Python Web Programming with Flask

Pertemuan 10

MK Algoritma Pemrograman II

Ika Qutsiati Utami, S.Kom., M.Sc.

Program Studi S1 Teknologi Sains Data

Fakultas Teknologi Maju dan Multidisiplin

Universitas Airlangga Indonesia

Outline

- Introduction to Flask
- Why Flask? Advantages
- Quickstart
- Creating Simple Blog

Introduction to Flask

- Flask is a web framework, written in Python
- Using Flask, a python application can run on webserver (specifically WSGI) and serve web content (comparable to PHP on apache/nginx)
- Flask can be used as a base for any web application with user interface, or even headless webapp/webservice
- The Web Server Gateway Interface (WSGI, pronounced whiskey or WIZ-ghee) is a simple calling convention for web servers to forward requests to web applications or frameworks written in the Python programming language. The current version of WSGI, version 1.0.1, is specified in Python Enhancement Proposal (PEP) 3333.

Flask is Built on..

Werkzeug

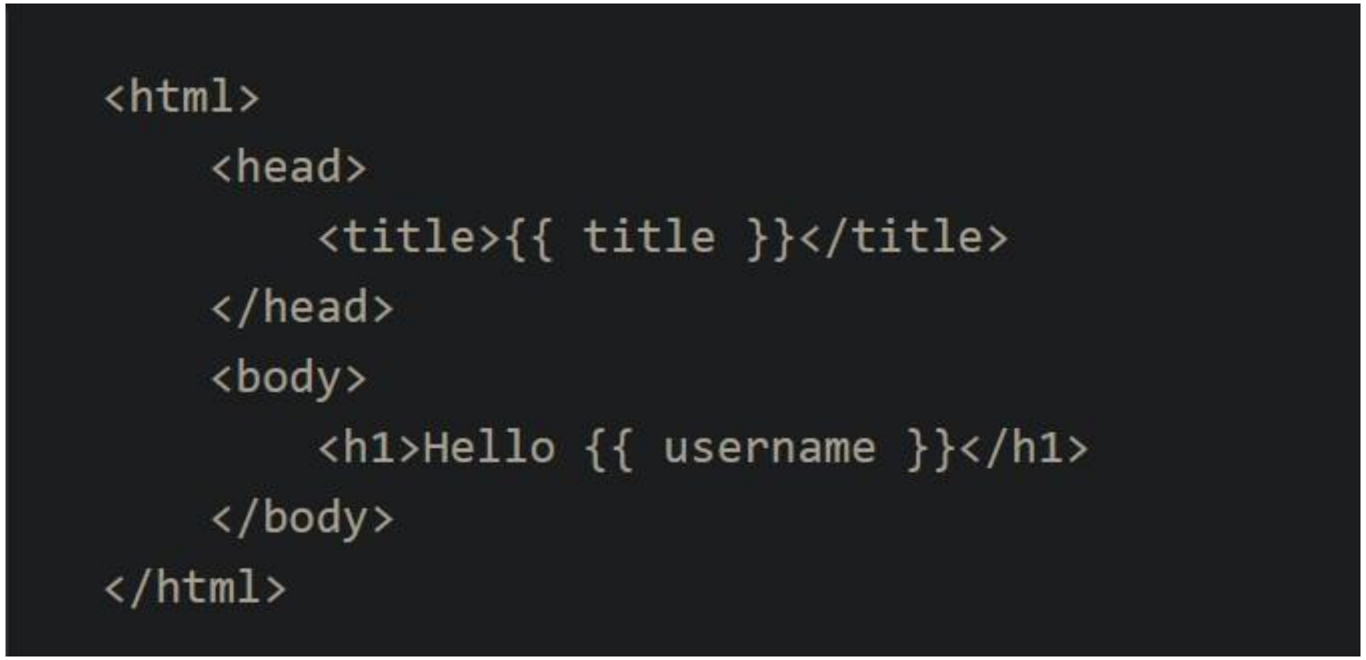
- Werkzeug is a WSGI toolkit that implements requests, response objects, and utility functions. This enables a web frame to be built on it. The Flask framework uses Werkzeug as one of its bases.

jinja2

- jinja2 is a popular template engine for Python. A web template system combines a template with a specific data source to render a dynamic web page.

