Web Programming
Server-side programming I.

Krisztian Balog | University of Stavanger

Recall from Last Time

- HTTP requests
  - GET, POST

- HTTP responses

Recall from Python

- A python file can be executed on its own (i.e., *script*) or can be imported into other python code (i.e., *module*)

- When Python reads a source file, it executes all the code in it

- **__name__** is a special variable (automatically set by Python) set to the scope in which the top-level code executes
  - Set to **"__main__"** when read from standard input, a script, or from an interactive prompt

- To be able to use it as a module, as well as a standalone script

```python
if __name__ == "__main__":
    main()
```

main() will only execute if the file was run as a script, not when it was imported as a module

Recall from Python

- String formatting
  - Positional arguments

    ```python
    str = "This is the {} of {}"
    print(str.format("homepage", "John"))
    ```

  - Named placeholders

    ```python
    str = "This is the {property} of {name}"
    data = {"name": "John", "property": "homepage"}
    print(str.format(**data))
    ```

- **format()** knows aligning, truncating, etc.

- See https://pyformat.info/

Recall from Python

- Dictionaries

- Iterating a dictionary

- Iterating a dictionary sorted

- Nested dictionaries

Today

- Making a simple server-side application
  - Serve static content (css files, html files, images, etc.)
  - Generate dynamic content
  - Handle input from URL and forms
  - Use templates

Flask

- Framework for server-side web application development

- Included in the Anaconda distribution (i.e., no need to install anything)

- Practical notes
  - Put each application in a separate folder (even if it consist of a single file)
  - Don't call your application `flask` (that would conflict with Flask itself)

A Minimal Web Application

 examples/python/flask/0_minimal/app.py

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

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

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

A Minimal Web Application

 examples/python/flask/0_minimal/app.py

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

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


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

Create an instance of the Flask class
(the argument is the name of the module)

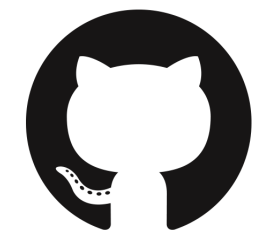# A Minimal Web Application

 examples/python/flask/0_minimal/app.py

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

@app.route("/")
def hello_world():
    return "Hello, world!"

