 7, 2017

WHO ARE WE?

FAYEZ BARAKJI

<u>CEO</u>

DESIGNER, PROJECT MANAGER, & DEVELOPER



ABDULWAHAB SAHYOUN

<u>CTO</u>

LEAD PROGRAMMER & DEVELOPER

WHAT WE DO

- MOBILE APPS (JAVA, C#, JS)
- VR APPS (C# UNITY, WEBVR)
- VIDEO GAMES (C# UNITY)
- IT SOLUTIONS

Our games

WHAT IS PYTHON?

H@CKER5??


PYTHON

While loops

```
count = 0
while (count < 9):
  print 'The count is:', count
  count = count + 1
print "Good bye!“
# -------------------------------------------------------
var = 1
while var == 1 :  # This constructs an infinite loop
  num = raw_input("Enter a number  :")
  print "You entered: ", num
print "Good bye!"
```

While loops

```
count = 0
while count < 5:
  print count, " is  less than 5"
  count = count + 1
else:
  print count, " is not less than 5“
# ------------------------------------------------------------
flag = 1
while (flag): print 'Given flag is really true!'
print "Good bye!"
```

For loops

```
for x in range(0, 3):
   print "We're on time %d" % (x)
 #-----------------------------------
for x in xrange(3):
   print x
else:
   print 'Final x = %d' % (x)
 #-----------------------------------
```

For loops (nested)

```
list_of_lists = [ [1, 2, 3], [4, 5, 6], [7, 8, 9]]
for list in list_of_lists:
  for x in list:
    print x

# ---------------------------------------

numbers = range(1,6)
      for count in numbers:
        print (count)
```

Appending arrays (lists) vs. extending arrays

```
x = [1, 2, 3]
x.extend([4, 5])
print (x)
```

gives you: [1, 2, 3, 4, 5]

```
x = [1, 2, 3]
x.append([4, 5])
print (x)
```

gives you: [1, 2, 3, [4, 5]]

Date & Times

```
>>> from datetime import datetime
>>> now = datetime.now()
>>> print now
2012-12-31 15:54:42.915204
>>> now = datetime.utcnow()
>>> print now
2012-12-31 23:55:13.635874
```

Math - Circle

```
from math import pi
class Circle:

    def __init__(self,radius):
            self.r = radius

    def area(self):
        return pi * self.r * 2

    def circumference(self):
        return 2 * pi * self.r
```

Apps & Sites built with Flask

- https://skylines.aero/flights/pilot/2913
- https://raw.githubusercontent.com/Jahaja/psdash/master/docs/screenshots/overview.png
- https://github.com/Jahaja/psdash
- https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world
- https://moinmo.in/
- https://github.com/tsah20/bean-counter
- http://flask.pocoo.org/community/poweredby/

WHERE IS PYTHON USED?

- Google
- Facebook
- Dropbox
- Yahoo
- NASA
- IBM
- Mozilla
- Quora

PYTHON IS SLOW

Unlike Java and C, Python is much slower.

WHY PYTHON 2.7?

Python 2.7 is the last iteration of Python 2, Python 3 is the new iteration, but lacks the community support of python 2.7, for now.

THINGS YOU SHOULD’VE DONE

- **Installed Pycharm**
- **Made sure it works properly**

CREATING A VARIABLE

X = 1000

X = banana

X= 3.145674245

PRINTING

Print x

Print “hi my favorite fruit is %s” %x

Print “hi my favorite number is %d” %x

LOOPS

```
x=5
  while x < 10:
    x=x+1
    print x
Print ‘loop is done’
```

LOOPS CONTINUED

```
for letter in 'Python':
    print 'Current Letter :', letter
fruits = ['banana', 'apple',  'mango']
    for fruit in fruits:
        print 'Current fruit :', fruit
print "Good bye!"
```

ARRAYS

List = [50,25,32,21]

Secondlist=[‘apple’,’banana’, ‘cucumber’]

Thirdlist= [‘a’,’b’,’c’]

List[0]= 88

Print secondlist[1]

Print list[:2]

Print list[0:2]

Print list[2:]

IMPORTING

from ctypes import *

From Flask import flask

OTHER COOL ****

```
from ctypes import *
windll.User32.MessageBoxA(0,body,title,4)
windll.WINMM.mciSendStringW(u"set cdaudio door open", None, 0,
    None)
```

-File reading and writing

-Complex mathematics and statistics

-Web pages, servers, and anything in between.

-accessing another computer on the same network

-Completely destroying computers

THE POSSIBILITIES ARE ENDLESS!

FUNCTIONS

```
def cookie():
        print "I love cookies"
        return 45
```

```
def printme( str ):
        for I in str
         print str
        return
```

CLASSES

```
class Fruit(object):
    """A class that makes various tasty fruits."""
    def __init__(self, name, color, flavor, poisonous):
        self.name = name
        self.color = color
        self.flavor = flavor
        self.poisonous = poisonous
```

```
    def description(self):
        print "I'm a %s %s and I taste %s." % (self.color,
     self.name, self.flavor)
    def is_edible(self):
        if not self.poisonous:
            print "Yep! I'm edible."
        else:
            print "Don't eat me! I am super poisonous."

lemon = Fruit("lemon", "yellow", "sour", False)
lemon.description()
lemon.is_edible()
```

FLASK

DOWNLOADING FLASK AND IMPORTING

SYNTAX

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
   def page1():
       return "welcome to the show! "
   if __name__ == "__main__":
       app.run(debug=True)
```

SECOND PAGE

```
@app.route('/page2')
   def page2():
       return "welcome to page 2! "
```

ADDING OUTSIDE CODE

Create static and templates files.