Using Jinja2

- With jinja2, you can insert dynamic content on html.
- Use python as the controller (as in MVC structure) to decide which content to serve



```
<html>
    <head>
        <title>{{ title }}</title>
    </head>
    <body>
        <h1>Hello {{ username }}</h1>
    </body>
</html>
```

Flask Advantages

EASY TO READ, EASY TO CODE

HIGH-LEVEL WEB PROGRAMMING FRAMEWORK

VERSATILE

RESTFUL

EASY TO ROUTE URLS

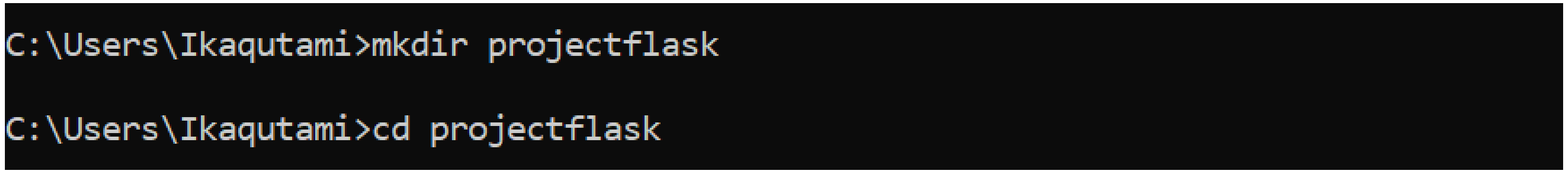
EASY TO MAINTAIN (MODULAR, MVC PARADIGM BY DEFAULT)

SCALABLE

MOST IMPORTANTLY, IT'S IN PYTHON!

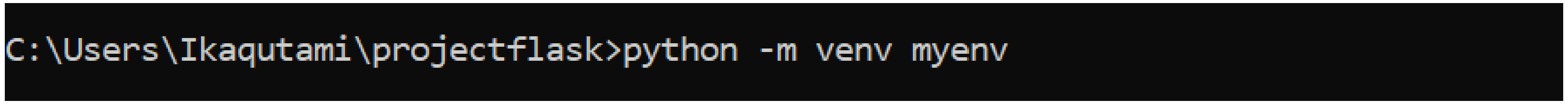
Let’s get started

- Create folder of your app



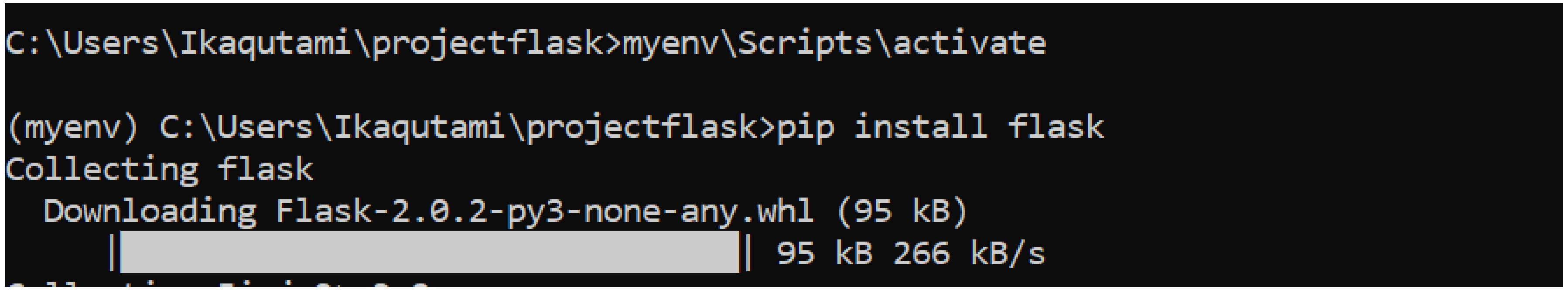
```
C:\Users\Ikaqutami>mkdir projectflask

C:\Users\Ikaqutami>cd projectflask
```

- Create a virtual environment for flask app development (to prevent ‘dependency’)



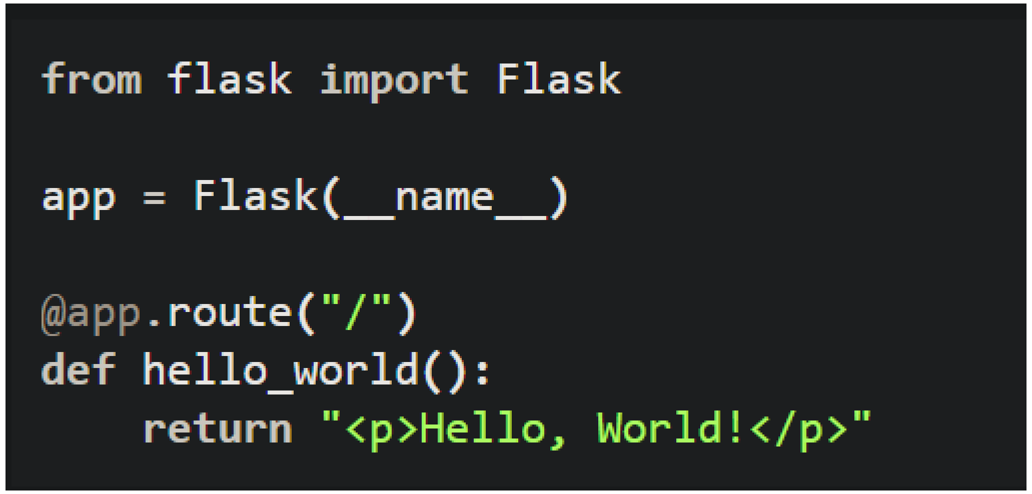
```
C:\Users\Ikaqutami\projectflask>python -m venv myenv
```

