

PYTHON

- Python is an interpretative language
 - ↳ interpreter goes through code line by line /
no compilation of code to machine code

[skipped Python Basics]

HTTP

- search website address → sends HTTP request
- server handles request and sends back response which is then rendered in the browser

FLASK

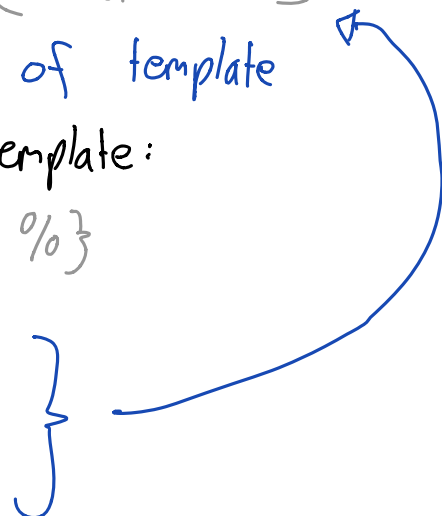
- microframework to create web applications (handle requests)
- application.py → main file
- `app = Flask(__name__)` ← the current file represents newly created application
- `@app.route("/<page-route>")` ← route that triggers decorated function
- `$ flask run` (runs flask application in cwd)
 - ↳ relies on environment variable
 - `export FLASK_APP=file.py`
- `@app.route("/<string:name>")` → template for set of routes
 - ↳ passes name as argument to func

- return String → could return HTML content but very limited and starts to get messy
- HTML templates → **templates** folder
- return render_template("file.html") ← return HTML from file
- pass variables as arguments to render_template
 - ↳ inside file.html: {{ variable }} → Jinja2 syntax
 - ↳ placeholder for variable value
 - gets inserted when rendered
- {% if variable %} → python-like syntax
 - html content
 - {% else %}
 - html content
 - {% endif %}
 - generate different HTML content for different variable values
- {% for var in list %}
 - html content {{ var }}
 - {% endfor %}


LINKING ROUTES

- Text

TEMPLATE INHERITANCE

- basic layout/template HTML
 - `{% block heading %} {%endblock %}`
 - ↳ variable parts of template
 - webpages that use file as template:
 - `{% extends "layout.html" %}`
 - `{% block heading %}`
 - html content
 - `{% endblock %}`
- 

FORMS

- `<form action = " {{ url_for('func') }} " method = "post" > ... input/button... </form>`
 - ↳ ^{set} name attribute
 - `@app.route("/route", methods = ["POST"])`
`def function_route():`
`name = request.form.get("name")`
- 

SESSIONS

- store user-specific data
- append values to list, submit using forms
 - ↳ variables are stored as long as server not shutdown
- global variables are common for all users → concept of sessions

EXAMPLE

```
from flask import session ...
```

```
from flask_session import Session
```

```
app.config["SESSION_PERMANENT"] = False
```

```
app.config["SESSION_TYPE"] = "filesystem"
```

```
⋮
```

```
@app.route("/")
```

```
def func():
```

```
    if session.get("notes") is None:
```

```
        session["notes"] = []
```

```
    session["notes"].append(note)
```