

FLASK

Flask, Python dilinde yazılmış web uygulamaları geliştirmek için kullanılan bir **web framework**'üdür.

Web Framework Nedir?

Bir Web Framework'ü, web uygulaması geliştiricilerinin protokol, iş parçacığı yönetimi vb. gibi düşük düzeyli ayrıntılar hakkında endişelenmeden uygulamalar yazmasına olanak tanıyan bir kitaplıklar ve modüller koleksiyonunu temsil eder.

Flask, dinamik içerikli web siteleri oluşturmak için tasarlanmıştır. Basitliği ve esnekliği ile popüler bir seçenekdir. Açık kaynak kodlu bir **micro framework'dür**.

Micro Framework Nedir?

Flask'a genellikle mikro framework denir. Uygulamanın çekirdeğini basit ve ölçülebilir tutmak için tasarlanmıştır.

Veritabanı desteği için bir soyutlama katmanı (abstraction layer) yerine Flask, uygulamaya bu tür yetenekleri eklemek için uzantıları (extensions) destekler.

Micro kavramını için **basit ama genişletilebilir** denilebilir. Şu şekilde açıklanabilir:

Kendinize bir araba oluşturduğunuzu varsayıñ. Motor, kapılar, tampon, bagaj gibi birçok parça ihtiyacınız var. Bu motorun parçalarını almak için iki farklı yere gidiyorsunuz. İlk yer ihtiyacınız olan parçaları ve daha fazlasını paket halinde vermek istiyor. İkinci yer ise sadece motoru veriyor ve dilerseniz eklemeler yaparsınız diyor. İşte gittiğimiz ikinci yer micro framework'ün ta kendisi. İstediğiniz parçayı kullanma veya kullanmama özgürlüğü tanıyan özünde basit bir yapıdır.

Bu parçalara bilgisayar dünyasından örnek vermek gerekirse; veri tabanı, kimlik doğrulama, form doğrulama gibi sistemlerdir. Bu sistemlere, ihtiyaca göre ekleme yapabilme özgürlüğü vardır.

Flask Kullanan Global Şirketler:



Flask, **Werkzeug WSGI** araç setini ve **Jinja2** şablonu temel alır.

WSGI

WSGI (Web Server Gateway Interface), Python ile web uygulamaları geliştirmek için kullanılan bir standarttır. Bu, web sunucuları ile web uygulamaları arasında iletişim kurmak için bir yol sağlar.

Werkzeug

Werkzeug, istekleri, yanıtları ve yardımcı işlevleri yöneten bir araç setidir (toolkit). Bu araç seti, bir web framework oluşturmak için kullanılabilir. Flask frameworkü, temelini oluştururken Werkzeug'u kullanır.

jinja2

Jinja2, Python için yaygın olarak kullanılan bir şablon aracıdır. Bu araç, web sayfalarını oluşturmak için kullanılan şablonları ve verileri birleştirerek dinamik web sayfaları oluşturmaya yardımcı olur.

Bu, [Python değişkenlerini](#) aşağıdaki gibi HTML şablonlarına aktarmanıza olanak tanır:

```
<html>
    <head>
        <title> {{ title }} </title>
    </head>
    <body>
        <h1>Hello {{ username }} </h1>
    </body>
</html>
```

Neden Flask?

Flask, Python programlama dilini kullanarak web uygulamaları oluşturma işlemini çok kolaylaştırır. Önemli özellikleri arasında şunlar yer alır:

- ★ Basit yapısı ile öğrenmesi ve kullanması kolaydır.
- ★ Minimum gereksinimleri vardır.
- ★ Esnek ve genişletilebilirdir.
- ★ Ölçeklenebilir ve güvenilirdir.

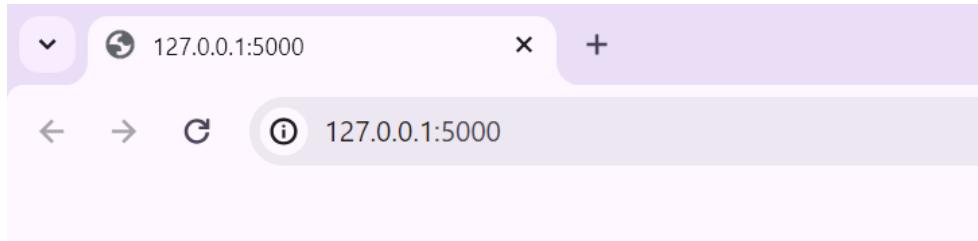
Bunun üzerine, çok açık bir yapıya sahiptir, bu da okunabilirliği artırır. "Hello World" uygulamasını oluşturmak için sadece birkaç satır kod gereklidir.

İşte bir örnek temel kod yapısı:

```
app.py > ...
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route('/')
5  def hello_world():
6      return 'Hello World!'
7
8  if __name__ == '__main__':
9      app.run()
10
11
12
```

```
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [06/Mar/2024 16:27:10] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [06/Mar/2024 16:27:10] "GET /favicon.ico HTTP/1.1" 404 -
```

Ardından, yalnızca kendi bilgisayarınızda erişilebilir olan bir web sunucusu başlatır. Bir web tarayıcısında 5000 numaralı bağlantı noktasında localhost'u açarsanız (URL), "Hello World!"'ü görürüz.



Hello World!

Kodu inceleyelim: Bu kod ne yapıyor?

1-İlk olarak, Flask kütüphanesinden Flask sınıfını içeri aktarıyoruz. Bu sınıf, uygulamamızın temelini oluşturacak.

2-Sonra, Flask sınıfından bir örnek oluşturuyoruz. Bu örneği oluştururken, uygulamamızın adını (name) veriyoruz. Bu, Flask'in şablonlar, statik dosyalar vb. için nereye bakacağını belirtmemize yardımcı olur.

