Flask Python micro web framework

Request from the browser → URL rule → **View function** @app.route → return value → **Response** str, dict, tuple → served by → **Client** HTML or JSON

CONCEPTS

Name your file app.py or wsgi.py and you can drop --app. Never name it flask.py.

request, session and g are context locals: they only work while a request or app context is active.

The dev server is not for production: put the app behind a WSGI server such as gunicorn.

RUN

`flask --app <module> run`
Start the dev server

`flask --app <module> run --debug`
Run with reloader and debugger

`flask run -p <port>`
Listen on another port

`flask run --no-reload`
Debugger without the reloader

CONFIG & DISCOVERY

`flask --app <module>:<name> routes`
Pick the app variable explicitly

`flask --app "<module>:create_app('<arg>')" run`
Call an app factory with arguments

`FLASK_APP=<module>`
Environment form of --app

`pip install python-dotenv`
Enable .env and .flaskenv loading

SHELL & ROUTES

`flask shell`
Python shell with the app context pushed

`flask routes`
List routes, endpoints and methods

CUSTOM COMMANDS

`@app.cli.command("<name>")`
Register flask <name>

CUSTOM COMMANDS (cont.)

`@click.option("--<flag>")`
Add an option to the command

ROUTING

`@app.route("/")`
Bind a URL to a view

`@app.route("/user/<name>")`
Dynamic URL part

`@app.route("/post/<int:id>")`
Typed converter

`@app.route("/login", methods=["GET", "POST"])`
Accept more HTTP methods

`@app.post("/login")`
Method shortcut decorator

`url_for("<endpoint>", id=1)`
Build the URL for a view

`url_for("static", filename="<file>")`
URL of a static file

`Blueprint("<name>", __name__)`
Group routes into a module

`app.register_blueprint(<bp>, url_prefix="/<p>")`
Attach a blueprint

REQUESTS

`request.method`
HTTP method of the request

`request.args.get("<key>")`
Query string value

`request.form["<key>"]`
Form field from POST

`request.get_json()`
Parsed JSON body

`request.files["<name>"]`
Uploaded file

`file.save(secure_filename(file.filename))`
Store an upload safely

RESPONSES

`return "<p>Hi</p>"`
String becomes a 200 HTML response

`return {"a": 1}`
Dict or list becomes JSON

`return <body>, 404`
Body plus status code

`redirect(url_for("<endpoint>"))`
Redirect to another view

`abort(404)`
Stop with an HTTP error

RESPONSES (cont.)

`resp.set_cookie("<name>", "<value>")`
Set a cookie on a response

`@app.errorhandler(404)`
Custom error page

TEMPLATES & STATE

`render_template("<file>.html", x=1)`
Render templates/<file>.html

`{{ x }}`
Print an escaped value in Jinja

`{% extends "base.html" %}`
Inherit a layout

`app.secret_key = <bytes>`
Key needed for sessions and flash

`session["<key>"] = <value>`
Per-user data in a signed cookie

`flash("<msg>")`
Message for the next request