Back End

Basic Fundamentals

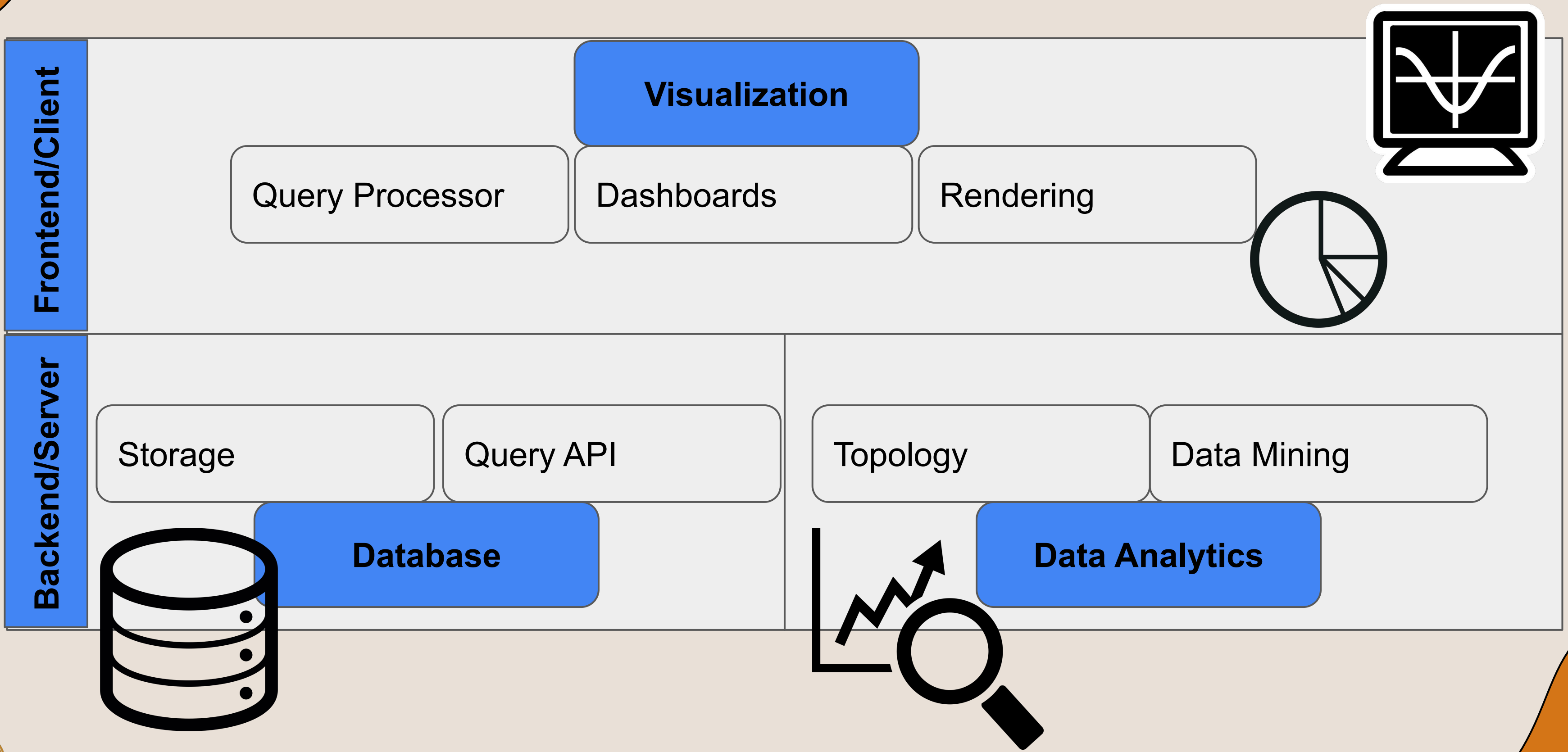
Table of Contents

- What Budget Collector Uses?
- Visualization System
- Client and Server
- HTTP Requests
- Flask
- Summary

What Budget Collector Uses?