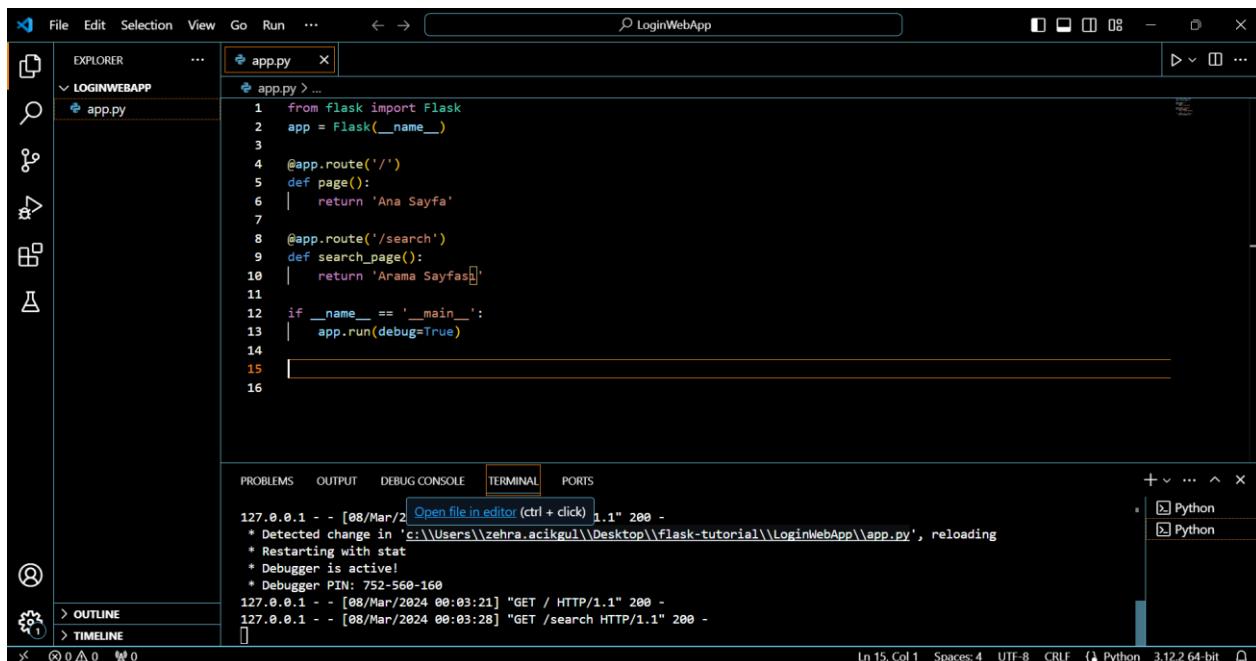
3-@app.route('/') ifadesi ile '/' yolunu belirtiyoruz. Bu, tarayıcıda ana URL olarak görünecek olan yolu belirtir.

4-Son olarak, hello_world() adında bir fonksiyon tanımlıyoruz. Bu fonksiyon, kullanıcının tarayıcısında Hello, World!' metnini görüntüleyecek.

Flask Temel İşlemler

1- Flask URL Yapısı ve Dinamik URL

URL yapısı şu şekildedir:



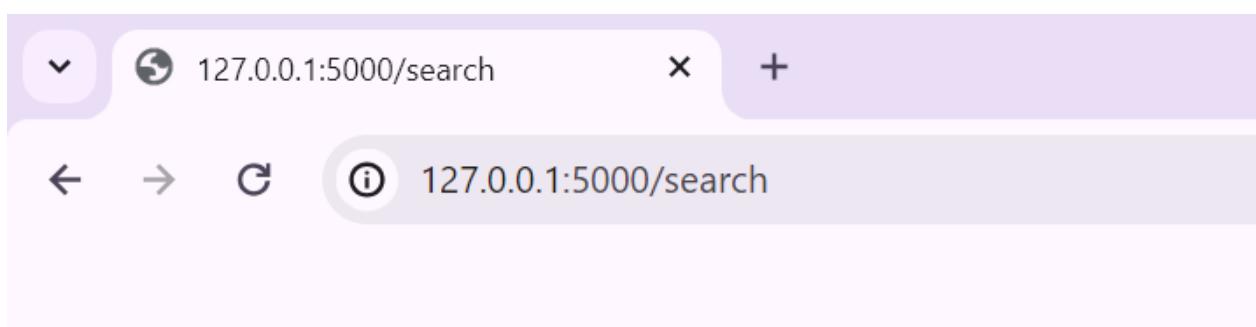
The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left has a 'LOGINWEBAPP' folder expanded, containing an 'app.py' file. The code editor window displays the following Python code:

```
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route('/')
5  def page():
6      return 'Ana Sayfa'
7
8  @app.route('/search')
9  def search_page():
10     return 'Arama Sayfası'
11
12 if __name__ == '__main__':
13     app.run(debug=True)
14
15
16
```

The terminal tab at the bottom shows the application's log output:

```
127.0.0.1 - - [08/Mar/2024 00:03:21] "GET / HTTP/1.1" 200 -
* Detected change in 'c:\Users\zehra.acikgul\Desktop\flask-tutorial\LoginWebApp\app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
127.0.0.1 - - [08/Mar/2024 00:03:21] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Mar/2024 00:03:28] "GET /search HTTP/1.1" 200 -
```

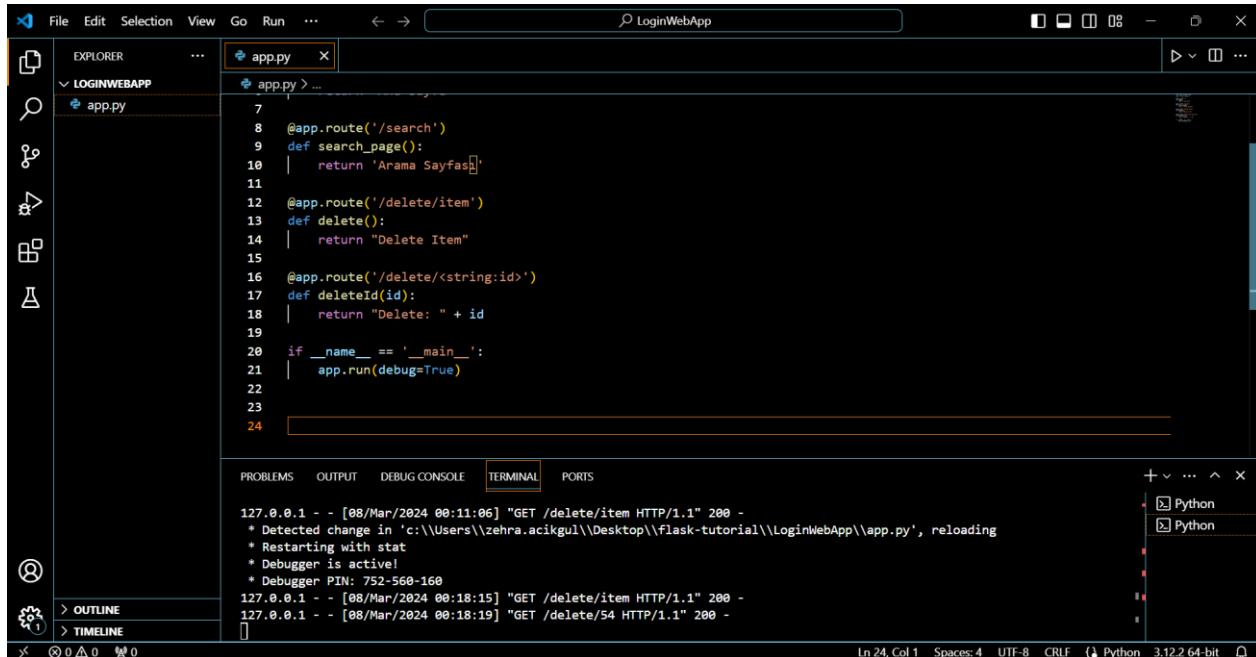
Arama yerine <http://127.0.0.1:5000/search> yazdığımız zaman karşımıza çıkan sayfa:



Arama Sayfası

Dinamik URL:

Bir sayfamız olduğunu düşünelim ve bu sayfada yüzlerce, binlerce makalemiz olsun. delete/2, delete/54, delete/100... şeklinde her makale için route yazmak mantıklı mı? Tabi ki değil. Dinamik URL burda yardımımıza yetişir:



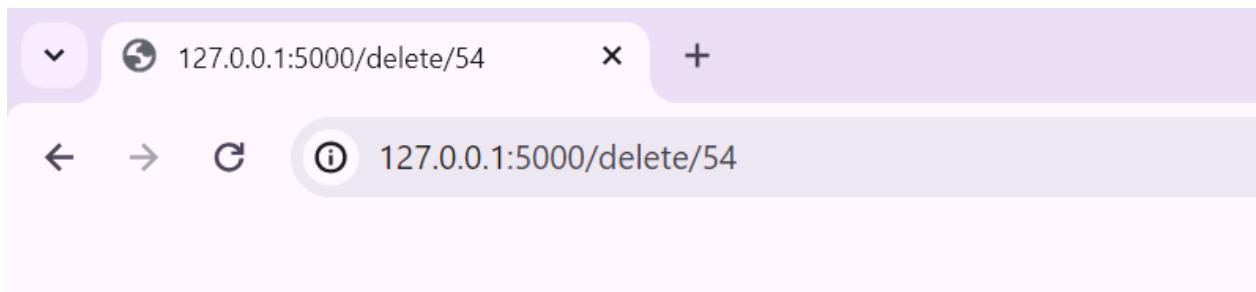
The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder named "LOGINWEBAPP" containing "app.py".
- Code Editor:** Displays the "app.py" file content:

```
7
8     @app.route('/search')
9     def search_page():
10        |         return 'Arama Sayfası'
11
12    @app.route('/delete/item')
13    def delete():
14        |         return "Delete Item"
15
16    @app.route('/delete/<string:id>')
17    def deleteid(id):
18        |         return "Delete: " + id
19
20    if __name__ == '__main__':
21        |         app.run(debug=True)
22
23
24
```
- Terminal:** Shows log output from the application:

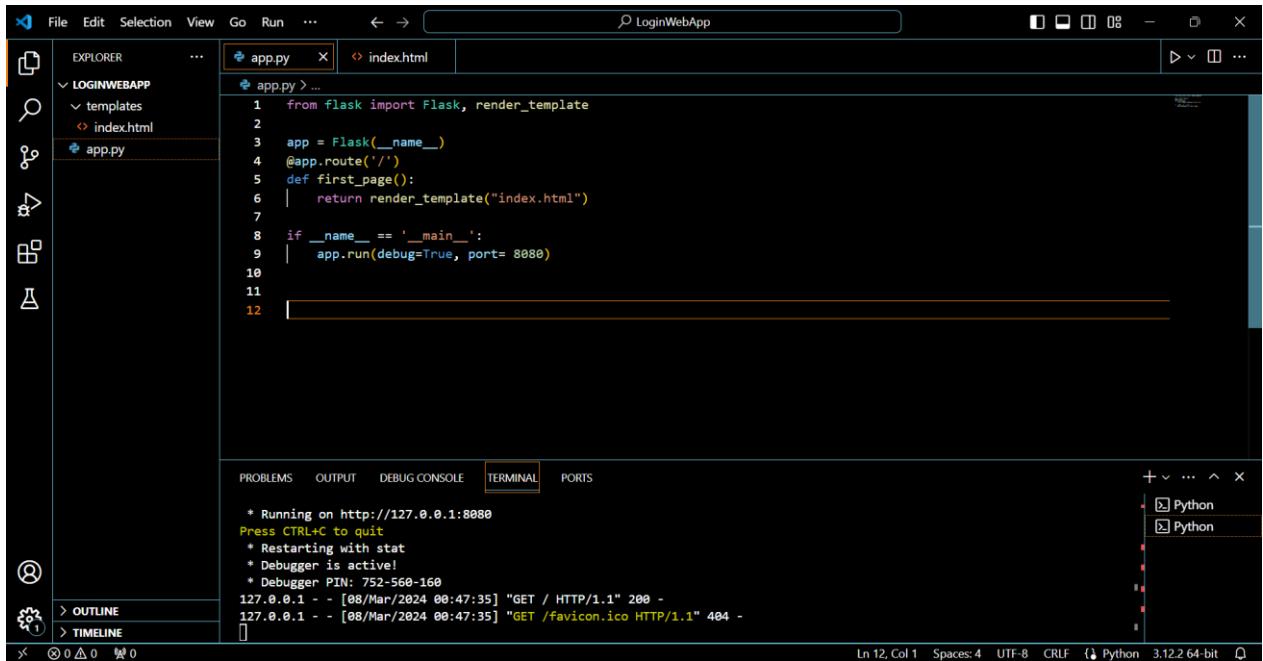
```
127.0.0.1 - - [08/Mar/2024 00:11:06] "GET /delete/item HTTP/1.1" 200 -
* Detected change in 'c:\\Users\\zehra.acikgu\\Desktop\\flask-tutorial\\LoginWebApp\\app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
127.0.0.1 - - [08/Mar/2024 00:18:15] "GET /delete/item HTTP/1.1" 200 -
127.0.0.1 - - [08/Mar/2024 00:18:19] "GET /delete/54 HTTP/1.1" 200 -
```
- Status Bar:** Shows "Ln 24, Col 1" and "Python 3.12.2 64-bit".