```
<title>test 1</title>
   <h1> Fayez Rules</h1>
```

REROUTING PAGES

```
<button type="button"  onclick="window.location='/page2';"
   >page2</button>
```

SENDING DATA TO OUTSIDE PAGES

```
from flask import Flask, render_template

@app.route("/user/<name>")
   def user(name):
       return render_template("profile.html", name=name)
```

```
<h2>Hey there {{ name }}</h2>
```

GETTING DATA FROM OUTSIDE PAGES

```
<form method="post">
<input id="username" name="username" type="text"/>
<input id="email" name="email" type="text"/>
<input type="submit"/>
</form>

@app.route('/gettingdata', methods=['POST'])
def a():
    y = request.form['username']
    print y
    return 0
```

Embedding HTML

```
@app.route('/calculator', methods=['GET', 'POST'])
def index():
    if request.method == 'GET':
        # show html form
        return '''
            <form method="post">
                <input type="text" name="expression" />
                <input type="submit" value="Calculate" />
            </form>
        '''
    elif request.method == 'POST':
        # calculate result
        expression = request.form.get('expression')
        result = eval(expression)
        return 'result: %s' % result
```

Courtesy of Pythonbeginner.org

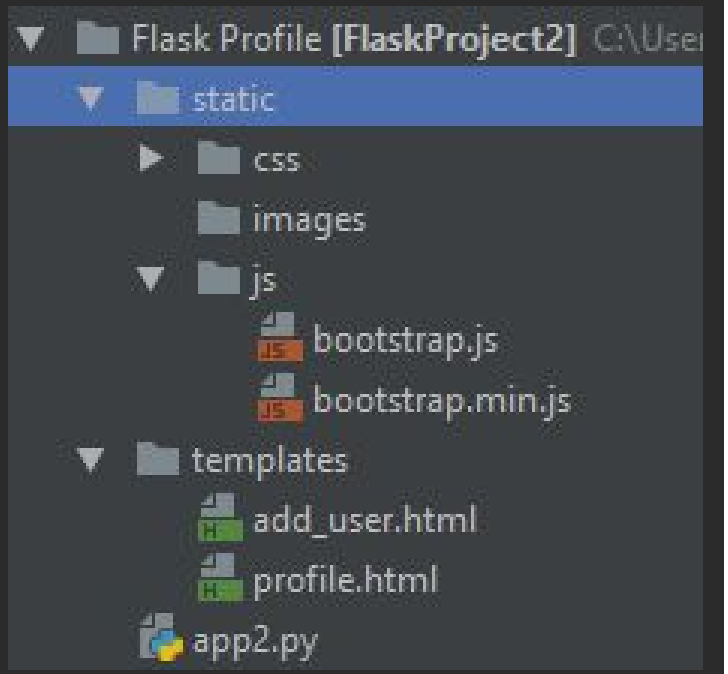
CSS INTEGRATION

```
<link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='style.css')}}">
```

```
h1{
color: deepskyblue;}
```

User profile https://goo.gl/hJTzBG

```python
from flask import Flask
from flask import request
from flask import render_template

app = Flask(__name__)


class User:
    def __init__(self, email, password):
        self.email = email
        self.password = password
```



User profile

```
@app.route('/')
def index():
    return render_template('add_user.html')


@app.route('/login_user', methods=['POST'])  # login simulation
example only ( does not check DB)
def login_user():
    user = User(request.form['email'],
request.form['password'])
    print user.email
    print user.password

    return render_template('profile.html', user=user)

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

add_user.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>User Reg</title>
</head>
<body>
 <!-- Bootstrap core CSS -->
    <link href="static/css/bootstrap.min.css" rel="stylesheet">
    <!-- Custom styles for this template -->
    <link href="static/css/signin.css" rel="stylesheet">
 <div class="container">
      <form method="post" action="/login_user" class="form-signin" >
        <h2 class="form-signin-heading">Please Register</h2>
        <label for="email" class="sr-only">Email address</label>
        <input name="email" type="email" id="email" class="form-control" placeholder="Email
address" required autofocus>
        <label for="password" class="sr-only">Password</label>
        <input name="password" type="password" id="password" class="form-control"
placeholder="Password" required>
        <div class="checkbox">
          <label>
            <input type="checkbox" value="remember-me"> Remember me
          </label>
        </div>
        <button class="btn btn-lg btn-primary btn-block" type="submit">Register</button>
      </form>
    <script src="static/js/ie10-viewport-bug-workaround.js"></script>
</body>
</html>
```

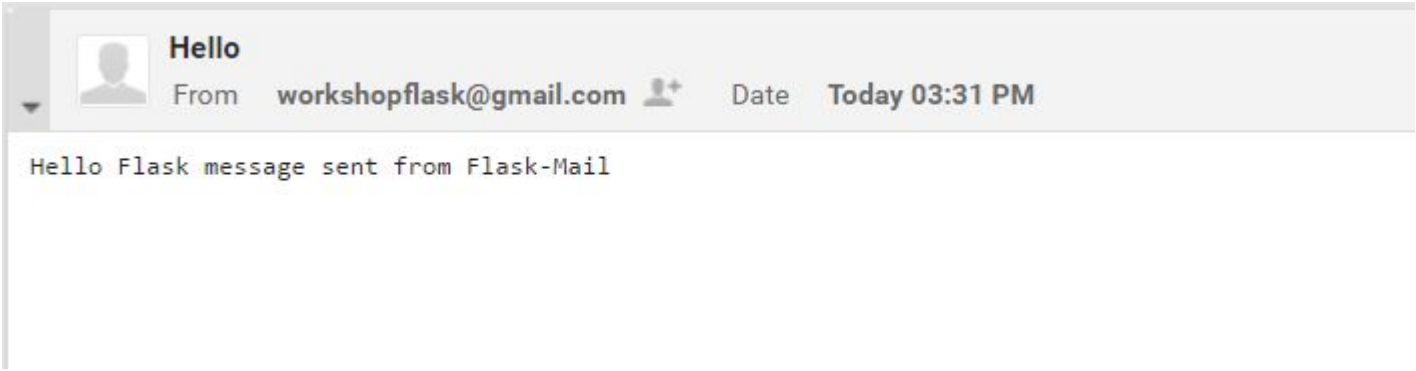
profile.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>User Profile</title>
</head>
<body>
<table>
      <tr valign="top">
          <td><img
src="http://i-cdn.phonearena.com/images/articles/97667-image/retina-display.png"
alt="profile avatar" style="width:304px;height:228px;"> </td>
          <td><h1>User: {{ user.email }}</h1></td>
      </tr>
</table>
</body>
</html>
```