- To activate, navigate to the virtualenv path and invoke

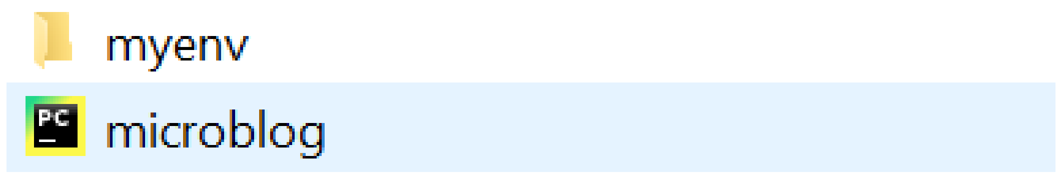


```
C:\Users\Ikaqutami\projectflask>myenv\Scripts\activate

(myenv) C:\Users\Ikaqutami\projectflask>pip install flask
Collecting flask
  Downloading Flask-2.0.2-py3-none-any.whl (95 kB)
     |████████████████████████████████| 95 kB 266 kB/s
```

- Try a hello world! Save it as <microblog.py> / or whatever



```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

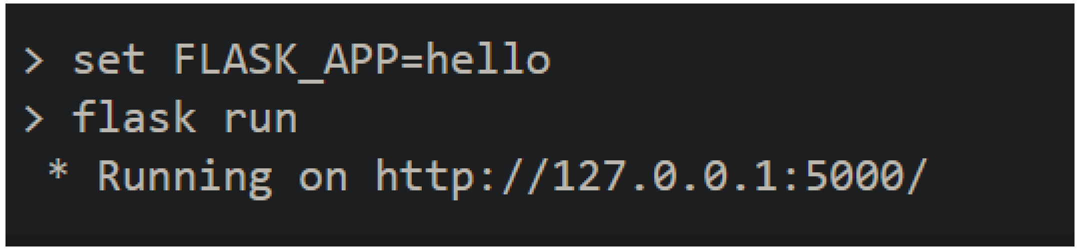


myenv
microblog

To serve flask...

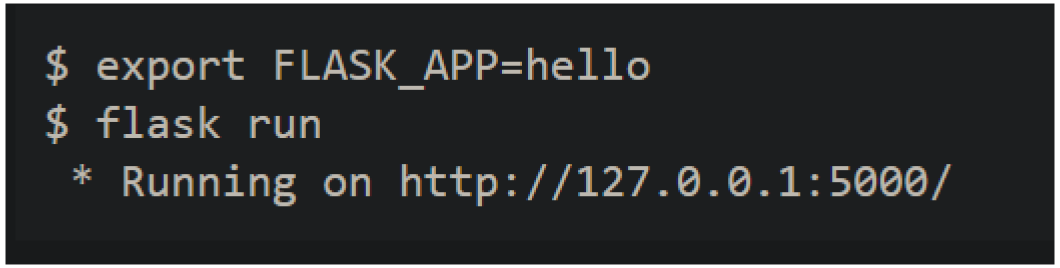
- To run the application, use the flask command or python -m flask.
- Before you can do that you need to tell your terminal the application to work with by exporting the FLASK_APP environment variable:

Windows



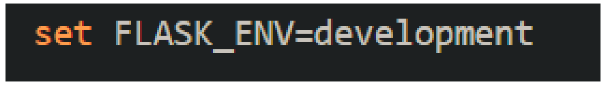
```
> set FLASK_APP=hello
> flask run
 * Running on http://127.0.0.1:5000/
```

MacOS/Bash