id nin 54 olduğu düşünelim:



Delete: 54

2- Jinja Templateler

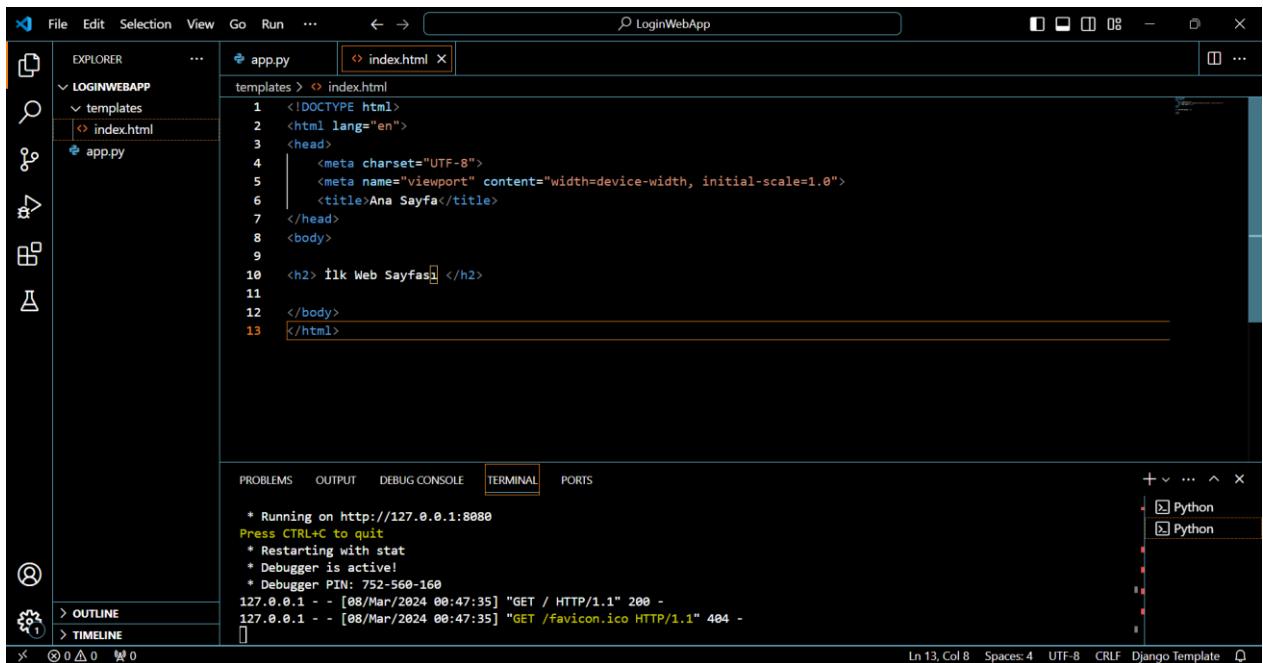


The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a project structure for "LOGINWEBAPP" with "templates" and "index.html".
- Code Editor:** Displays the "app.py" file containing Python code for a Flask application.
- Terminal:** Shows the application running on port 8080, displaying the message "Ana Sayfa".
- Status Bar:** Shows "Ln 12, Col 1" and "Python 3.12.2 64-bit".

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route('/')
def first_page():
    return render_template("index.html")
if __name__ == '__main__':
    app.run(debug=True, port= 8080)
```

```
* Running on http://127.0.0.1:8080
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
127.0.0.1 - - [08/Mar/2024 00:47:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Mar/2024 00:47:35] "GET /favicon.ico HTTP/1.1" 404 -
```



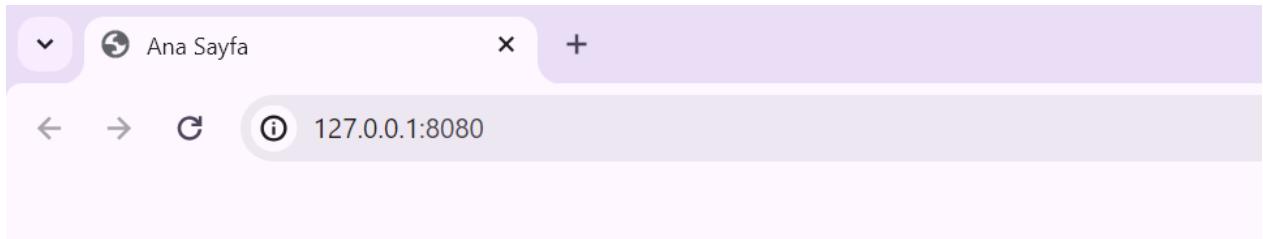
The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a project structure for "LOGINWEBAPP" with "templates" and "index.html".
- Code Editor:** Displays the "index.html" file containing a basic HTML structure with a title and an h2 tag.
- Terminal:** Shows the application running on port 8080, displaying the message "Ana Sayfa".
- Status Bar:** Shows "Ln 13, Col 8" and "Django Template".

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ana Sayfa</title>
</head>
<body>
    <h2> İlk Web Sayfası </h2>
</body>
</html>
```

```
* Running on http://127.0.0.1:8080
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
127.0.0.1 - - [08/Mar/2024 00:47:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Mar/2024 00:47:35] "GET /favicon.ico HTTP/1.1" 404 -
```

Respond şu şekilde olur:



İlk Web Sayfası

HTML sayfamıza veri de gönderebiliriz. Python verisi nasıl gönderilir? index.html sayfamıza iki tane number göndermek istiyoruz ve number değişkenin değerlerinin 10 ve 20 olduğunu kabul edelim:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project named "LOGINWEBAPP" with a "templates" folder containing "index.html" and an "app.py" file.
- Code Editor:** Displays the "app.py" file content:

```
1  from flask import Flask, render_template
2
3  app = Flask(__name__)
4  @app.route('/')
5  def first_page():
6      return render_template("index.html", number1 = 10, number2 = 20)
7
8  if __name__ == '__main__':
9      app.run(debug=True, port= 8080)
```
- Terminal:** Shows log messages indicating the application is restarting and a debugger is active.

```
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
* Detected change in 'c:\\\\Users\\\\zehra.acikgul\\\\Desktop\\\\flask-tutorial\\\\LoginWebApp\\\\app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
```
- Status Bar:** Shows the current line (Ln 12, Col 1), spaces (Spaces: 4), encoding (UTF-8), and file type (Python).