Flask Mail

```
from flask_mail import Message
from flask_mail import Mail
from flask import Flask

app = Flask(__name__)

app.config['MAIL_SERVER'] = 'smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'workshopflask@gmail.com'
app.config['MAIL_PASSWORD'] = 'flaskworkshop34'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True

mail = Mail(app)

@app.route("/page8")
def index():
    msg = Message('Hello', sender='workshopflask@gmail.com',
recipients=['abdulwahab@cg-interactive.com'])
    msg.body = "Hello Flask message sent from Flask-Mail"
    mail.send(msg)
    return "Sent"

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

Result



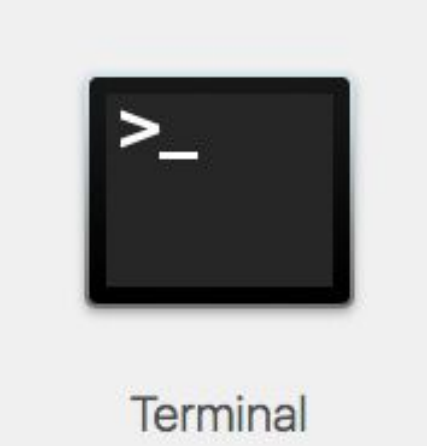
courtesy of tutorialspoint.com


PostgreSQL

MAC OSX Fix for psycopg2 (Homebrew)

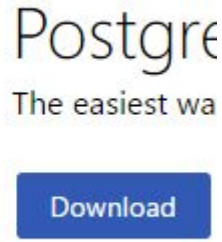