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

The **app.route** decorator tells Flask which URL should trigger this function

Exercise #0

https://github.com/kbalog/web-programming/tree/master/
exercises/python/flask

Routing

- The **route()** decorator is used to bind a function to a URL

- Add variable parts to the URL by marking them as **\<varname\>**

```python
@app.route("/")
def index():
    return "Index Page"

@app.route("/info")
def hello():
    return "This is a static info page"

@app.route("/user/<username>")
def user(username):
    return "Showing the profile page for user {}".format(username)
```
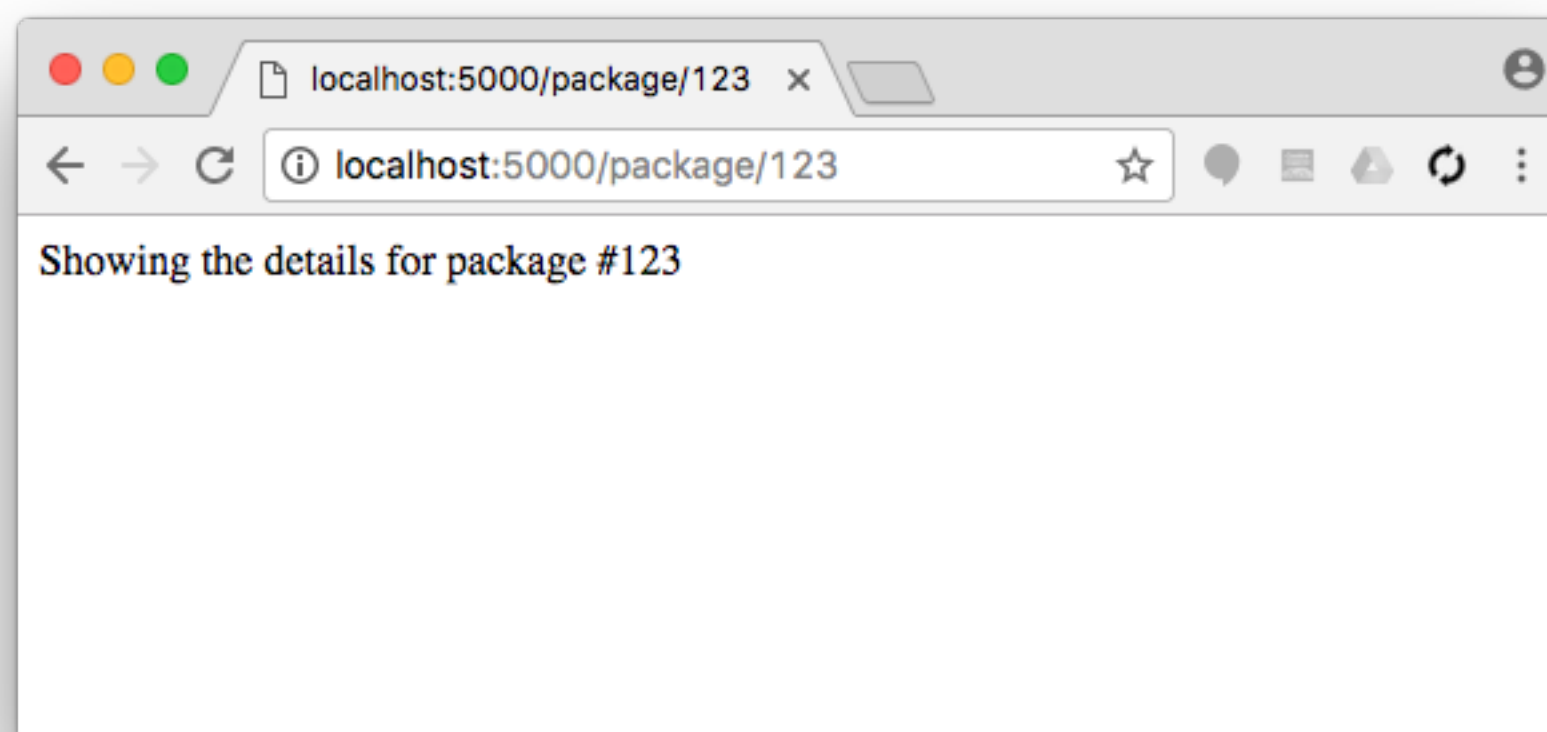
Variable rules

- **`<converter:varname>`** — optionally, a converter may be used to convert the input variable to the specified format