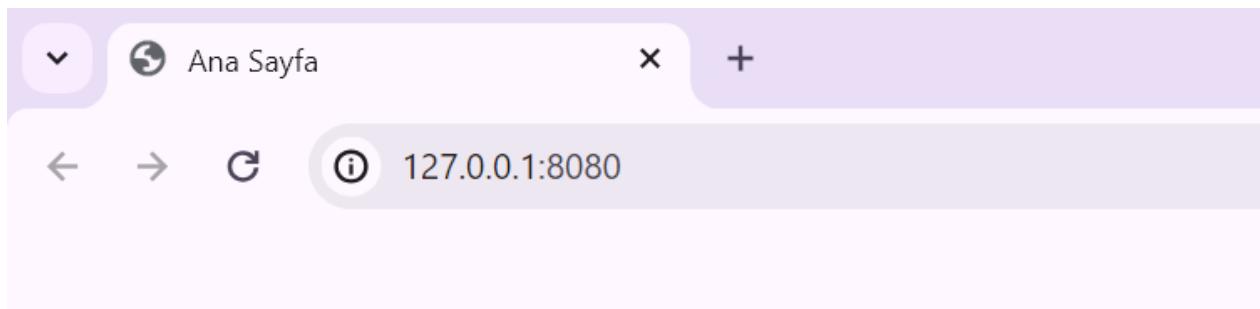
The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Title Bar:** LoginWebApp
- Explorer:** Shows a project structure for "LOGINWEBAPP" with "templates" and "index.html".
- Editor:** Displays "index.html" content:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ana Sayfa</title>
</head>
<body>
    <h2> İlk Web Sayfası </h2>
    1.sayı: {{number1}} <br>
    2.sayı: {{number2}}
</body>
</html>
```
- Terminal:** Shows logs from a debugger:

```
* Debugger PIN: 752-560-160
127.0.0.1 - - [08/Mar/2024 00:47:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Mar/2024 00:47:35] "GET /favicon.ico HTTP/1.1" 404 -
* Detected change in 'c:\\Users\\zehra.acikgu1\\Desktop\\flask-tutorial\\LoginWebApp\\app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
```
- Status Bar:** Line 16, Col 8, Spaces: 4, UTF-8, CRLF, Django Template

Respondumuz şu şekilde olur:



1.sayı: 10

2.sayı: 20

3- Templatelerde Koşullu Durum ve Döngüler

The image displays two side-by-side code editors in the Visual Studio Code interface, both titled "LoginWebApp".

Left Editor (app.py):

```
File Edit Selection View Go Run ... ← → LoginWebApp
EXPLORER ... app.py x index.html
LOGINWEBAPP
templates
index.html
app.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ v ... ^ x
Python
Python
Ln 12, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.2 64-bit Q
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
* Detected change in 'c:\\Users\\zehra.acikgul\\Desktop\\flask-tutorial\\LoginWebApp\\app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
Ln 12, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.2 64-bit Q
```

Right Editor (index.html):

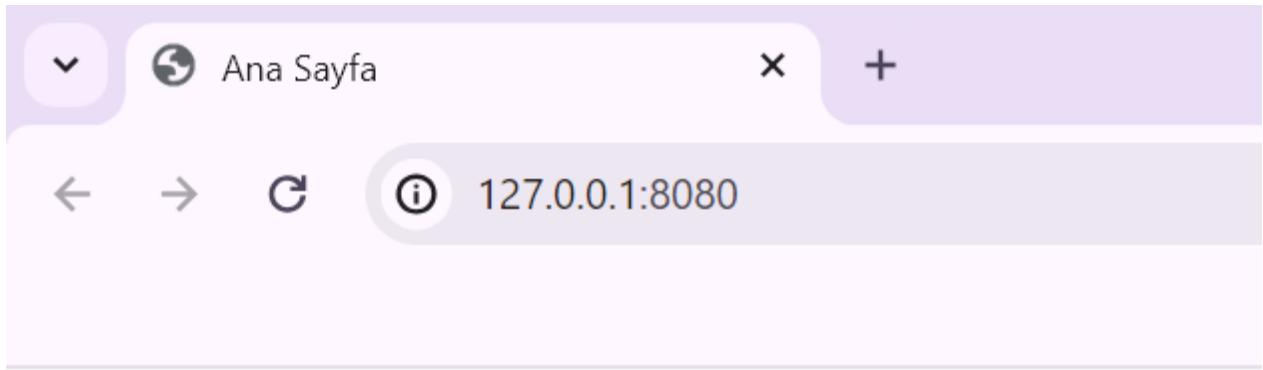
```
File Edit Selection View Go Run ... ← → LoginWebApp
EXPLORER ... app.py x index.html
LOGINWEBAPP
templates > index.html
index.html
app.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ v ... ^ x
Python
Python
Ln 16, Col 1 Spaces: 4 UTF-8 CRLF Django Template Python 3.12.2 64-bit Q
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ana Sayfa</title>
</head>
<body>
    {% if message %}
        <p> {{message}} </p>
    {% else %}
        <p> Mesaj gönderilemedi... </p>
    {% endif %}
</body>
</html>
Ln 16, Col 1 Spaces: 4 UTF-8 CRLF Django Template Python 3.12.2 64-bit Q
```

The left editor contains the following Python code:

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route('/')
def page():
    message = "Bu bir mesajdır..."
    return render_template("index.html", message=message)
if __name__ == '__main__':
    app.run(debug=True, port= 8080)
```