1. /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

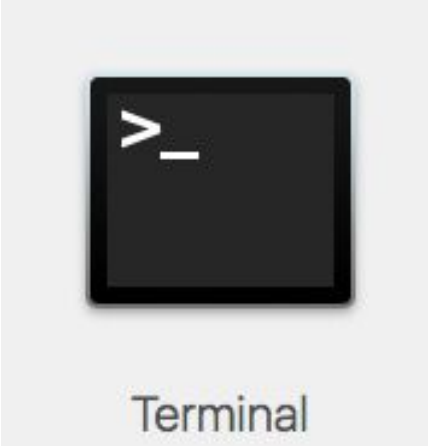
2. brew install python



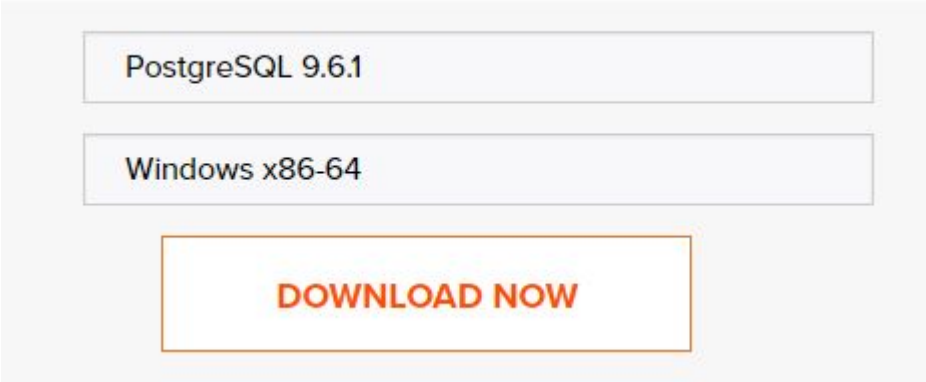
MAC OSX Fix 2 for psycopg2 (PIP and EASY_INSTALL)

1. Download: http://postgresapp.com
2. PATH=$PATH:/Applications/Postgres.app/Contents/Versions/9.6/bin
3. easy_install psycopg2

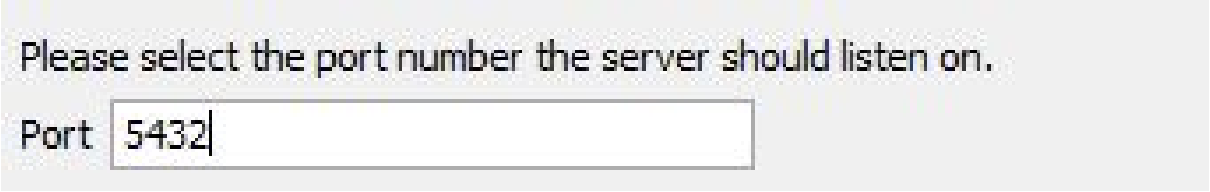




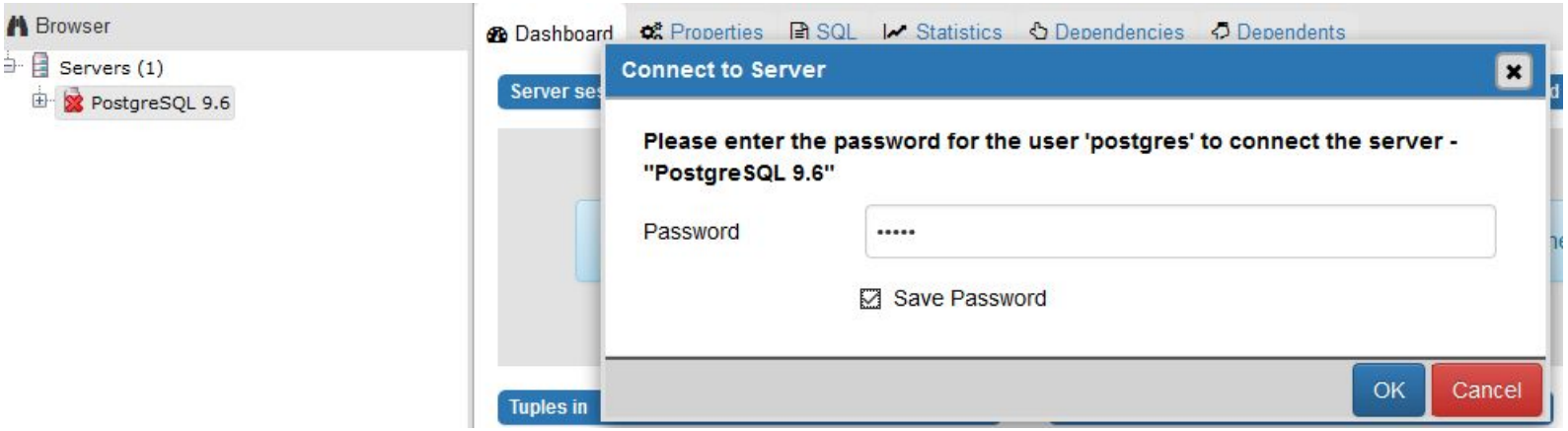
VERSION



PORT

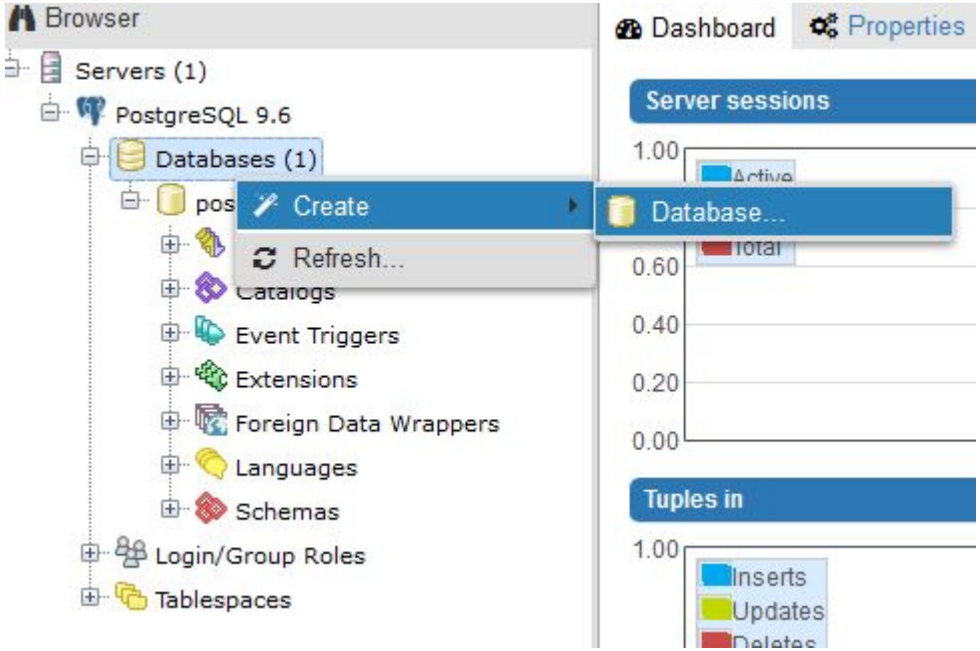


SETUP



Password you already entered during installation

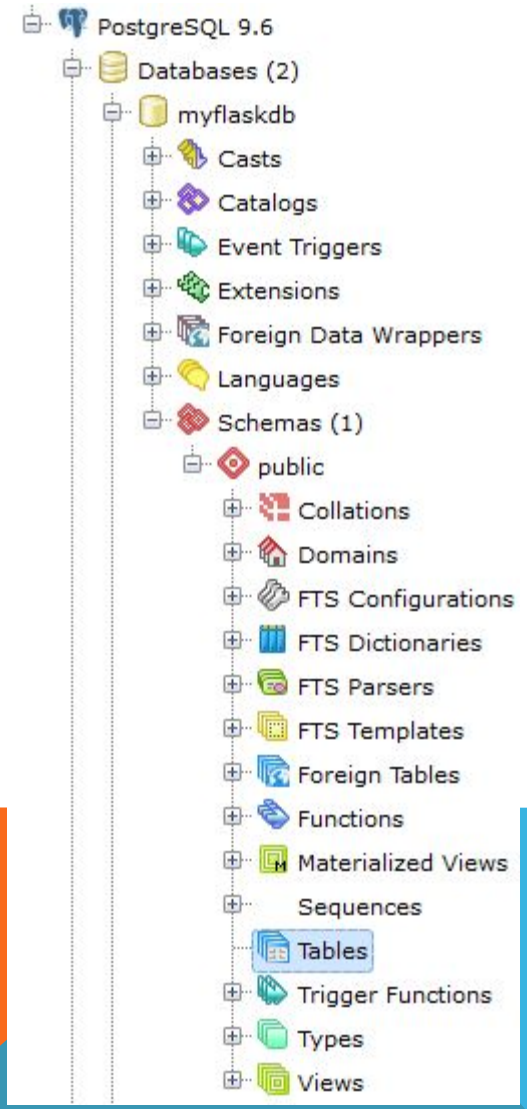
SETUP (CONT.)

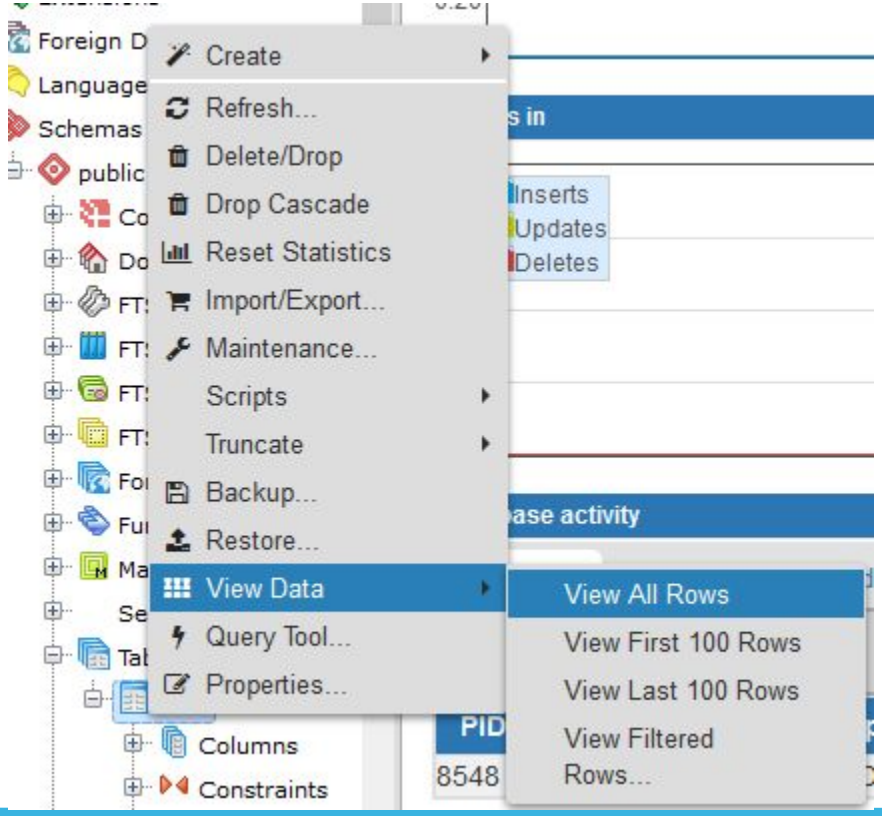


Create a new database and give it a custom name

Use the default user ‘postgres’ with all privileges

WHERE TO FIND TABLE DATA ?





SQLAlchemy

NEW IMPORTS

