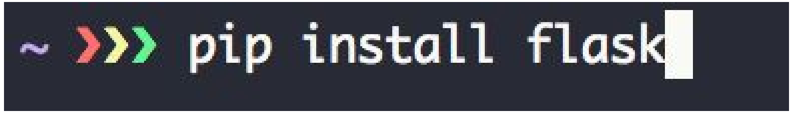
Introduction to Flask

A short introduction to the Flask microframework

Why Flask?

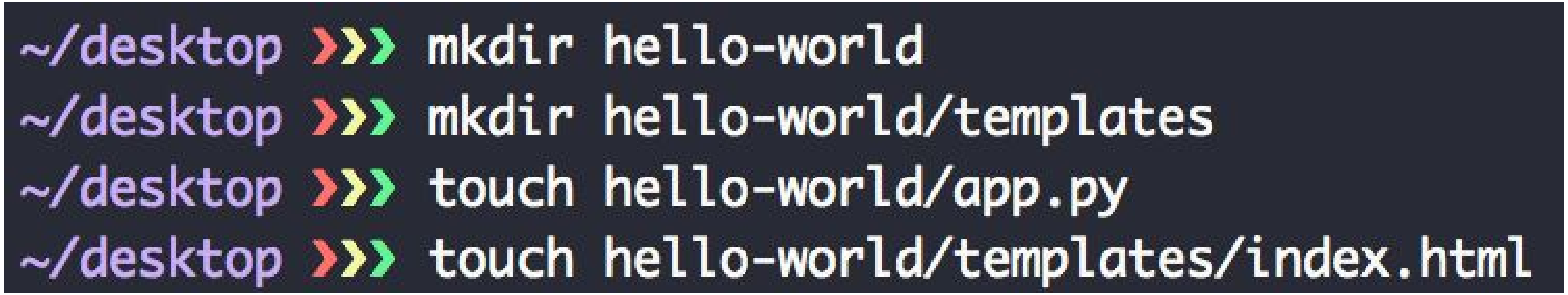
- Easy to learn
- Awesome documentation
- Vast array of extensions
- Routes by decorators*
- Runs on Python 2.7 & 3.3

Installing Flask



```
~ ››› pip install flask
```

Project Structure



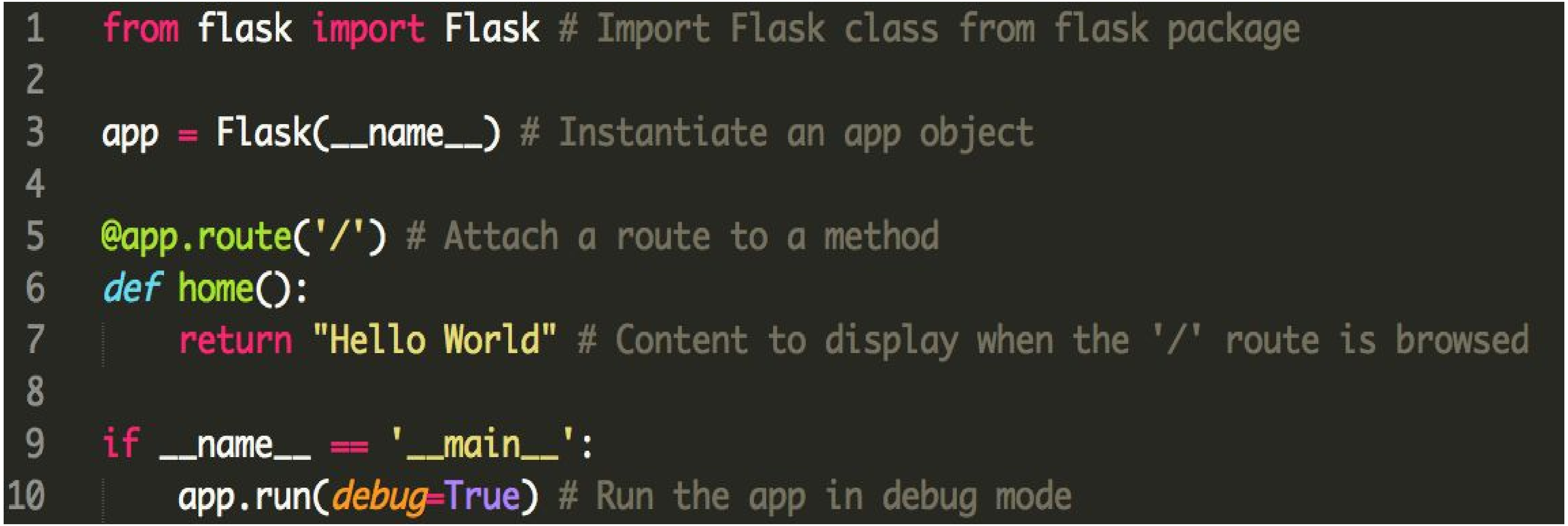
```
~/desktop >>> mkdir hello-world
~/desktop >>> mkdir hello-world/templates
~/desktop >>> touch hello-world/app.py
~/desktop >>> touch hello-world/templates/index.html
```