```
$ export FLASK_APP=hello
$ flask run
 * Running on http://127.0.0.1:5000/
```

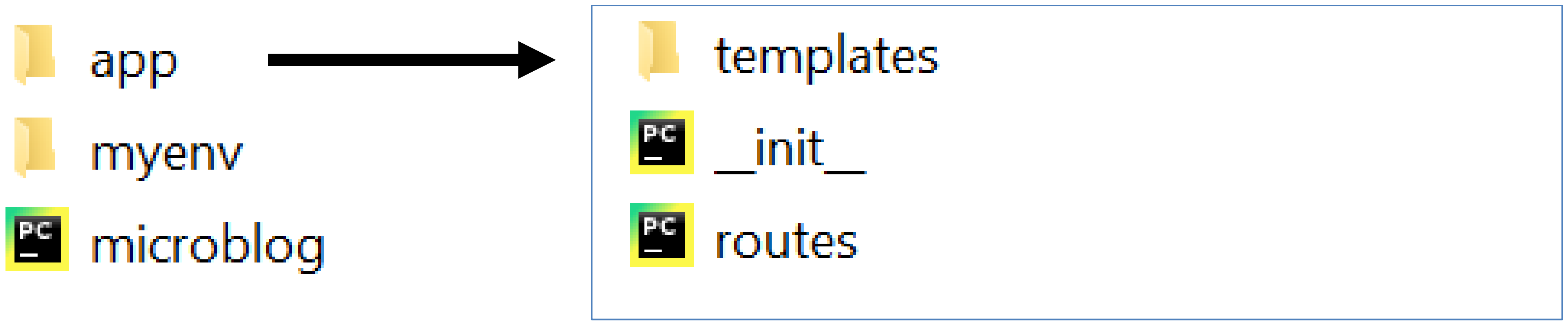
- If you need to debug (inevitable in the future), add this



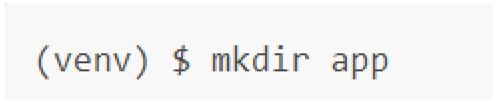
```
set FLASK_ENV=development
```

Basic Structure

- Create folder **app**:
  - Create **__init__.py**
  - Create **routes.py**
  - Create folder **‘templates’**

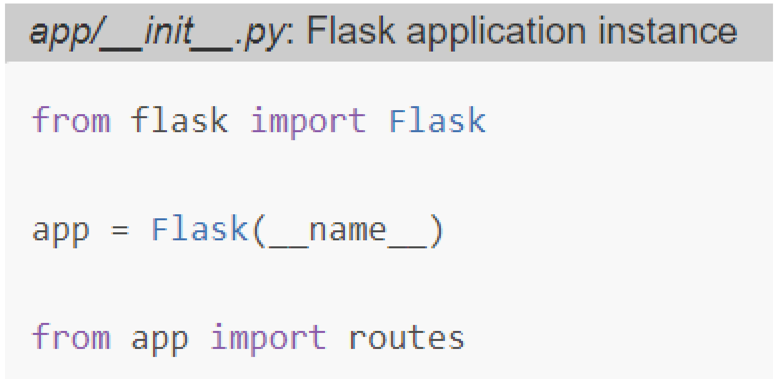


Basic Structure

- Create folder **app**: 

```
(venv) $ mkdir app
```

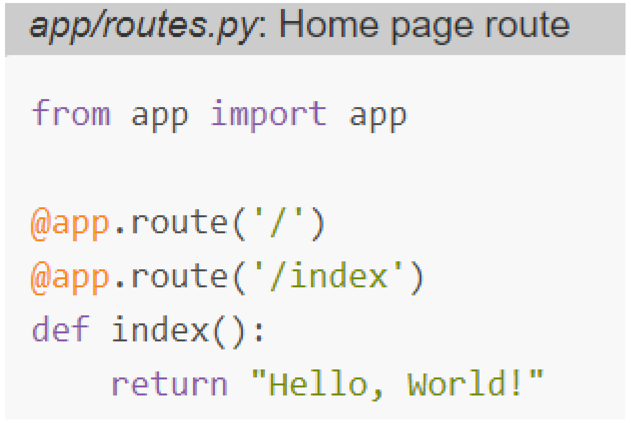
  - Create **__init__.py**



app/__init__.py: Flask application instance

```python
from flask import Flask

app = Flask(__name__)

from app import routes
```

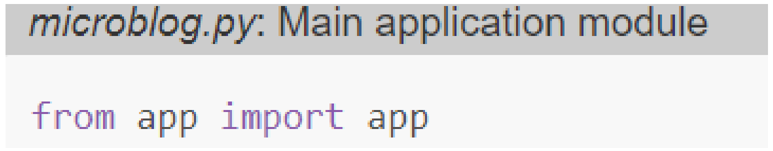
  - Create **routes.py**



app/routes.py: Home page route