- Front-end:
  - JavaScript, HTML, CSS
  - built within the wordpress ecosystem
- Back-end:
  - Airtable
    - Contains art information (artist names, images, region, etc)
    - CANNOT manipulate data within airtable
  - AWS EC2
    - Backend server and database for website
    - CAN post manipulated data or run backend code
- Your Github Repo:
  - Needs to detail:
    - how to run it locally
    - what front-end & backend technologies that your using (flask, django, etc)

Visualization System



Frontend/Client

Visualization

Query Processor

Dashboards

Rendering

Backend/Server

Storage

Query API

Database

Topology

Data Mining

Data Analytics

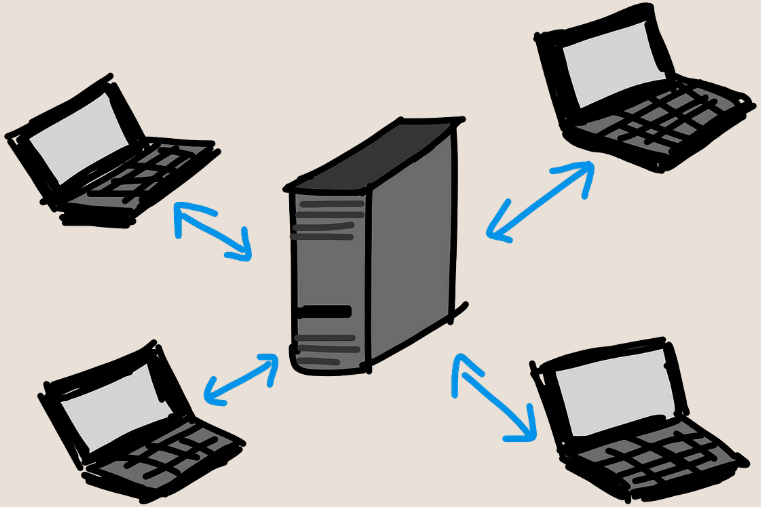
Client and Server

Client

- Any internet connected device or software (e.g. iphone, web browser, etc)
- **Makes HTTP requests**

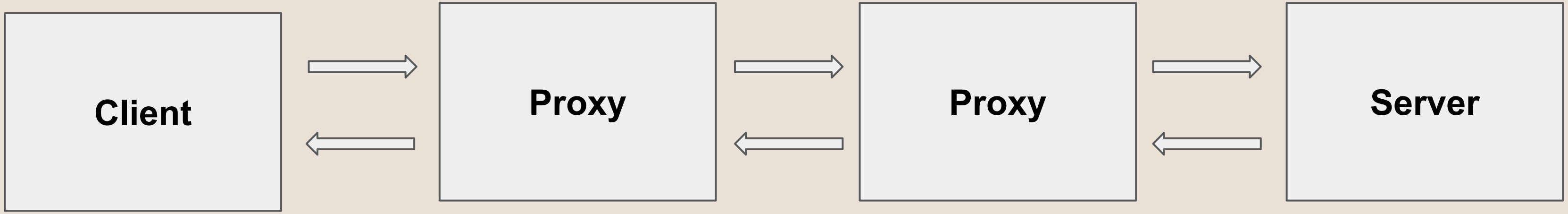
Server

- A device that stores web server software and component files (e.g. html, cvs, etc)
- **Responds to clients with component files (i.e. data)**



HTTP Requests

- HTTP is a request-response protocol for supporting client-server communications
- Proxy: Entities in between the client server request (eg, modems, routers, etc)



Client

Proxy

Proxy

Server

From: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

HTTP Requests: Methods

GET: Request a representation of the specified data.

POST: Submits data to the server.

HEAD: Request information similar to GET, but without response body

PUT: Submits data to replace an existing resource.

DELETE: Deletes the specified data.

CONNECT: Starts communication with the server.