- The templates folder contains all the template files
- ***app.py*** contains the python code for the web app

Anatomy of a Flask Application

1. Initialize and configure application
2. Define routes and view functions
3. Start development server!

Anatomy of a Flask Application

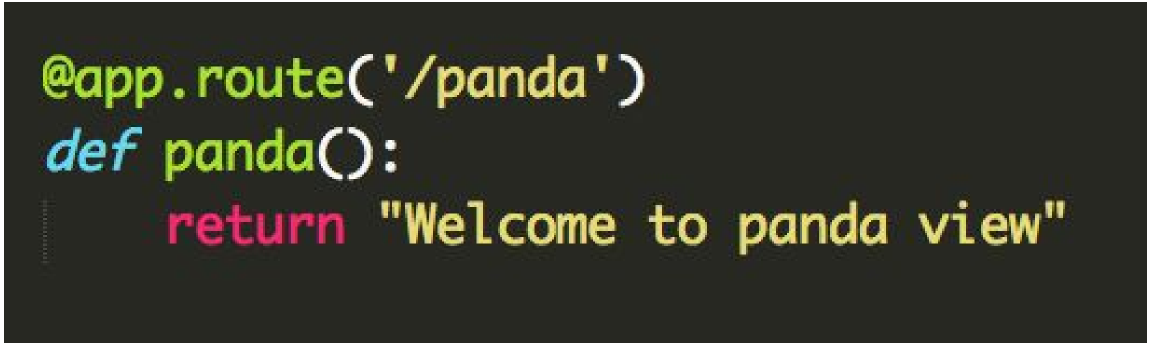


```python
from flask import Flask # Import Flask class from flask package

app = Flask(__name__) # Instantiate an app object

@app.route('/') # Attach a route to a method
def home():
    return "Hello World" # Content to display when the '/' route is browsed

if __name__ == '__main__':
    app.run(debug=True) # Run the app in debug mode
```

Open the terminal and run ***python app.py***, *open http://localhost:5000*

Routes In Flask

Static Routes:

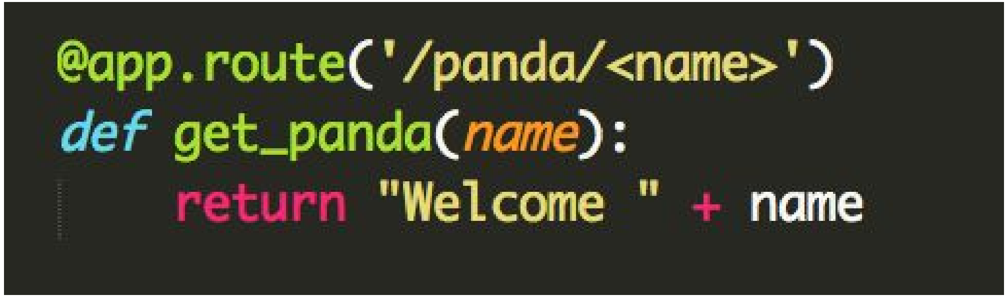


```
@app.route('/panda')
def panda():
    return "Welcome to panda view"
```

- Routes are declared using *decorators* ie the “@” syntax
- A callback method is attached to each route
- You can browse to the url ***/panda*** and see the view binded to that specific route

Routes In Flask

Dynamic Routes:

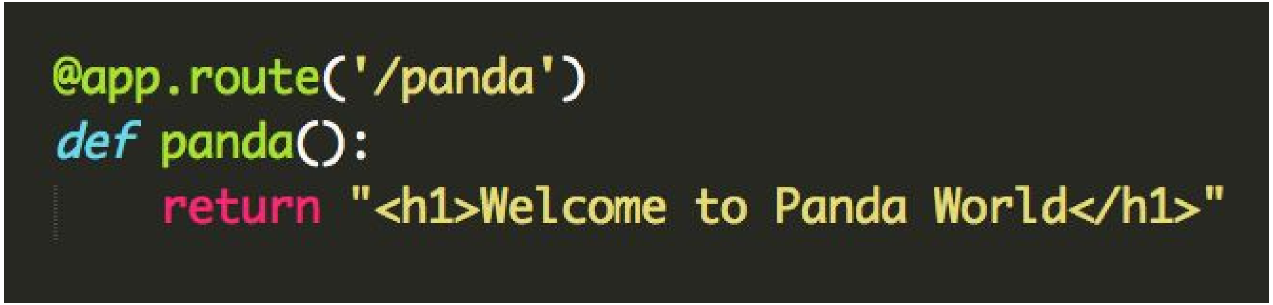


```
@app.route('/panda/<name>')
def get_panda(name):
    return "Welcome " + name
```

- You can add variable parts to a URL.
- These special sections are marked as ***<variable_name>***
- The name of the variable is passed into the callback function