```python
from app import app

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

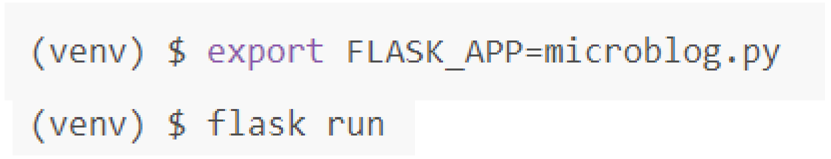
- Create **microblog.py**



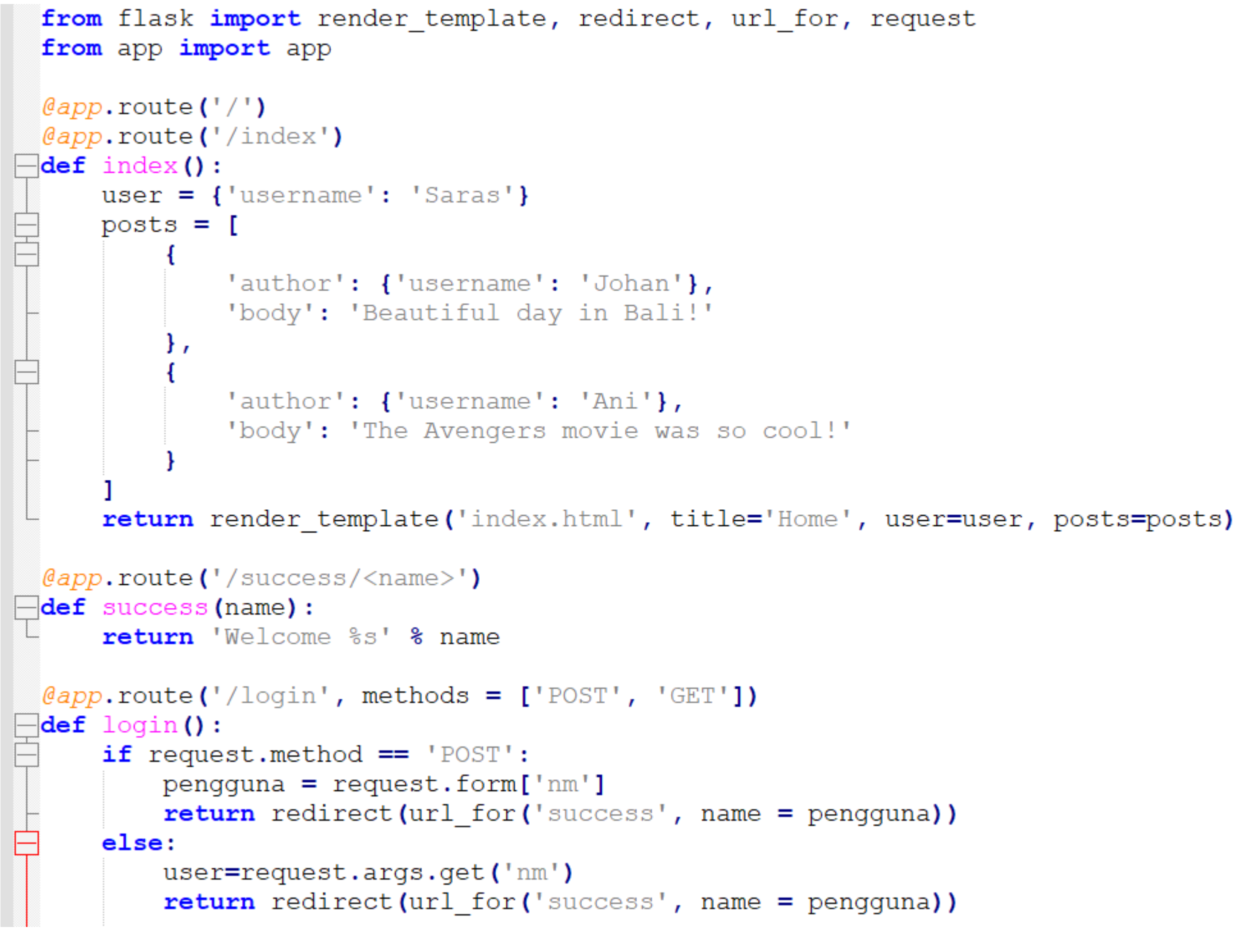
microblog.py: Main application module

```python
from app import app
```

