9. Gyakorlat

Python kurzus

Flask,
Jinja2

Flask és Jinja2

A **Flask** és a **Jinja2** két fontos eszköz a Python alapú webfejlesztésben, amelyek együttműködve lehetővé teszik **dinamikus webalkalmazások** létrehozását.

Flask: egy Python webkeretrendszer, amely beépítve használja a **Jinja2 sablonmotort**

A Flask fő jellemzői:

- Webalkalmazások készítése: Flask segítségével könnyen létrehozhatók **REST API**-k és **dinamikus weboldalak**.
- Minimalista: Flask egy "**microframework**": az alapvető funkciókat biztosítja, és szabadon bővíthető különféle bővítményekkel.
- **Routing:** URL-eket társíthatunk Python függvényekhez, pl. az @app.route dekorátorral.
- HTTP metódusok kezelése: Támogatja a **GET, POST, PUT, DELETE** és más HTTP metódusokat.
- **JSON támogatás:** Könnyen kezelhetők JSON adatok, ami hasznos API-k fejlesztésekor.

Flask telepítése: `pip install flask`

Jinja2:

egy sablonmotor, amelyet a Flask használ HTML oldalak dinamikus generálására. Tartalmaz űrlapokhoz, navigációs sávokhoz, gombokhoz, táblázatokhoz és más UI-elemekhez szerkesztési lehetőségeket. Jinja2 sablonnyelvet használ.

Fő jellemzői:

- **Dinamikus HTML generálás**: Lehetővé teszi, hogy Python változókat és logikát ágyazzunk be HTML sablonokba.
- **Vezérlési szerkezetek**: Támogatja az if, for és más vezérlési szerkezeteket a sablonokban.
- **Sablonöröklés**: Segítségével új sablonok készíthetők meglévő sablonok kiterjesztésével, ami megkönnyíti az ismétlődő kód elkerülését.
- **Biztonság**: Automatikusan escape-eli a HTML kódot, hogy megvédjen az XSS támadásoktól.

Együttműködésük: a Flask a Jinja2-t használja a HTML sablonok **renderelésére**:

- A Flask egy Python függvényben adatokat készít elő.
- Az adatokat és változókat átadja egy Jinja2 sablonnak a **render_template()** függvénnyel (a sablon feldolgozása, megjelenítése).
- A Jinja2 sablon a kapott adatokat beilleszti a HTML-be, és a kész oldalt visszaküldi a böngészőnek.
- A sablonokat a **templates** mappában tároljuk, a Flask alapértelmezetten itt keresi a sablonokat.
- HTML oldalon belül {{data}} formában hivatkozhatók változók.
- {% ... %} formában vezérlési szerkezetek is megadhatóak.

Jinja2 működése: https://jinja.palletsprojects.com/en/stable/

Példa:

```
Udv_app/
│
├── app.py                  # Flask alkalmazás belépési pontja.
├── templates/              # HTML sablonok (pl. index.html).
│   └── index.html          # Kezdőlap sablonja.
├── static/                 # Statikus fájlok (CSS, JS, képek).
│   ├── css/                # CSS fájlok.
│   ├── js/                 # JavaScript fájlok.
│   └── images/             # Képek.
└── requirements.txt        # Függőségek listája (pl. Flask).
```

opcionális

app.py:

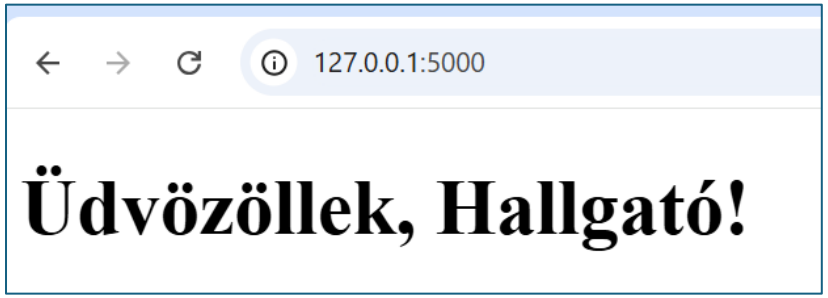
```
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    user = {"name": "Hallgató"}
    return render_template("index.html", user=user)

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

index.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Üdvözlet</title>
</head>
<body>
  <h1>Üdvözöllek, {{ user.name }}!</h1>
</body>
</html>
```