OPTIONS: Details the options allowed with the server.

TRACE: A loopback test.

PATCH: Makes partial modifications to a specified data on the server.

HTTP Requests: GET and POST Examples

GET

- Visible to everyone within the URL. Less secure.

```
GET /test?field1=value1&field2=value2
```

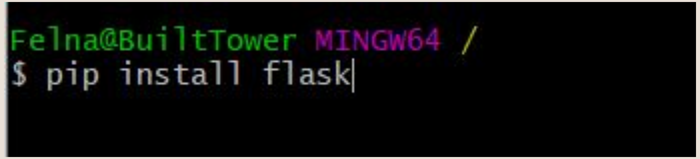
POST

- Data not displayed in the URL. Safer.

```
POST /test HTTP/1.1
Host: foo.example
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

field1=value1&field2=value2
```

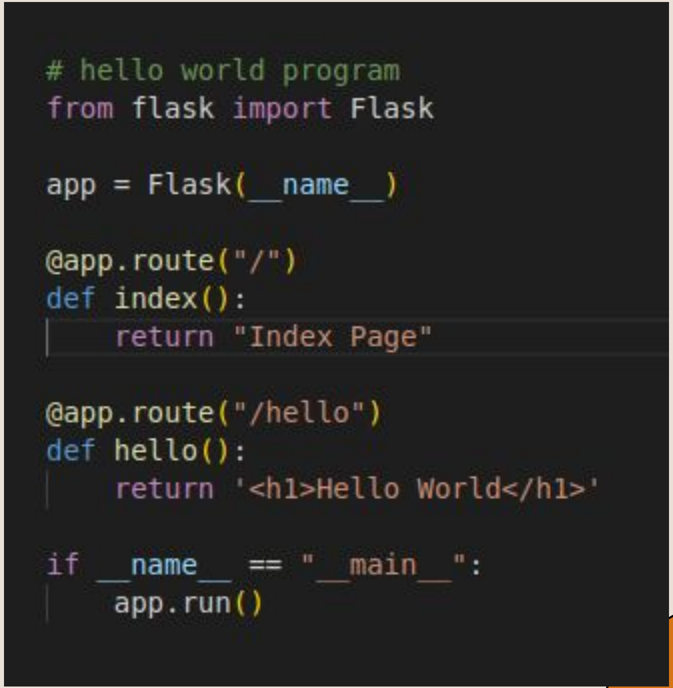
Flask

- Python web framework for developing applications
- Can build smaller applications compared to Django
- Easily changeable and integrates with front-end and back-end applications



```
Felna@BuiltTower MINGW64 /
$ pip install flask
```