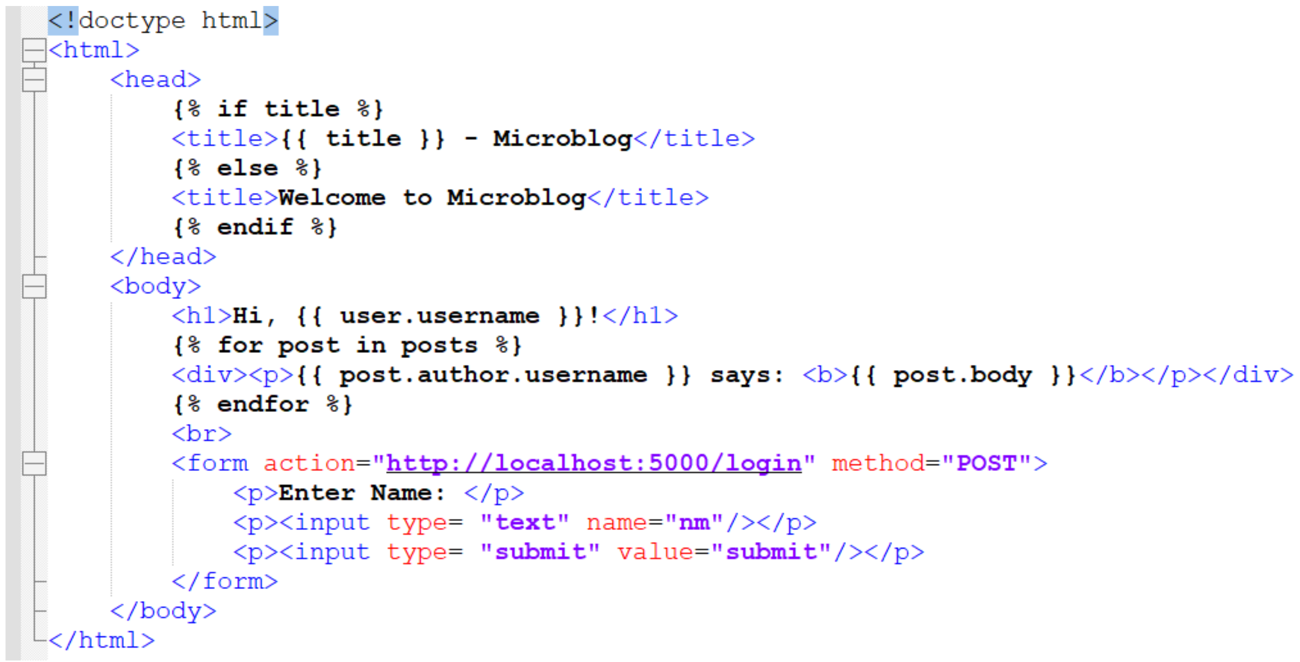
- Run app



```
(venv) $ export FLASK_APP=microblog.py
(venv) $ flask run
```



```python
from flask import render_template, redirect, url_for, request
from app import app

@app.route('/')
@app.route('/index')
def index():
    user = {'username': 'Saras'}
    posts = [
        {
            'author': {'username': 'Johan'},
            'body': 'Beautiful day in Bali!'
        },
        {
            'author': {'username': 'Ani'},
            'body': 'The Avengers movie was so cool!'
        }
    ]
    return render_template('index.html', title='Home', user=user, posts=posts)

@app.route('/success/<name>')
def success(name):
    return 'Welcome %s' % name

@app.route('/login', methods = ['POST', 'GET'])
def login():
    if request.method == 'POST':
        pengguna = request.form['nm']
        return redirect(url_for('success', name = pengguna))
    else:
        user=request.args.get('nm')
        return redirect(url_for('success', name = pengguna))
```

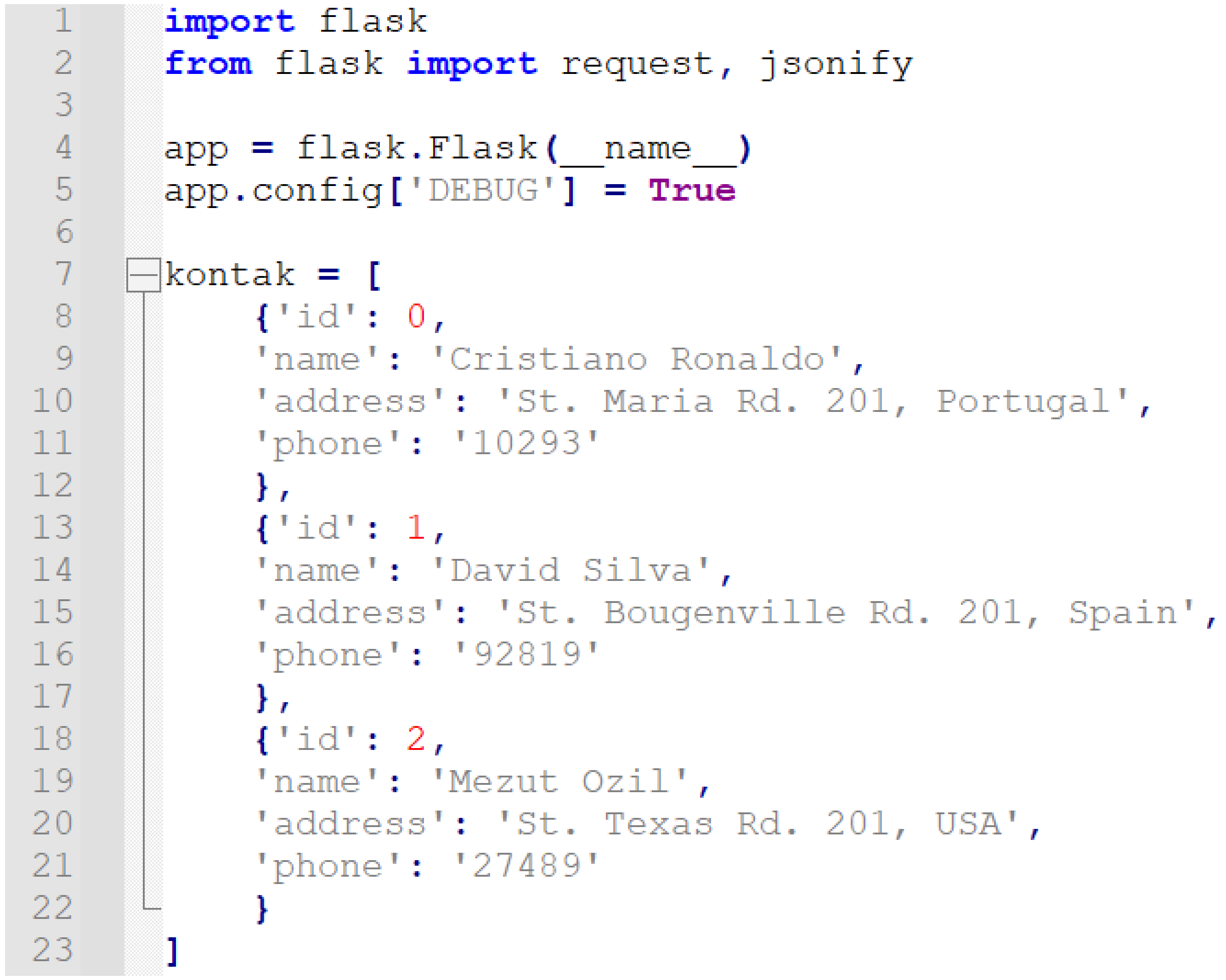


```html
<!doctype html>
<html>
    <head>
        {% if title %}
        <title>{{ title }} - Microblog</title>
        {% else %}
        <title>Welcome to Microblog</title>
        {% endif %}
    </head>
    <body>
        <h1>Hi, {{ user.username }}!</h1>
        {% for post in posts %}
        <div><p>{{ post.author.username }} says: <b>{{ post.body }}</b></p></div>
        {% endfor %}
        <br>
        <form action="http://localhost:5000/login" method="POST">
            <p>Enter Name: </p>
            <p><input type= "text" name="nm"/></p>
            <p><input type= "submit" value="submit"/></p>
        </form>
    </body>
</html>
```