Futtatás terminálból: python app.py

Megnyitás: http://127.0.0.1:5000 (alapértelmezett az 5000-es port)



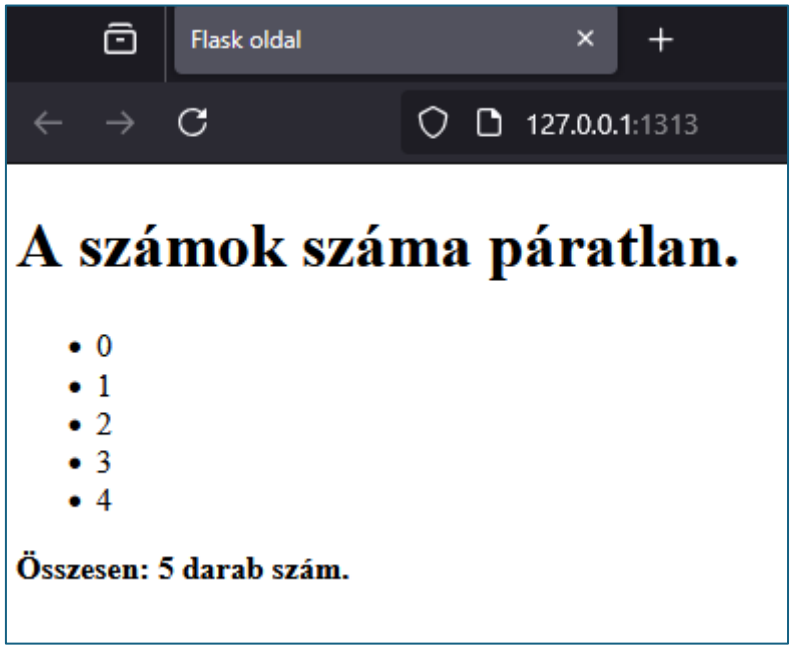
Példa: flask_test.py

```python
from flask import Flask, render_template

app = Flask("flaskpelda")
#futtatás: flask run, vagy python flask_test.py
#fontos a helyes mappaszerkezet!

@app.get("/") #alap oldalra (index.html) vonatkozó kód
def root():
    flask_title = "Flask oldal"
    lst = [i for i in range(5)]
    #ezeket a változókat fogja átadni a HTML oldal számára!
    return render_template("index.html",
                           page_name = flask_title,
                           nums = lst
                           ) #a kulccsal lehet hivatkozni ott rájuk!

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=1313)
```

index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
        …
    <title>{{page_name}}</title>
</head>
<body>
    {% if (nums | length) % 2 == 0 %}
        <h1>A számok száma páros.</h1>
    {% else %}
        <h1>A számok száma páratlan.</h1>
    {% endif %}
    <ul>
        {% for num in nums %}
            <li>{{ num }}</li>
        {% endfor %}
    </ul>
    <b>Összesen: {{(nums | length)}} darab szám.</b>
</body>
</html>
```

Flask példa – Jinja2 template



Fontos a helyes mappaszerkezet!

Django

Magas szintű, full-stack Python webfejlesztési keretrendszer.

Nagyobb webalkalmazások készítésére használatos.

Összetettebb – kell ismerni hozzá a szerver- és kliensoldali webfejlesztési technikákat.

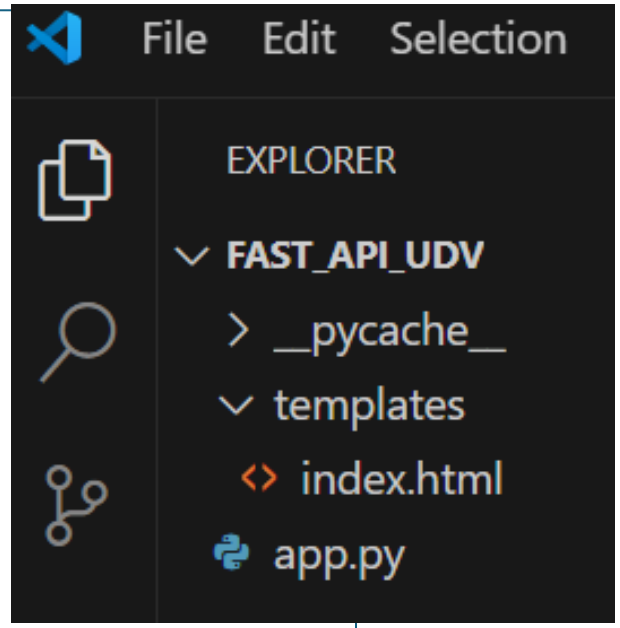
Flask is és Django is összekapcsolható FastAPI-val.

Bővebben Django-ról: https://www.djangoproject.com/

Fast API és Jinja2

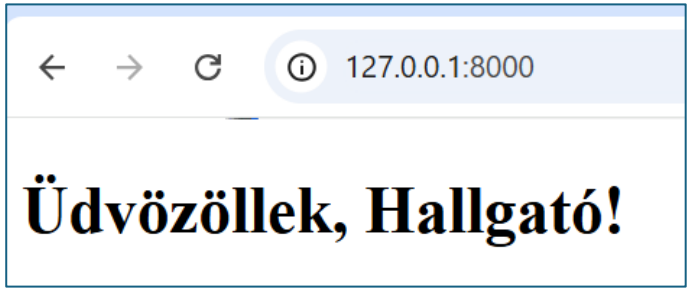
app.py:

```
from fastapi import FastAPI, Request
from fastapi.responses import HTMLResponse
from fastapi.templating import Jinja2Templates
from fastapi.staticfiles import StaticFiles

app = FastAPI()

# Jinja2 sablonok elérési útja
templates = Jinja2Templates(directory="templates")

@app.get("/", response_class=HTMLResponse)
async def index(request: Request):
    user = {"name": "Hallgató"}
    return templates.TemplateResponse("index.html", {"request": request, "user": user})
```



index.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>Üdvözlet</title>
</head>
<body>
    <h2>Üdvözöllek, {{ user.name }}!</h2>
</body>
</html>
```



1. feladat: egyszerű API létrehozása Flask és Jinja2 segítségével

Egyszerű felhasználói lista megjelenítése Flask alkalmazás:

```
from flask import Flask, render_template, jsonify

app = Flask(__name__)

users = [
   {"id": 1, "name": "Anna"},
   {"id": 2, "name": "Béla"}
]

@app.route("/api/users")
def get_users():
   return jsonify(users)

@app.route("/users")
def user_page():
   return render_template("users.html", users=users)
```

Mappa szerkezet:

```
flask_demo1/

|— app.py
|— templates/
|    └— users.html
```

users.html sablon részlet:

```
<h1>Felhasználók</h1>
<ul>
 {% for user in users %}
   <li>{{ user.name }}</li>
 {% endfor %}
</ul>
```

2. feladat: egyszerű Flask alkalmazás készítése:

- egy végpont, JSON formátumban ad vissza adatokat;
- egy HTML oldal megjelenítése Jinja2 sablonnal;
- alap Flask szerkezet.

Mappa szerkezet:

```
flask_demo2/
|— app.py
|— templates/
|    └— users.html
```

app.py:

```
from flask import Flask, jsonify, render_template
app = Flask(__name__)
users = [                                                    # Teszt adatok
   {"id": 1, "name": "Anna", "email": "anna@example.com"},
   {"id": 2, "name": "Béla", "email": "bela@example.com"},
   {"id": 3, "name": "Cecil", "email": "cecil@example.com"}
]
@app.route("/api/users")                                     # JSON API végpont
def get_users():
   return jsonify(users)
 @app.route("/users")                                        # HTML oldal megjelenítése
def user_list():
   return render_template("users.html", users=users)
 @app.route("/")                                             # Főoldal
def home():
   return "<h1>Üdv a Flask alkalmazásban!</h1><p>Menj a /users vagy /api/users oldalra.</p>"
if __name__ == "__main__":
   app.run(debug=True)
```

templates/users.html – Jinja2 sablon:

```
<!DOCTYPE html>
<html lang="hu">
<head>
  <meta charset="UTF-8">
  <title>Felhasználók listája</title>
</head>
<body>
  <h1>Felhasználók</h1>
  <ul>
    {% for user in users %}
      <li>{{ user.name }} – {{ user.email }}</li>
    {% endfor %}
  </ul>
</body>
</html>
```