```
from flask import redirect, url_for
from flask_sqlalchemy import SQLAlchemy
```

INSTALL THESE PACKAGES

- Flask-SQLAlchemy
- Flask-Psycopg2

SQL ALCHEMY – DB & USER SETUP

```
app = Flask(__name__)# previous code
db = SQLAlchemy(app)

class User(db.Model): #DB table / User class
id = db.Column(db.Integer, primary_key = True)
username = db.Column(db.String(80), unique = True)
email = db.Column(db.String(120), unique = True)

def __init__(self, username, email): #sqlalchemy initialization
method
self.username = username
self.email = email

def __repr__(self):
  return '<User %r>' % self.username
```

SQL ALCHEMY – SETUP (CONT.)

```
app.config['SQLALCHEMY_DATABASE_URI'] =
'postgresql+psycopg2://postgres:admin@localhost/myflaskdb'
# postgresql://user:password@localhost
app.debug = True
db.create_all()
db.session.commit()
```

PSYCOPG2 IS A POSTGRESQL ADAPTER FOR PYTHON

Adding a User to Database

```
@app.route('/page9')
def page1():
    return render_template("add_user.html")


@app.route('/post_user', methods=['POST'])
def post_user():
    user = User(request.form['username'],
request.form['email'])
    db.session.add(user)
    db.session.commit()
    return redirect(url_for('user', name=user.username))
```

ADD_USER.HTML

```
<!DOCTYPE html> <html lang="en">
<head>
<meta charset="UTF-8">
<title>User Reg</title> </head>
<body>
<form method="post" action="/post_user"> <label>Username:</label>
<input id="username" name= "username" type="text" />
<label>Email:</label>
<input id="email" name= "email" type="text" />

<input type="submit" />
</form>

</body> </html>
```

PROFILE PAGE (AFTER REG.)