Another trials

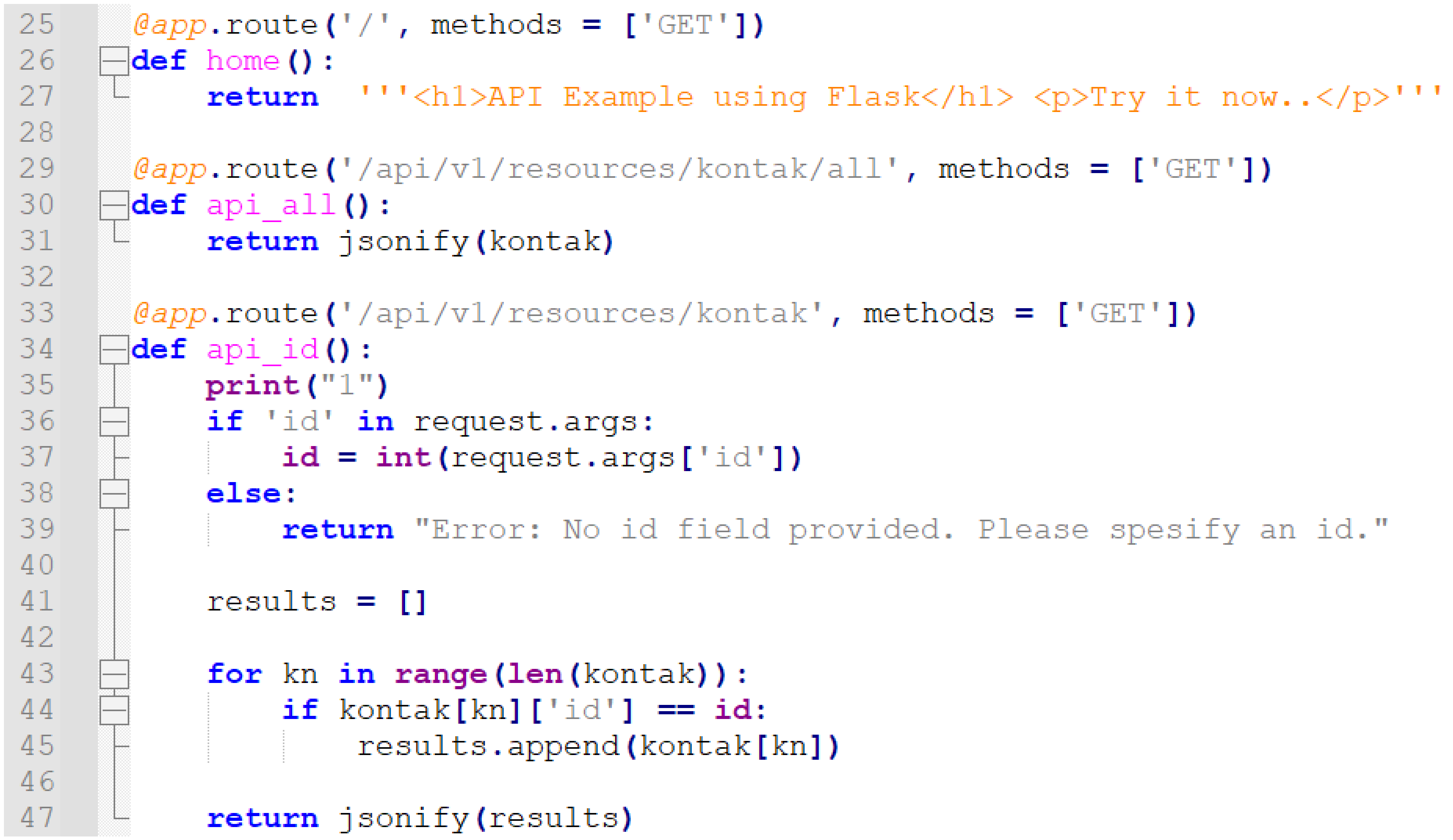
- Left for controller
- Right for view / UI