The right editor contains the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ana Sayfa</title>
</head>
<body>
    {% if message %}
        <p> {{message}} </p>
    {% else %}
        <p> Mesaj gönderilemedi... </p>
    {% endif %}
</body>
</html>
```



Bu bir mesajdır...

FOR Döngüsü:

The screenshot shows a code editor interface with the following details:

- File Structure:** The left sidebar shows a project named "LOGINWEBAPP" containing "templates" and "app.py".
- Code Editor:** The main pane displays the "app.py" file content:

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route('/')
def page():
    numbers=[1,2,3,4,5]
    return render_template("index.html", numbers=numbers)
if __name__ == '__main__':
    app.run(debug=True, port= 8080)
```
- Terminal:** The bottom pane shows the terminal output of the Flask application:

```
* Debugger is active!
* Debugger PIN: 752-560-160
* Detected change in 'c:\\\\Users\\\\zehra.acikgul\\\\Desktop\\\\flask-tutorial\\\\LoginWebApp\\\\app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
127.0.0.1 - - [08/Mar/2024 01:15:29] "GET / HTTP/1.1" 200 -
```
- Status Bar:** The bottom right shows status information: Ln 13, Col 1, Spaces: 4, UTF-8, CRLF, Python 3.12.2 64-bit.

The screenshot shows the Visual Studio Code interface with the following details:

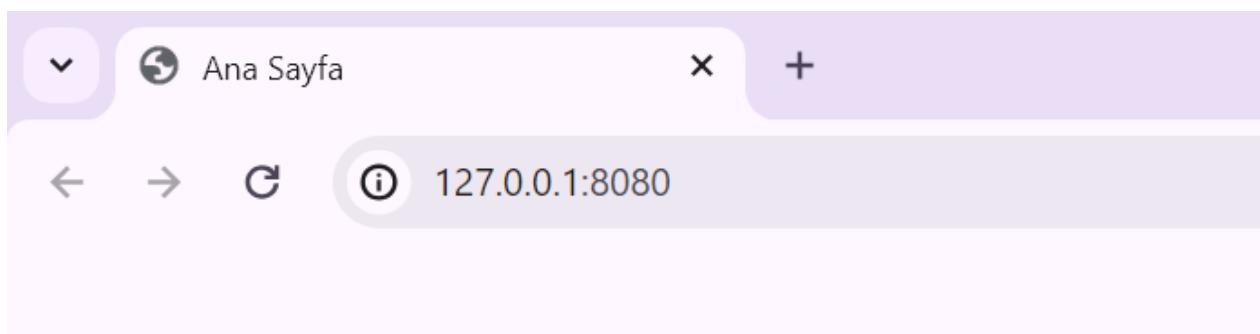
- File Explorer:** Shows a project structure for "LOGINWEBAPP" with "templates" and "index.html".
- Editor:** Displays the content of "index.html":

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ana Sayfa</title>
</head>
<body>
<ul>
    {% for number in numbers %}
        <li>{{number}}</li>
    {% endfor %}
</ul>
</body>
</html>
```

- Terminal:** Shows logs indicating a debugger is active and a file is being reloaded.

```
* Debugger is active!
* Debugger PIN: 752-560-160
* Detected change in 'c:\\\\Users\\\\zehra.acikgul\\\\Desktop\\\\flask-tutorial\\\\LoginWebApp\\\\app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
127.0.0.1 - - [08/Mar/2024 01:15:29] "GET / HTTP/1.1" 200 -
```

respond:



- 1
- 2
- 3
- 4
- 5

4- GET ve POST Request, Form İşlemleri

The screenshot shows the VS Code interface with the title bar "LoginWebApp". The Explorer sidebar on the left shows a project structure under "LOGINWEBAPP" with files "index.html", "number.html", and "app.py". The "app.py" tab is active, displaying the following Python code:

```
from flask import Flask, render_template, request
app = Flask(__name__)
@app.route('/')
def page():
    numbers=[1,2,3,4,5]
    return render_template("index.html")
@app.route('/toplaml', methods=["GET","POST"])
def toplam():
    if request.method == "POST":
        number1 = request.form.get("number1")
        number2 = request.form.get("number2")
        return render_template("number.html", total = int(number1) + int(number2))
    else:
        return render_template("number.html")
```

The "OUTPUT" tab at the bottom shows logs from the terminal:

```
127.0.0.1 - - [08/Mar/2024 02:09:11] "GET /toplaml?_debugger_=yes&cmd=resource&f=console.png HTTP/1.1" 304 -
127.0.0.1 - - [08/Mar/2024 02:09:11] "GET /toplaml?_debugger_=yes&cmd=resource&f=console.png HTTP/1.1" 304 -
* Detected change in 'c:\\Users\\zehra.acikgul\\Desktop\\flask-tutorial\\LoginWebApp\\app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
127.0.0.1 - - [08/Mar/2024 02:10:42] "POST /toplaml HTTP/1.1" 200 -
```

At the bottom right, status indicators show "Ln 19, Col 1" and "Python 3.12.2 64-bit".

The screenshot shows the VS Code interface with the title bar "LoginWebApp". The Explorer sidebar on the left shows a project structure under "LOGINWEBAPP" with files "index.html", "number.html", and "app.py". The "index.html" tab is active, displaying the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ana Sayfa</title>
</head>
<body>
    <form action="/toplaml" method=post>
        <input type="number" name="number1"> <br>
        <input type="number" name="number2"> <br>
        <button type="submit">Gönder</button>
    </form>
</body>
</html>
```

The "OUTPUT" tab at the bottom shows logs from the terminal:

```
127.0.0.1 - - [08/Mar/2024 02:09:11] "GET /toplaml?_debugger_=yes&cmd=resource&f=console.png HTTP/1.1" 304 -
127.0.0.1 - - [08/Mar/2024 02:09:11] "GET /toplaml?_debugger_=yes&cmd=resource&f=console.png HTTP/1.1" 304 -
* Detected change in 'c:\\Users\\zehra.acikgul\\Desktop\\flask-tutorial\\LoginWebApp\\app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
127.0.0.1 - - [08/Mar/2024 02:10:42] "POST /toplaml HTTP/1.1" 200 -
```

At the bottom right, status indicators show "Ln 15, Col 8" and "Django Template".