- Converters:
  - string (default)
  - int
  - float
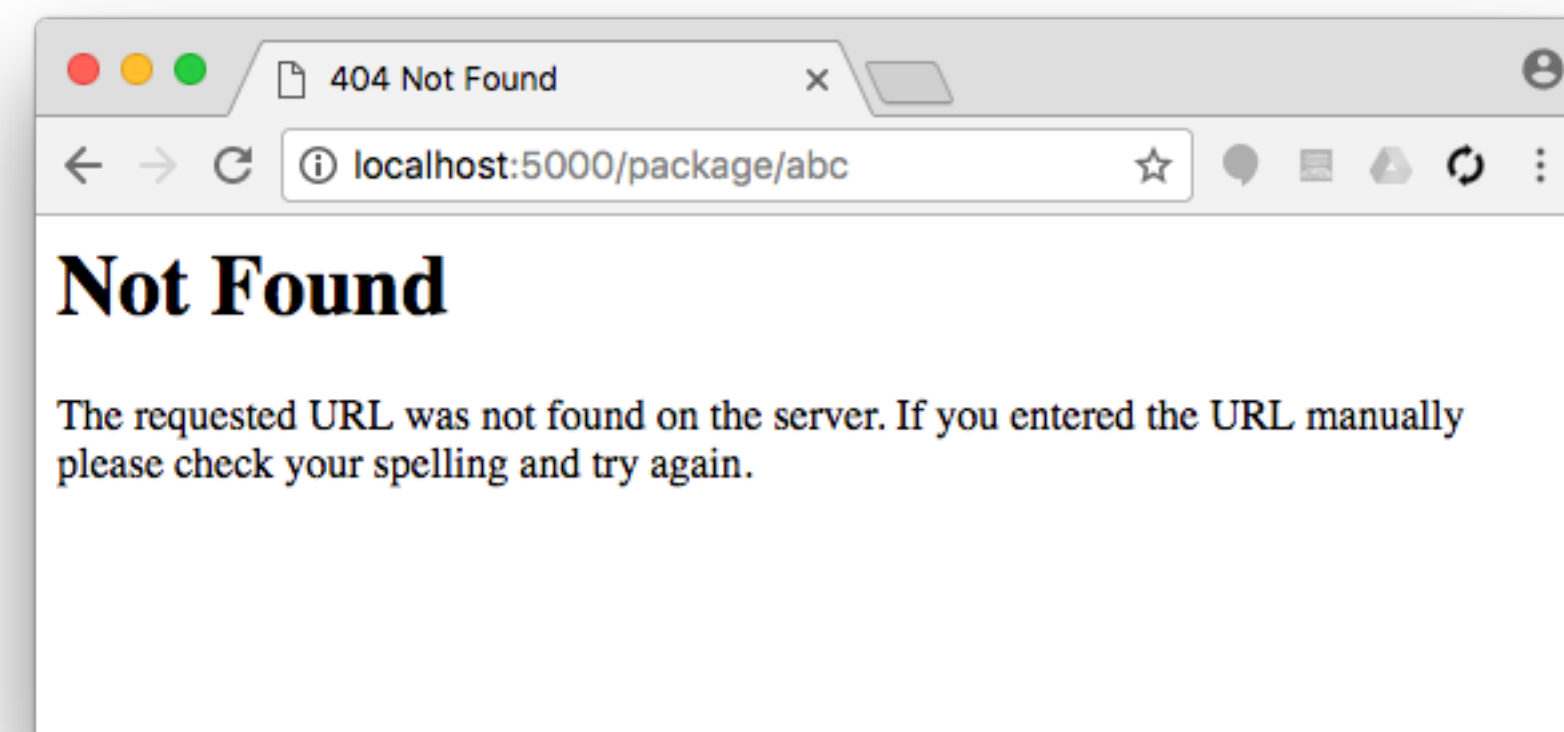 - path (same as the default, but also accepts slashes)
  - …

Example

```python
@app.route("/package/<int:package_id>")
def package(package_id):
    return "Showing the details for package #{}".format(package_id)
```

http://localhost:5000/package/123

Showing the details for package #123

http://localhost:5000/package/abc

Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

URL Building

- **`url_for()`** — generates a URL for a specific function
  - first argument is the function, optionally a number of additional arguments corresponding to the variable part of the URL rule
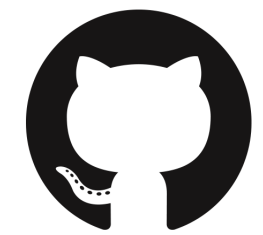
```python
@app.route("/")
def index():
    # ...

@app.route("/user/<username>")
def user(username):
    # ...
```

```python
from flask import url_for

print(url_for("index"))  # /
print(url_for("user", username="JohnDoe"))  # /user/JohnDoe
```

Exercise #1

 https://github.com/kbalog/web-programming/tree/master/ **exercises/python/flask**

Serving Static Files

- Dynamic web applications also need static files (css, javascript, images, etc.)

- Keep them under a folder called **static**

- To generate URLs for them use the special **static** endpoint name

```
url_for("static", filename="style.css")
```

Example

```python
HTML_FRAME_TOP = "<!DOCTYPE HTML>\n<html>\n<head>\n<title>My site</title>\n" \
                 "<link rel=\"stylesheet\" href=\"{css}\"/>\n</head>\n<body>\n"
HTML_FRAME_BOTTOM = "</body>\n</html>\n"


@app.route("/")
def index():
    html = HTML_FRAME_TOP.replace("{css}", url_for("static", filename="style.css"))
    html += "<h1>Hello world</h1>"
    html += HTML_FRAME_BOTTOM
    return html
```

we replace the string {css} with the URL generated for the static "style.css" file

Accessing Request Data

- In Flask, this information is provided by the global **request** object

```python
from flask import request

@app.route("/login", methods=["POST", "GET"])
def login():
    error = None
    if request.method == "POST":
        if valid_login(request.form["username"],
                       request.form["password"]):
            # do something
        else:
            error = "Invalid username/password"
```