Create FlaskAPI Using traditional flask routing



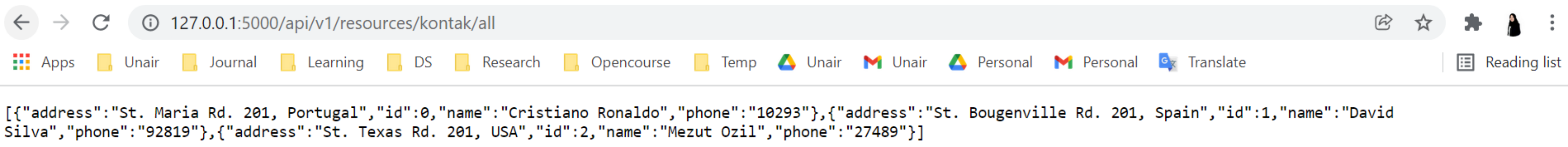
```python
import flask
from flask import request, jsonify

app = flask.Flask(__name__)
app.config['DEBUG'] = True

kontak = [
    {'id': 0,
    'name': 'Cristiano Ronaldo',
    'address': 'St. Maria Rd. 201, Portugal',
    'phone': '10293'
    },
    {'id': 1,
    'name': 'David Silva',
    'address': 'St. Bougenville Rd. 201, Spain',
    'phone': '92819'
    },
    {'id': 2,
    'name': 'Mezut Ozil',
    'address': 'St. Texas Rd. 201, USA',
    'phone': '27489'
    }
]
```

Create FlaskAPI Using traditional flask routing

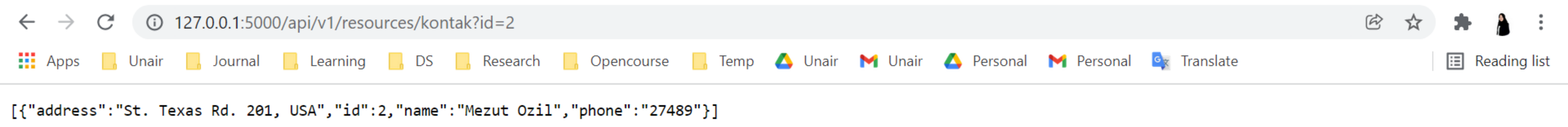


```python
@app.route('/', methods = ['GET'])
def home():
    return  '''<h1>API Example using Flask</h1> <p>Try it now..</p>'''

@app.route('/api/v1/resources/kontak/all', methods = ['GET'])
def api_all():
    return jsonify(kontak)

@app.route('/api/v1/resources/kontak', methods = ['GET'])
def api_id():
    print("1")
    if 'id' in request.args:
        id = int(request.args['id'])
    else:
        return "Error: No id field provided. Please spesify an id."

    results = []

    for kn in range(len(kontak)):
        if kontak[kn]['id'] == id:
            results.append(kontak[kn])

    return jsonify(results)
```

Create FlaskAPI Using traditional flask routing



```
[{"address":"St. Maria Rd. 201, Portugal","id":0,"name":"Cristiano Ronaldo","phone":"10293"},{"address":"St. Bougenville Rd. 201, Spain","id":1,"name":"David
Silva","phone":"92819"},{"address":"St. Texas Rd. 201, USA","id":2,"name":"Mezut Ozil","phone":"27489"}]
```



```
[{"address":"St. Texas Rd. 201, USA","id":2,"name":"Mezut Ozil","phone":"27489"}]
```

Now try!

- Add another routes / pages (re-create your personal blog!)
- Try to import external / pre-made html (will be referred as view)
- Try to import python functions from other py files
- Try to serve csv data as JSON (for API), and consume with python requests.
- Any ideas?

Reference:
flask.palletsprojects.com