```
@app.route("/user/<name>")
def user(name):
    user = User.query.filter_by(username=name).first()
    return render_template("profile.html", user=user)
```

FLASK AUTHENTICATION

Why ?

1. Pre-built routes for registration and login
2. Automatic security built in to your site server
3. Easy to use and setup for starters

What to do beforehand?

1. Delete existing tables from postrgeSQL
2. Remove existing DB models from python app

FLASK AUTHENTICATION (CONT.)

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] =
'postgresql+psycopg2://postgres:admin@localhost/mydb'
app.config['SECRET_KEY'] = 'super-secret'
app.config['SECURITY_REGISTERABLE'] = True
app.debug = True
db = SQLAlchemy(app)

# Define models
roles_users = db.Table('roles_users',
                       db.Column('user_id', db.Integer(),
db.ForeignKey('user.id')),
                       db.Column('role_id', db.Integer(),
db.ForeignKey('role.id')))
```

FLASK AUTHENTICATION (CONT.)

```
class Role(db.Model, RoleMixin):
    id = db.Column(db.Integer(), primary_key=True)
    name = db.Column(db.String(80), unique=True)
    description = db.Column(db.String(255))


class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(255), unique=True)
    password = db.Column(db.String(255))
    active = db.Column(db.Boolean())
    confirmed_at = db.Column(db.DateTime())
    roles = db.relationship('Role', secondary=roles_users,
backref=db.backref('users', lazy='dynamic'))
```

FLASK AUTHENTICATION (CONT.)

```
# Setup Flask-Security
user_datastore = SQLAlchemyUserDatastore(db, User, Role)
security = Security(app, user_datastore)

# Create a user to test with
@app.before_first_request
def create_user():
    db.create_all()
    db.session.commit()

@app.route('/')
@login_required
def home():
    return "Index"
```

Run on first launch

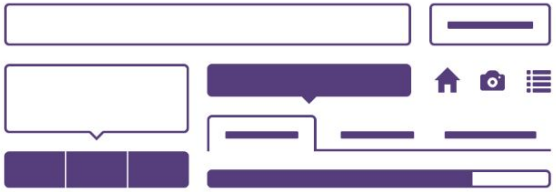
Default route to site will now require users to register and login

REQUIRE SIGN IN ON PROFILE PAGE (EMAIL)

```
@app.route('/profile/<email>')
@login_required
def profile(email):
    user = User.query.filter_by(email=email).first()
    return render_template('profile.html', user=user)
```

Email sign in this time !

Bootstrap is the most popular HTML, CSS, and JS framework in the world for building responsive, mobile-first projects on the web.

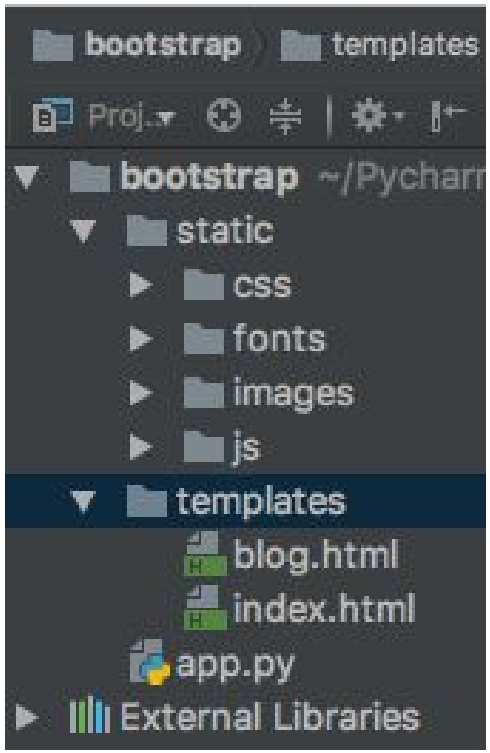


WHY BOOTSTRAP ??