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Toplam Sayfası</title>
</head>
<body>
    {% if total %}<p>Toplamlar: {{total}}</p>
    {% else %}<p> Bu bir get request olduğu için bir değer yoktur! </p>
    {% endif %}
</body>
</html>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
127.0.0.1 - - [08/Mar/2024 02:09:11] "GET /toplamsayfasi HTTP/1.1" 304 -
127.0.0.1 - - [08/Mar/2024 02:09:11] "GET /toplamsayfasi?_yes&cmd=resource&f=console.png HTTP/1.1" 304 -
* Detected change in 'c:\\Users\\zehra.acikgul\\Desktop\\flask-tutorial\\LoginWebApp\\app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 752-560-160
127.0.0.1 - - [08/Mar/2024 02:10:42] "POST /toplamsayfasi HTTP/1.1" 200 -
```

Ana Sayfa

127.0.0.1:8080

12

23

Gönder

Toplam Sayfası

127.0.0.1:8080/toplamsayfasi

Toplamları: 35

5- Redirect – URL Yönlendirme

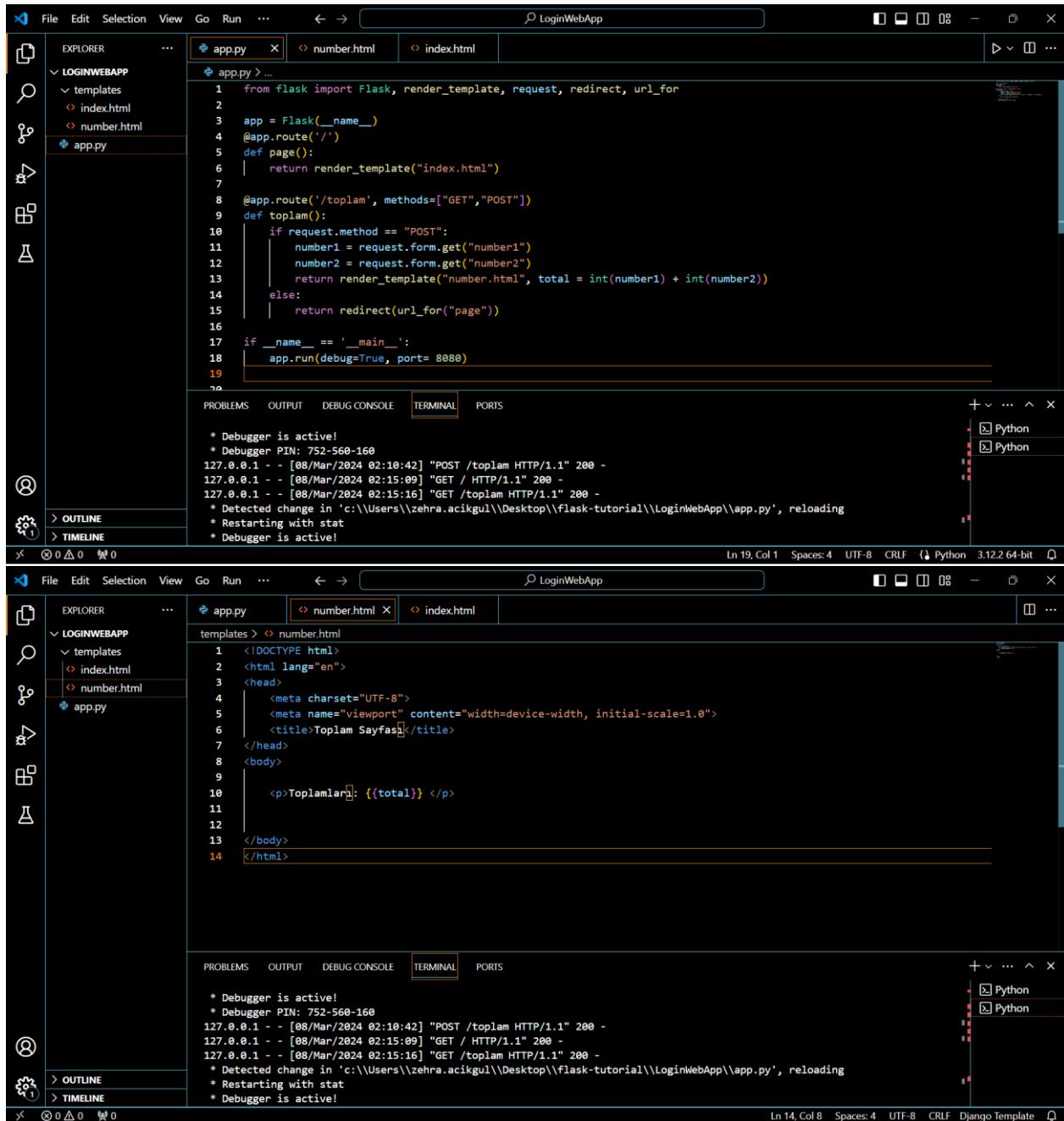
<http://127.0.0.1:8080/toplamsayfasi> bu address gittiğimizi yani bir get request yaptığımızı düşünelim.

Toplam Sayfası

127.0.0.1:8080/toplamsayfasi

Bu bir get request olduğu için bir değer yoktur!

Böyle bir uyarı vermek yerine kullanıcıyı ana sayfaya yönlendirmek istiyoruz. O zaman:



The screenshot shows two code editors side-by-side in VS Code. Both editors have the title bar "LoginWebApp".

Editor 1 (Top): Contains the file `app.py`. The code defines a Flask application with routes for the home page and a sum calculation endpoint. It uses `request.form.get` to get values from a POST request and `render_template` to return the result.

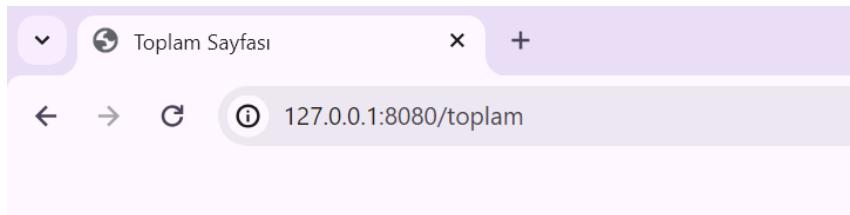
```
from flask import Flask, render_template, request, redirect, url_for
app = Flask(__name__)
@app.route('/')
def page():
    return render_template("index.html")
@app.route('/toplasm', methods=["GET","POST"])
def toplam():
    if request.method == "POST":
        number1 = request.form.get("number1")
        number2 = request.form.get("number2")
        return render_template("number.html", total = int(number1) + int(number2))
    else:
        return redirect(url_for("page"))
if __name__ == '__main__':
    app.run(debug=True, port= 8080)
```

Editor 2 (Bottom): Contains the file `number.html`. It is a simple HTML template with a `<p>` tag that displays the value of the `total` variable.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Toplam Sayfası</title>
</head>
<body>
    <p>Toplamlar: {{total}}</p>
</body>
</html>
```

