PYTHON

Flask

FULL COURSE

Web Apps • APIs • Python



LONG VIDEO • 10 MIN

PYTHON WEB • FLASK FULL COURSE

Python Flask Full Course

Routes, templates, forms, databases, REST APIs and deployment

FREE FULL COURSE • MyWebUniversity.com



Wahidullah Lutfy
Founder, MyWebUniversity • CS Teacher

FLASK

What You Will Learn

How web apps work
Browsers, servers, requests and responses.

Routes and views
URLs, dynamic routes and HTTP methods.

Templates
Jinja2 HTML templates, inheritance and static files.

Forms
Reading input, validating it, and redirecting.

Database
Flask-SQLAlchemy models and queries.

APIs and deployment
JSON APIs, blueprints, security and production servers.

Python Flask • Full Course

PART 1 · FIRST STEPS

FLASK

Hello, Flask!

app.py — Python 3.13

```
$ python -m venv .venv && source .venv/bin/activate
$ pip install flask


from flask import Flask
app = Flask(__name__)

@app.route("/")
def home():
    return "<h1>Hello, Flask!</h1>"


$ flask --app app run --debug
 * Running on http://127.0.0.1:5000
 * Debugger is active!
```

The app object

Flask(__name__) creates your application.

@app.route

A decorator that connects a URL to a Python function, called a view.

Debug mode

Reloads automatically when you save, and shows helpful error pages. Never in production.

FLASK

Routes, Variables and Methods

app.py — Python 3.13

```
@app.route("/hello/<name>")
def hello(name):
    return f"Hello, {name}!"


@app.route("/square/<int:n>")
def square(n):
    return {"n": n, "square": n * n}       # dict -> JSON automatically



from flask import request
@app.route("/search")
def search():
    q = request.args.get("q", "")
    return f"You searched for: {q}"
```

Dynamic routes

<name> captures part of the URL; <int:n> converts it to a number.

Return types

Return a string for HTML, or a dict or list for JSON.

request.args

Reads query parameters like /search?q=python.

FLASK

Templates with Jinja2

templates/ — Python 3.13

```
# templates/base.html
<html><body><nav>MyWebUniversity</nav>
 {% block content %}{% endblock %}
</body></html>


# templates/courses.html
{% extends "base.html" %}
{% block content %}
 {% for c in courses %}<li>{{ c }}</li>{% endfor %}
{% endblock %}


from flask import render_template
return render_template("courses.html", courses=["Python", "Linux"])
```

Templates folder

HTML files live in templates/; CSS and images in static/.

{{ }} and {% %}

{{ }} shows a value; {% %} runs logic like for and if.

Inheritance

One base.html for the layout; each page fills in its own block.

Python Flask • Full Course

PART 2 · FORMS AND DATABASES

FLASK

Forms: GET, POST and Redirect

app.py — Python 3.13

```
from flask import request, redirect, url_for, flash
@app.route("/contact", methods=["GET", "POST"])
def contact():
   if request.method == "POST":
       email = request.form.get("email", "").strip()

       if "@" not in email:
           flash("Please enter a valid email.")
       else:
           save_message(email, request.form["message"])
           return redirect(url_for("thanks"))
   return render_template("contact.html")

app.config["SECRET_KEY"] = os.environ["SECRET_KEY"]
```

GET shows, POST submits

One view handles both: show the empty form, then process it.

Validate everything

Never trust user input. Check it on the server.

Post / Redirect / Get

Redirect after a successful POST, so refreshing does not submit twice.

FLASK

Databases with Flask-SQLAlchemy

models.py — Python 3.13

```
$ pip install flask-sqlalchemy
from flask_sqlalchemy import SQLAlchemy
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///school.db"
db = SQLAlchemy(app)

class Student(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    score = db.Column(db.Integer, default=0)

with app.app_context(): db.create_all()
db.session.add(Student(name="Ali", score=91)); db.session.commit()
top = db.session.scalars(db.select(Student).where(Student.score >
80)).all()
```

ORM

Python classes become database tables; objects become rows. No raw SQL needed.

SQLite to start

A single file database, built into Python. Switch to PostgreSQL later.

Session

add, then commit to save. select().where() to query.

Python Flask • Full Course

PART 3 · APIS, STRUCTURE AND DEPLOYMENT

FLASK

Building a JSON API

api.py — Python 3.13

```python
@app.get("/api/students")
def list_students():
    rows = db.session.scalars(db.select(Student)).all()
    return [{"id": s.id, "name": s.name, "score": s.score} for s in rows]


@app.post("/api/students")
def create_student():
    data = request.get_json()
    if not data or "name" not in data:
        return {"error": "name is required"}, 400
    s = Student(name=data["name"]); db.session.add(s); db.session.commit()
    return {"id": s.id}, 201
```

@app.get / @app.post

Short forms of route(..., methods=[...]).

Status codes

Return (body, status): 201 Created, 400 Bad Request, 404 Not Found.

Test with requests

requests.post(url, json={...}) from our requests course.

FLASK

Blueprints, Security and Deployment

production — Python 3.13

```
from flask import Blueprint
admin = Blueprint("admin", __name__, url_prefix="/admin")
app.register_blueprint(admin)


$ pip install gunicorn
$ gunicorn -w 4 -b 0.0.0.0:8000 "app:app"
# Windows: pip install waitress && waitress-serve --port=8000 app:app
```

Blueprints

Split a big app into parts: admin, api, blog. Each has its own routes.

Security checklist

Debug off, secret key from env, validate input, hash passwords, HTTPS.

Production server

gunicorn (Linux) or waitress (Windows), behind nginx, or on a cloud platform.

Recap and Practice

Remember

Routes → views → templates. Forms, SQLAlchemy models, JSON APIs, blueprints.

Try it yourself

Build a to-do list app: add, list and delete tasks, with a JSON API.

Watch out

Never deploy with debug on, and never trust user input.