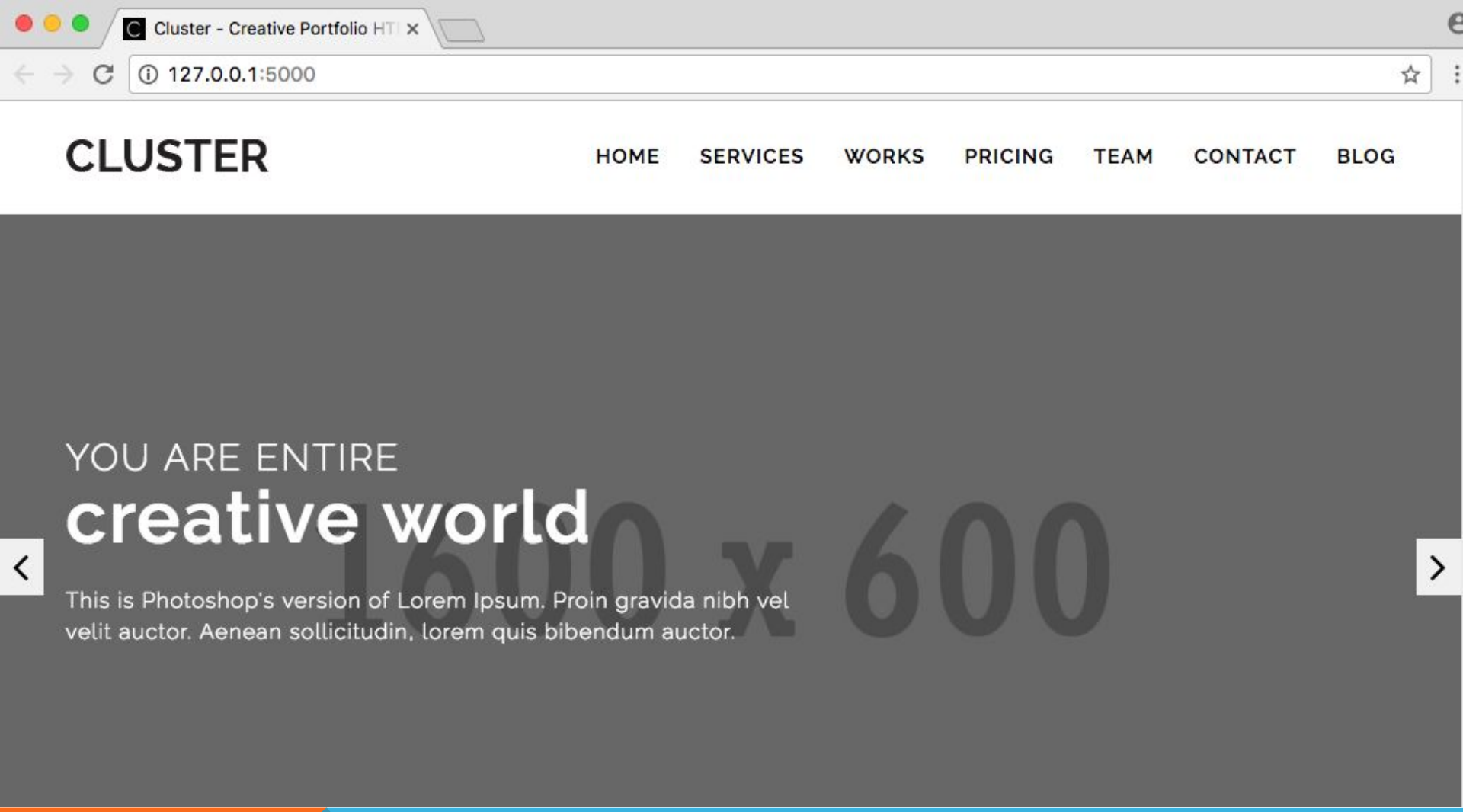
- **Bootstrap** is responsive, mobile-first, prevailing, and front-end framework, which is developed along with CSS, JavaScript, and HTML
- Huge number of resources accessible for **Bootstrap**
- **VERY EASY TO SETUP**

https://www.devsaran.com/blog/10-best-reasons-use-bootstrap-amazing-web-designs

ADD TEMPLATES AND JS/CSS FILES



https://goo.gl/eQf9xV


Cluster - Creative Portfolio HT
127.0.0.1:5000
CLUSTER
HOME
SERVICES
WORKS
PRICING
TEAM
CONTACT
BLOG
YOU ARE ENTIRE
creative world
1600 x 600
This is Photoshop's version of Lorem Ipsum. Proin gravida nibh vel velit auctor. Aenean sollicitudin, lorem quis bibendum auctor.

BOOTSTRAP CSS CORE

```
<!-- Bootstrap core CSS -->
   <link href="static/css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->
   <link href="static/css/signin.css" rel="stylesheet">
 </head>
```

BOOTSTRAP JS CORE

```
<!-- Bootstrap core JavaScript
  ================================================== -->
  <!-- Placed at the end of the document so the pages load faster -->
  <script src="https://code.jquery.com/jquery-3.1.1.slim.min.js"
integrity="sha384-A7FZj7v+d/sdmMqp/nOQwliLvUsJfDHW+k9Omg/a/EheAdgtzNs3hpfag6Ed950n"
crossorigin="anonymous"></script>
  <script>window.jQuery || document.write('<script src="../../assets/js/vendor/jquery.min.js"><\/script>')</script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.4.0/js/tether.min.js"
integrity="sha384-DztdAPBWPRXSA/3eYEEUWrWCy7G5KFbe8fFjk5JAIxUYHKkDx6Qin1DkWx51bBrb"
crossorigin="anonymous"></script>
  <script src="static/js/bootstrap.min.js"></script>
  <!-- Just to make our placeholder images work. Don't actually copy the next line! -->
  <script src="static/js/vendor/holder.min.js"></script>
  <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->
  <script src="static/js/ie10-viewport-bug-workaround.js"></script>
```