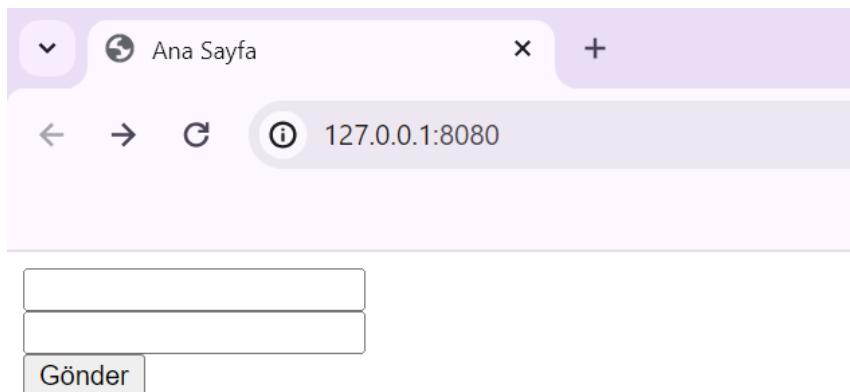
Both editors show the same terminal output at the bottom, indicating a debugger is active and the application is running on port 8080.

Öncesi:



Bu bir get request olduğu için bir değer yoktur!

Ve sonrası:



Flask vs Django



Python web framework'leri söz konusu olduğunda, **Django** ve **Flask** en popüler olan iki framework olarak karşımıza çıkmaktadır. Her ikisi de Python ile yazılan **açık kaynaklı** web framework'ü olsa da, Django daha **geniş kapsamlı** bir framework'ken, Flask **mikro** bir framework'dür.



Django

Django, Python kullanarak web siteleri oluşturmayı kolaylaştıran bir Python framework'dür. Ücretsiz ve açık kaynaklı bir framework'dür. Django, geliştiricilerin hızlı bir şekilde zengin özelliklere sahip, güvenli ve ölçeklenebilir bir web uygulaması oluşturmasına yardımcı olur.

DRY (Don't Repeat Yourself) olarak da adlandırılan bileşenlerin yeniden kullanılabilirliğini vurgulayan Django, oturum açma sistemi, veri tabanı bağlantısı ve CRUD işlemleri (Create Read Update Delete) gibi kullanıma hazır özelliklerle birlikte gelir.

Django'nun Temel Özellikleri

Çok yönlü: Django çeşitli alanlarda kullanılır. İçerik yönetim sistemleri (CMS), sosyal ağ siteleri ve bilgi işlem platformları gibi çeşitli uygulamalar oluşturmak için Django kullanılır.

Taşınabilir: Django, Python'da yazılmıştır. Herhangi bir platformda çalışmak için taşınabilirlik özelliği sağlar.

Güvenli: Django, kullanıcı kimlik doğrulama sistemi sunar. Tıklama hırsızlığı, siteler arası komut dosyası çalıştırma ve bunlar gibi yaygın güvenlik sorunlarından korunmaya yardımcı olur.

Ölçeklenebilir: DRY ilkelerinin birleşimiyle kodun yeniden kullanımı ve bakımı ölçeklenebilirlik sağlar.

Uygunlabilirlik: JSON, HTML, XML ve daha fazlası dahil olmak üzere çeşitli teknolojilerle uyumludur.

Popüler: Spotify, Instagram, Dropbox, Pinterest, Mozilla, YouTube, NASA ve National Geographic gibi şirketler Python framework'ü olarak Django kullanır.

Flask

Flask, geliştiricilerin tercih ettikleri veri tabanını ve eklentileri seçme esnekliği sunan Python tabanlı bir mikro framework'dür. Flask, açık kaynak kodlu bir web framework'üdür. Küçük ve orta ölçekli uygulamaların geliştirilmesi için ideal bir araçtır.

Flask, kolay entegrasyon sağlamaası ve geniş bir kullanıcı tabanına sahip olması nedeniyle popülerdir. Kullanımı kolay ve basit bir API'ye sahiptir ve hızlı bir şekilde öğrenilir.

Flask'ın Temel Özellikleri

Hafif ve Genişletilebilir: Geliştiriciler, uygulama mimarisi, kitaplıklar ve uzantılar üzerinde kontrole sahiptir.

Basit Arayüz: Flask'ın kullanımı kolaydır.

İstek İşleme: HTTP ve RESTful isteklerini destekler.

Test Etme ve Hata Ayıklama: Flask, test ve hata ayıklayıcı araçlarını sunar.

Esnek ve Ölçeklenebilir: WSGI şablonları desteği, esneklik ve ölçeklenebilirlik sağlar.

Popüler: Netflix, Reddit, Uber, Lyft, Zillow, Patreon, Airbnb ve MIT gibi büyük şirketler tarafından kullanılır.

Flask ve Django Ne Zaman Kullanılır?

Aşağıdakileri yapmanız gerekiğinde **Django** kullanabilirsiniz:

- Kesin teslim tarihleri olan çok sayfalı ve büyük projelerde
- Geliştirmeyi hızlandırmak istediğinizde
- Erişim desteği almak istediğinizde
- Güvenli projeler oluşturmak istediğinizde
- Gelecekte projeleri büyütme veya daha karmaşık hale getirme planınız varsa
- Yerel ORM desteği ile web uygulamaları oluşturmak istediğinizde

Aşağıdakileri yapmanız gerekiğinde **Flask** kullanabilirsiniz:

- Daha küçük projeler üzerinde çalışırken
- Veri tabanı desteğine ihtiyacınız olduğunda
- Kitaplıklar ve uzantıları seçme esnekliğine ve özgürlüğüne sahip olmak istediğinizde
- Gelecekte projeye yeni uzantılar eklemek istiyorsanız
- Statik web siteleri, hızlı prototipler ve RESTful web hizmetleri oluşturmak istiyorsanız