Az alkalmazás futtatása:

1. Egy virtuális környezet létrehozása (opcionális, de ajánlott):

   python -m venv venv

   venv\Scripts\activate

2. Az alkalmazás futtatása terminálból: **python app.py**
3. Megnyitás böngészőben:
   - http://127.0.0.1:5000 – főoldal
   - http://127.0.0.1:5000/users – HTML oldal
   - http://127.0.0.1:5000/api/users – JSON API végpont

3. feladat: Egy nyilvántartó alkalmazás létrehozása Flask-ben:

- Oktatókat és kurzusokat kezel
- Adatokat egy data.json fájlban tárol
- Jinja2 sablonokat használ dinamikus HTML oldalakhoz

Adatmodellek:

- **Kurzus** (név, leírás)
- **Oktató** (név, e-mail)
- **Hallgató** (név, e-mail)
- **Kurzus-Hallgató kapcsolat** (melyik hallgató melyik kurzuson van)

Funkciók:

- Új kurzus/oktató/hallgató felvétele
- Módosítási lehetőség minden entitáshoz
- Törlés
- Hallgatók hozzárendelése kurzusokhoz
- Szűrés: például adott hallgató kurzusainak listázása

Projektstruktúra:

```
course_app/
├── app.py
├── data.json
├── templates/
│   ├── layout.html
│   ├── index.html
│   ├── courses.html
│   ├── add_course.html
│   └── add_instructor.html
```

Összetevők:

- **Flask** – szerveroldali keretrendszer
- **Flask-WTF** – űrlapkezeléshez
- **Jinja2** – HTML oldalak rendereléséhez
- **JSON** adatok tárolása
- **Bootstrap** – a megjelenítés stílusához

data.json – kezdeti adatstruktúra

```
{
  "instructors": [],
  "courses": []
}
```

Jinja2 a HTML oldalba illeszti be a Pythonból jövő adatokat:

```
<!-- courses.html -->
<h1>Kurzusok listája</h1>
<ul>
 {% for course in courses %}
   <li>{{ course.name }} - {{ course.instructor.name }}</li>
 {% endfor %}
</ul>
```

A courses változót a Flask küldi a sablonnak:

```
@app.route("/courses")
def course_list():
    courses = Course.query.all()
    return render_template("courses.html", courses=courses)
```

Flask alkalmazás: app-py

```
from flask import Flask, render_template, request, redirect, url_for
import json
import os

app = Flask(__name__)
DATA_FILE = "data.json"

def load_data():
    if not os.path.exists(DATA_FILE):
        with open(DATA_FILE, "w") as f:
            json.dump({"instructors": [], "courses": []}, f)
    with open(DATA_FILE, "r") as f:
        return json.load(f)

def save_data(data):
    with open(DATA_FILE, "w") as f:
        json.dump(data, f, indent=4)

@app.route("/")
def index():
    return render_template("index.html")
```

- app létrehozása
- JSON fájl létrehozása
- Fájlkezelő metódusok
- Kezdőlap renderelése

app.py folytatása

```
@app.route("/courses")
def courses():
    data = load_data()
    return render_template("courses.html", courses=data["courses"])

@app.route("/add-course", methods=["GET", "POST"])
def add_course():
    data = load_data()
    if request.method == "POST":
        name = request.form["name"]
        description = request.form["description"]
        instructor_id = int(request.form["instructor"])
        instructor = next((i for i in data["instructors"] if i["id"] == instructor_id), None)
        new_course = {
            "id": len(data["courses"]) + 1,
            "name": name,
            "description": description,
            "instructor": instructor
        }
        data["courses"].append(new_course)
        save_data(data)
        return redirect(url_for("courses"))
    return render_template("add_course.html", instructors=data["instructors"])
```

- **Kurzusok listázása**

- **Kurzus hozzáadása:**
  - attribútumok bekérése: request.form-mal
  - new_course dict: létrehozás, hozzáfűzés, mentés
- redirect és url_for: **böngésző átirányítása** a kurzusok listázására