Routes In Flask

Return simple HTML from view function



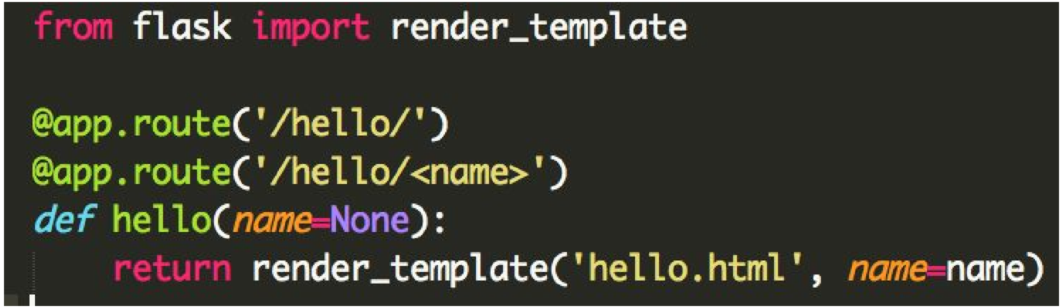
```
@app.route('/panda')
def panda():
    return "<h1>Welcome to Panda World</h1>"
```

- Simple HTML tags can also be used inside the strings
- These are then rendered in the browser

Templates In Flask

- Generating HTML from within python is not fun.
- We use the ***Jinja 2*** templating engine provided by flask
- Flask automatically looks for templates inside the ***templates*** folder
- We use the ***render_template*** method to render a template

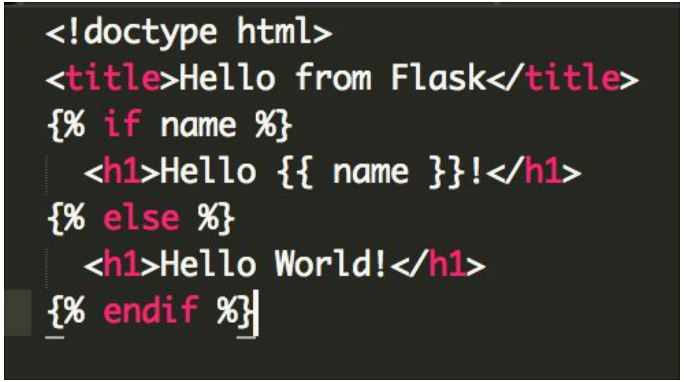
Templates In Flask



```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)
```

- ***render_template*** method is imported from the flask package
- Multiple routes binded to a single view function
- ***name*** variable is ***None*** by default

Templates In Flask

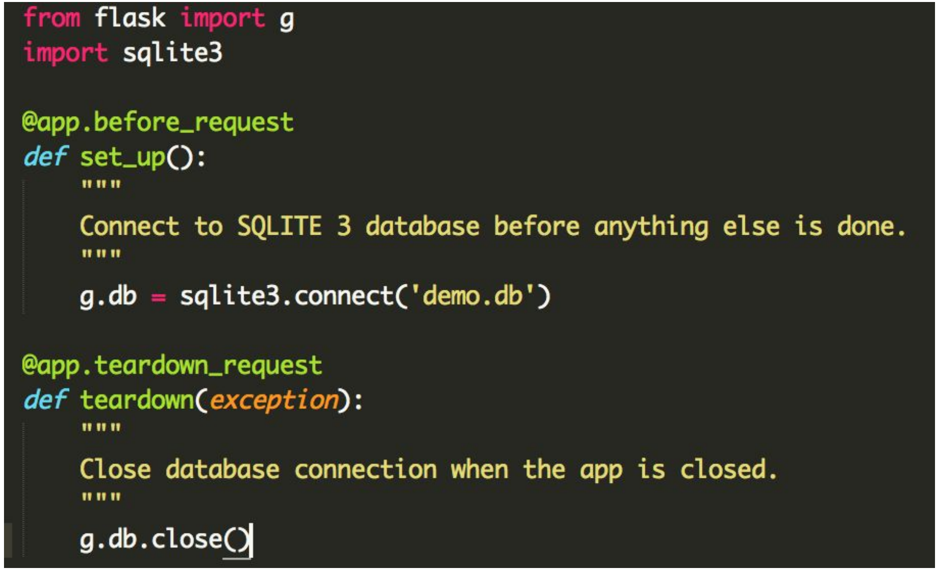


```
<!doctype html>
<title>Hello from Flask</title>
{% if name %}
  <h1>Hello {{ name }}!</h1>
{% else %}
  <h1>Hello World!</h1>
{% endif %}
```

- First block is executed if ***name*** is not none
- *{{ **var_name** }}* is used for template variables

Databases In Flask

- SQLite can be used with Flask



```
from flask import g
import sqlite3

@app.before_request
def set_up():
    """
    Connect to SQLITE 3 database before anything else is done.
    """
    g.db = sqlite3.connect('demo.db')

@app.teardown_request
def teardown(exception):
    """
    Close database connection when the app is closed.
    """
    g.db.close()
```