Accessing Request Data

- Accessing parameters submitted in the URL

- These are contained in **`request.args`** (dict)
  - Checking if a param has been provided

```python
if "name" in request.args:
    print(request.args["name"])
```

  - Getting param with default value

```python
print(request.args.get("name", ""))
```

  - Iterating all parameters

```python
for k, v in request.args.items():
    print("{:20} : {}".format(k, v))
```

Example

```python
from flask import Flask, url_for, redirect, request

@app.route("/")
def index():
    return redirect(url_for("static", filename="form.html"))
```

localhost:5000/

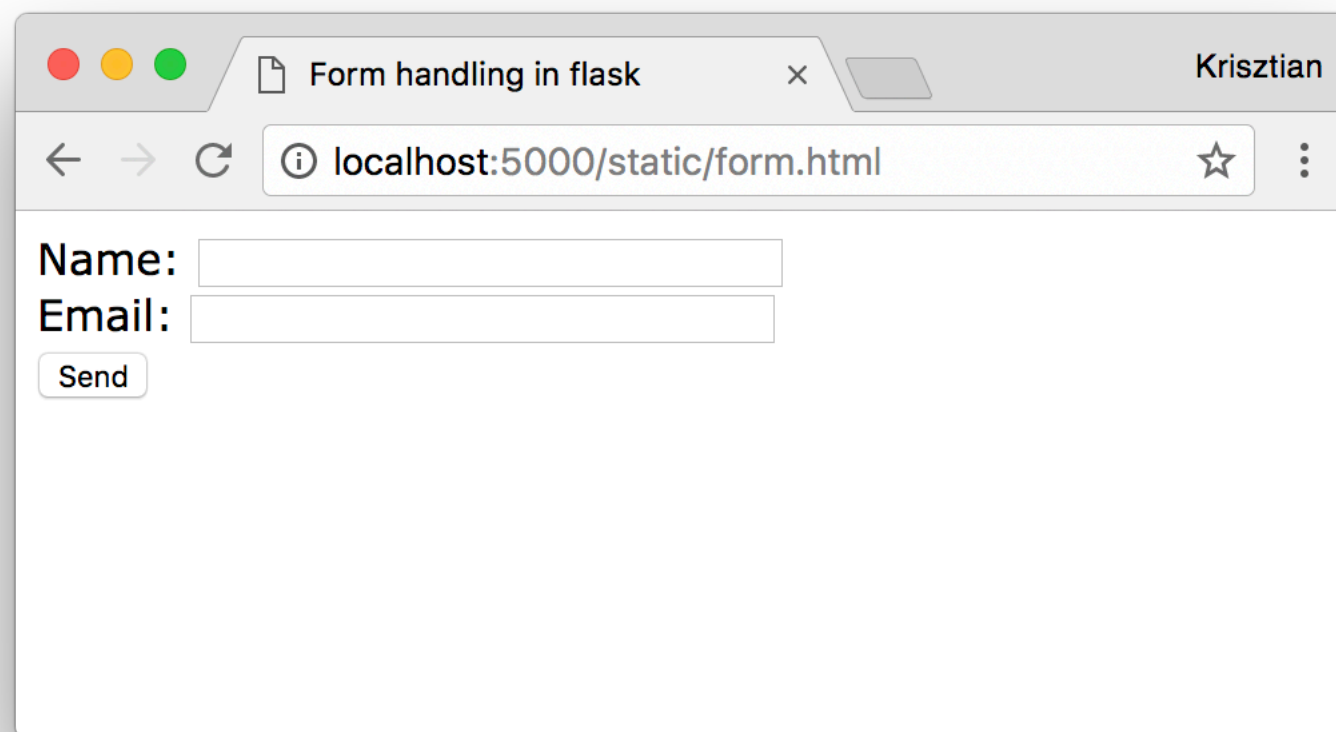**localhost:5000/static/form.html**

redirect to static form page from web root

Example

 examples/flask/3_forms/app.py

```python
@app.route("/sendform", methods=["POST"])
def sendform():
    html = HTML_FRAME_TOP.replace("{css}", url_for("static", filename="style.css"))
    html += "Name: " + request.form["name"] \
            + "<br />" \
            + "Email: " + request.form["email"]
    html += HTML_FRAME_BOTTOM
    return html
```
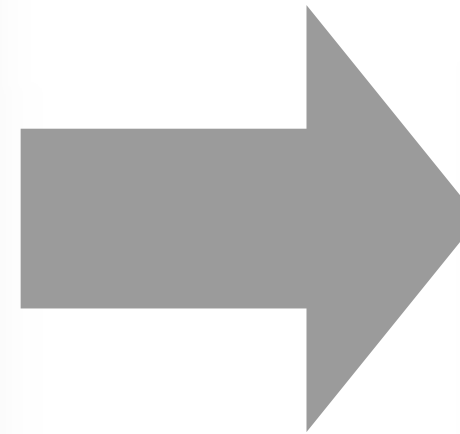
localhost:5000/static/form.html
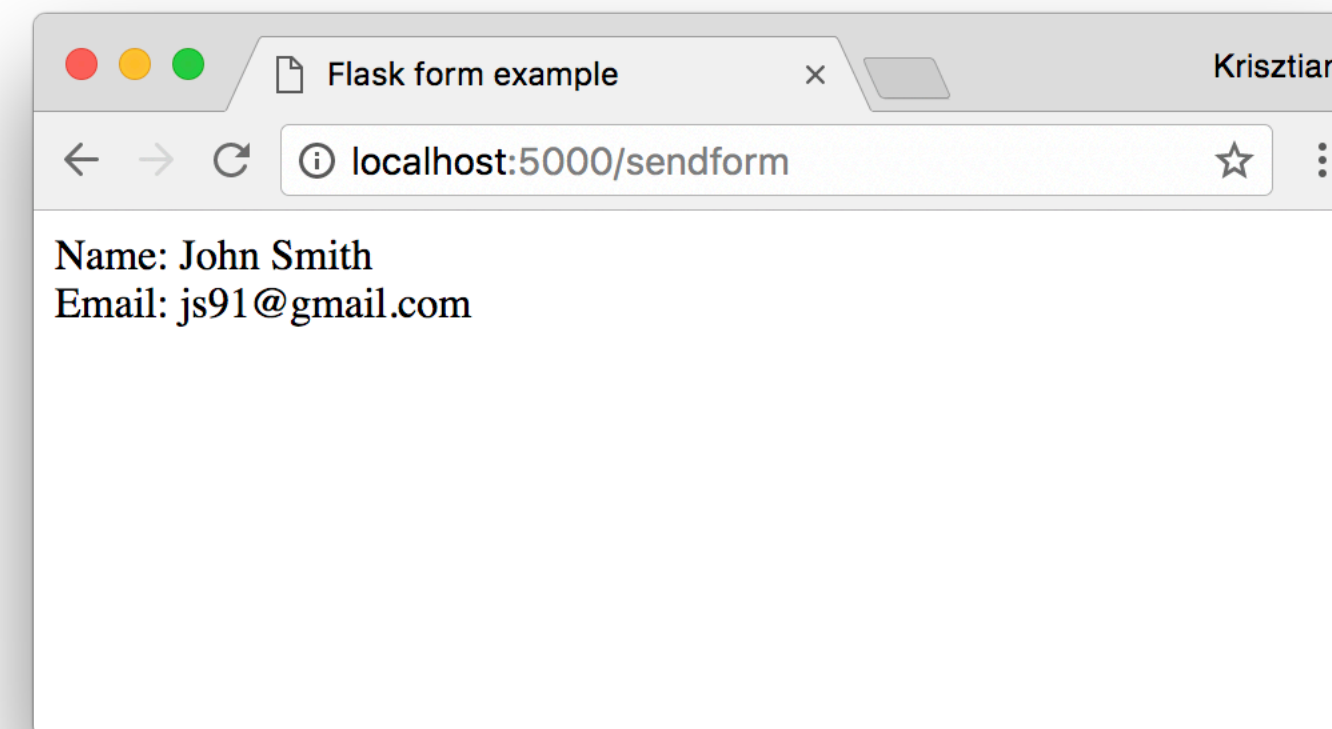
http://localhost:5000/sendform