- Sablon renderelése, oktatók listájának átadása

app.py folytatása

```
@app.route("/add-instructor", methods=["GET", "POST"])
def add_instructor():
    data = load_data()
    if request.method == "POST":
        name = request.form["name"]
        email = request.form["email"]
        new_instructor = {
            "id": len(data["instructors"]) + 1,
            "name": name,
            "email": email
        }
        data["instructors"].append(new_instructor)
        save_data(data)
        return redirect(url_for("index"))
    return render_template("add_instructor.html")

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

- **Oktató hozzáadása**
  - attribútumok bekérése:
    request.form-mal
  - new_instructor dict:
    létrehozás,
    hozzáfűzés,
    mentés
- Redirect, url_for: **böngésző átirányítása** az alapoldalra
- Sablon renderelése, oktatók listájának átadása
- **Futtatás**

Sablonfájlok:

layout.html: a főmenü oldala

```
<!DOCTYPE html>
<html lang="hu">
<head>
  <meta charset="UTF-8">
  <title>Kurzus Nyilvántartó</title>
</head>
<body>
  <h1>Kurzus Nyilvántartó</h1>
  <nav>
    <a href="{{ url_for('index') }}">Főoldal</a> |
    <a href="{{ url_for('courses') }}">Kurzusok</a> |
    <a href="{{ url_for('add_course') }}">Új kurzus</a> |
    <a href="{{ url_for('add_instructor') }}">Új oktató</a>
  </nav>
  <hr>
  {% block content %}{% endblock %}
</body>
</html>
```

- nav bar: információs sáv a menüpontok linkjeivel
- A content blockba kerülnek kitöltésre az egyes menüpontok html kódjai a kiterjesztések szerint.

index.html

```
{% extends "layout.html" %}
{% block content %}
<p>Üdvözöllek a Kurzus Nyilvántartó alkalmazásban!</p>
{% endblock %}
```

courses.html: kurzusok listázása

```
{% extends "layout.html" %}
{% block content %}
<h2>Kurzusok listája</h2>
<ul>
  {% for course in courses %}
    <li><strong>{{ course.name }}</strong> – {{ course.description }}
(Oktató: {{ course.instructor.name }})</li>
  {% else %}
    <li>Nincs elérhető kurzus.</li>
  {% endfor %}
</ul>
{% endblock %}
```

- Alap a layout.html oldal
- Ez a html kerül a content blokk helyére a layout-ban

add_course.html: űrlap (form) megjelenítése

```
{% extends "layout.html" %}
{% block content %}
<h2>Új kurzus felvétele</h2>
<form method="post">
  <label>Kurzus neve:</label><br>
  <input type="text" name="name" required><br>
  <label>Leírás:</label><br>
  <input type="text" name="description"><br>
  <label>Oktató:</label><br>
  <select name="instructor" required>
    {% for instructor in instructors %}
      <option value="{{ instructor.id }}">{{ instructor.name }}</option>
    {% endfor %}
  </select><br><br>
  <button type="submit">Mentés</button>
</form>
{% endblock %}
```

- Alap a layout.html oldal
- Ez a html kerül a content blokk helyére a layout-ban
- Az űrlap POST-ot használ
- 2 szövegmező megjelenítése
- Kötelező kitöltés: required
- Legördülő menü: select
- "submit" gomb: az űrlap elküldése

add_instructor.html: űrlap megjelenítése

```
{% extends "layout.html" %}
{% block content %}
<h2>Új oktató felvétele</h2>
<form method="post">
  <label>Név:</label><br>
  <input type="text" name="name" required><br>
  <label>Email:</label><br>
  <input type="email" name="email"><br><br>
  <button type="submit">Mentés</button>
</form>
{% endblock %}
```

- Alap a layout.html oldal
- Ez a html kerül a content blokk helyére a layout-ban
- Az űrlap POST-ot használ
- 1 szövegmező megjelenítése, kötelező kitöltés: required
- 1 email mező megjelenítése
- "submit" gomb: az űrlap elküldése

Indítás:

```
cd course_app

python app.py
```

Lépésenként kibővítve az alkalmazást:

1. Hallgatók hozzáadása
2. Hallgatók hozzárendelése kurzusokhoz
3. Szűrés: egy hallgató kurzusainak megjelenítése
4. Törlés és módosítás lehetőségek oktatóra, kurzusra, hallgatóra

Frissített projektstruktúra:

```
course_app/
├── app.py
├── data.json
├── templates/
│   ├── layout.html
│   ├── index.html
│   ├── courses.html
│   ├── add_course.html
│   ├── add_instructor.html
│   ├── instructors.html
│   ├── students.html
│   ├── add_student.html
│   ├── enroll_student.html
│   └── student_courses.html
```

Hallgatók hozzáadása

1. Bővítjük a data.json fájlt a hallgatókkal és kapcsolatokkal:

```
{
  "instructors": [],
  "courses": [],
  "students": [],
  "enrollments": []
}
```

2. app.py – új route-ok hozzáadása:

```
@app.route("/students")
def students():
    data = load_data()
    return render_template("students.html",
students=data["students"])

@app.route("/add-student", methods=["GET", "POST"])
def add_student():
    data = load_data()
    if request.method == "POST":
        name = request.form["name"]
        email = request.form["email"]
        new_student = {
            "id": len(data["students"]) + 1,
            "name": name,
            "email": email
        }
        data["students"].append(new_student)
        save_data(data)
        return redirect(url_for("students"))
    return render_template("add_student.html")
```

3. Menüfrissítés (layout.html)

A hallgatói oldalt egészítsük ki ezzel, hogy a hallgatók elérhetőek legyenek a menüből:

```
<a href="{{ url_for('students') }}">Hallgatók</a> |
<a href="{{ url_for('add_student') }}">Új hallgató</a> |
```

4. A students.html sablon létrehozása

templates/students.html

```
{% extends "layout.html" %}
{% block content %}
<h2>Hallgatók listája</h2>
<ul>
  {% for student in students %}
    <li>{{ student.name }} – {{ student.email }}</li>
  {% else %}
    <li>Nincs elérhető hallgató.</li>
  {% endfor %}
</ul>
{% endblock %}
```

5. add_student.html sablon létrehozása

templates/add_student.html

```
{% extends "layout.html" %}
{% block content %}
<h2>Új hallgató felvétele</h2>
<form method="post">
  <label>Név:</label><br>
  <input type="text" name="name" required><br>
  <label>Email:</label><br>
  <input type="email" name="email" required><br><br>
  <button type="submit">Mentés</button>
</form>
{% endblock %}
```

Tesztelés

1. Futtassuk az appot: python app.py
2. Nyissuk meg:
   - http://127.0.0.1:5000/students – hallgatók listája
   - http://127.0.0.1:5000/add-student – új hallgató felvétele

A **Bootstrap** egy nyílt forráskódú **front-end keretrendszer**, amelyet a weboldalak és webalkalmazások gyors és egyszerű fejlesztésére használnak. A Bootstrap HTML, CSS és JavaScript alapú eszközöket biztosít.

Fő jellemzők:

1. **Reszponzív dizájn**: A Bootstrap beépített rácsrendszert biztosít, amely lehetővé teszi, hogy a weboldalak különböző képernyőméretekhez igazodjanak.
2. **Előre definiált stílusok**: Tartalmaz előre elkészített CSS osztályokat gombokhoz, űrlapokhoz, navigációs sávokhoz, táblázatokhoz, kártyákhoz és más UI-elemekhez.
3. **Komponensek**: Számos újrahasználható UI-komponenst kínál, pl. modális ablakokat, legördülő menüket, figyelmeztetéseket, jelvényeket.
4. **JavaScript funkciók**: Tartalmaz JavaScript alapú interaktív elemeket, pl. diavetítéseket (carousel), legördülő menüket és modális ablakokat.
5. **Egyszerű használat**: Könnyen integrálható bármilyen webalkalmazásba, és gyorsan testreszabható a saját igények szerint.

A Bootsrap használata a Flask alkalmazásban:

A Bootstrap integrálható a HTML sablonokba. Ehhez a Bootstrap CSS és JavaScript fájljait kell hozzáadni a sablonokhoz.

Példa: HTML sablon Bootstrap integrációval:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Flask + Bootstrap</title>
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <div class="container">
    <h1 class="text-center mt-5">Üdvözöllek a Flask alkalmazásban!</h1>
    <button class="btn btn-primary">Kattints ide</button>
  </div>
  <!-- Bootstrap JS -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Köszönöm a figyelmet!

Konzultáció

Minden héten pénteken 18:00 – 20:00