127.0.0.1:5000/signin

Please sign in

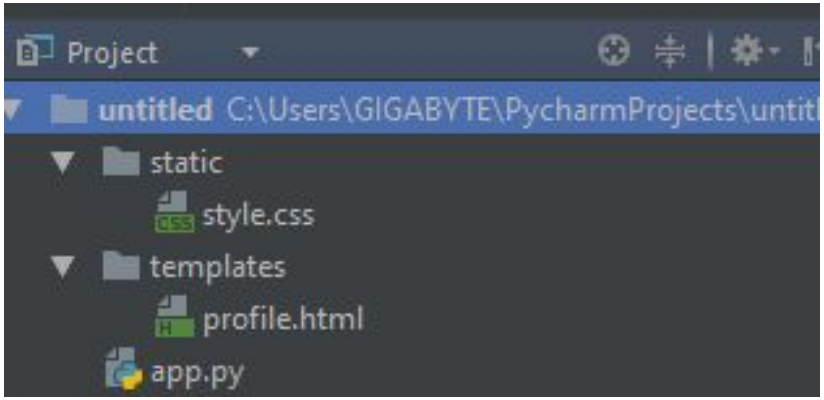
abdulwahab@cg-interactive.com

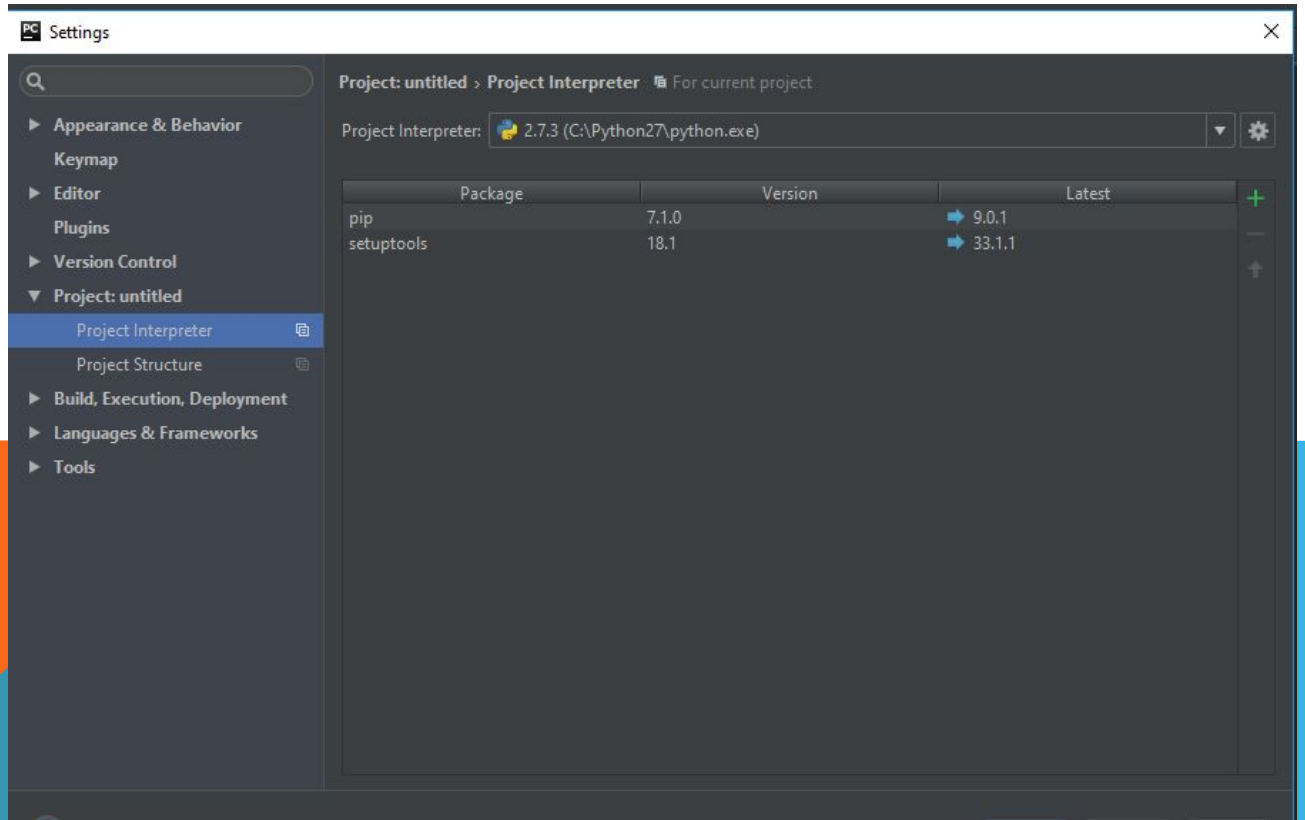
•••••••

Remember me

Sign in

EXTRA SLIDES – PROJECT STRUCTURE





EXTRA SLIDES – ADDING PACKAGES

Available Packages

Flask

Flask
Flask-ACL
Flask-API
Flask-API-Utils
Flask-API.yandex
Flask-APIBlueprint
Flask-APIForm
Flask-APScheduler
Flask-APScheduler-fork
Flask-ASGI
Flask-AYAH
Flask-Access
Flask-Actions
Flask-ActiveRecord
Flask-Admin
Flask-Admin-Utils
Flask-Administration
Flask-Aggregator
Flask-Airbrake
Flask-Alchemy
Flask-AlchemyDumps
Flask-AlchemyView
Flask-Alchy
Flask-Alcool
Flask-Alembic

Description

A microframework based on Werkzeug, Jinja2 and good intentions

Version

0.12

Author

Armin Ronacher

mailto:armin.ronacher@active-4.com
http://github.com/pallets/flask/

Specify version 0.12

Options

Install to user's site packages directory (C:\Users\GIGABYTE\AppData\Roaming\Python)

Install Package Manage Repositories

EXTRA SLIDES – ADDING IMAGES AND VIDEO

```
<img class="first-slide" src="/static/images/buffer3.gif" alt="First slide">

<div class="embed-responsive embed-responsive-16by9">
  <iframe class="embed-responsive-item"
src="https://www.youtube.com/embed/3ok2lTj4Ui4?autoplay=1"></iframe>
</div>
```