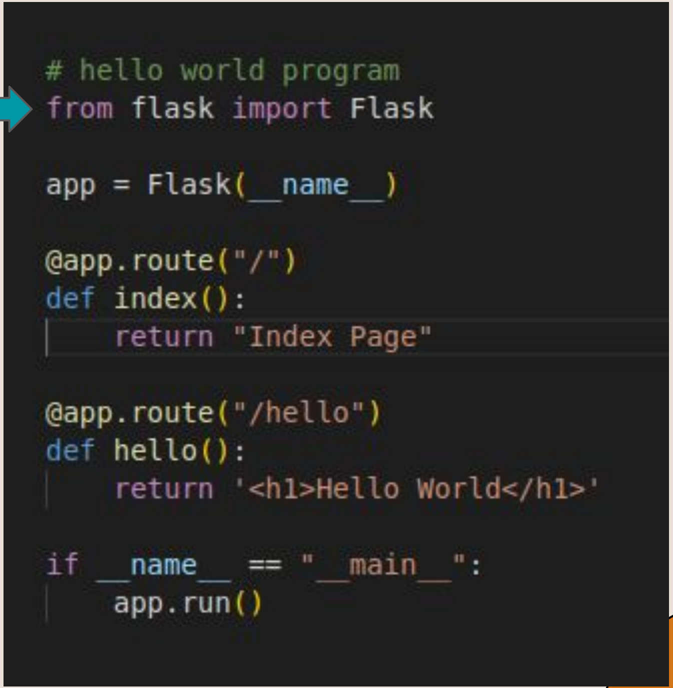
Flask: First Program



```
# hello world program
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "Index Page"

@app.route("/hello")
def hello():
    return '<h1>Hello World</h1>'

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

Flask: First Program

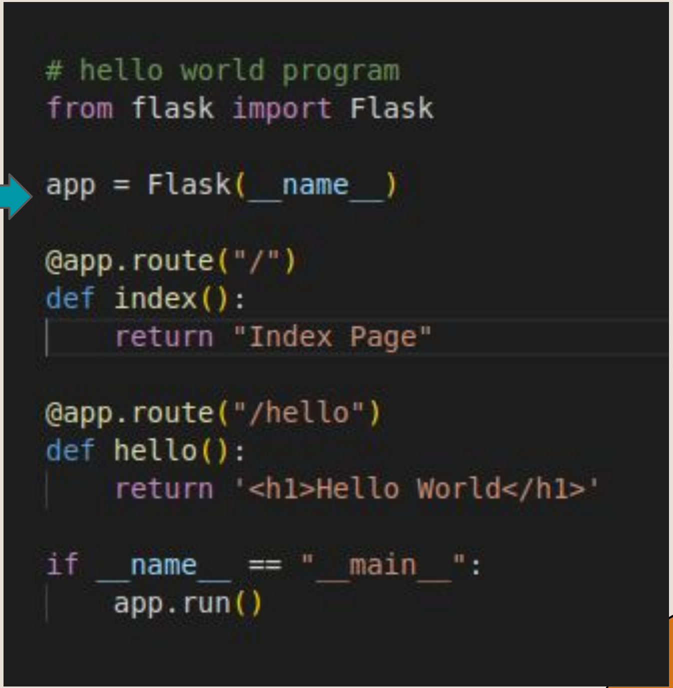
1. Import Flask class



```
# hello world program
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "Index Page"

@app.route("/hello")
def hello():
    return '<h1>Hello World</h1>'

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

Flask: First Program

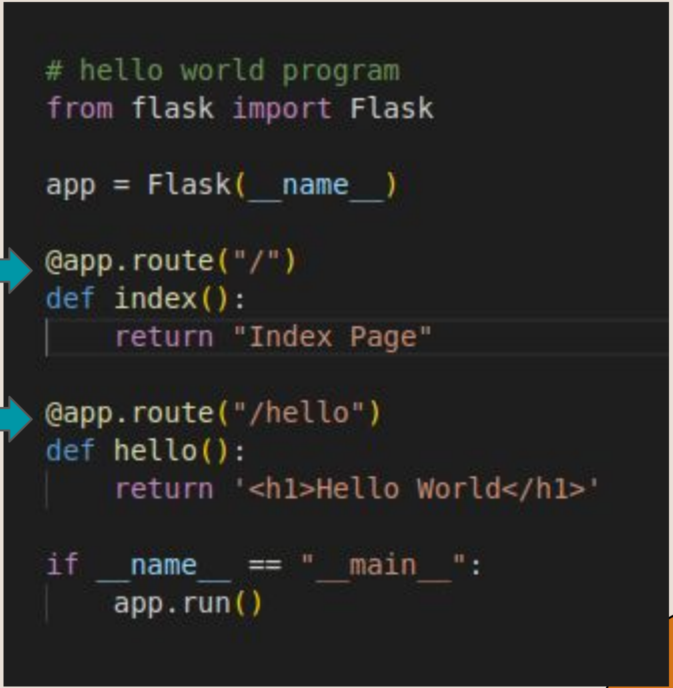
1. Import Flask class
2. Create an instance of the class



```python
# hello world program
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "Index Page"

@app.route("/hello")
def hello():
    return '<h1>Hello World</h1>'

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

Flask: First Program

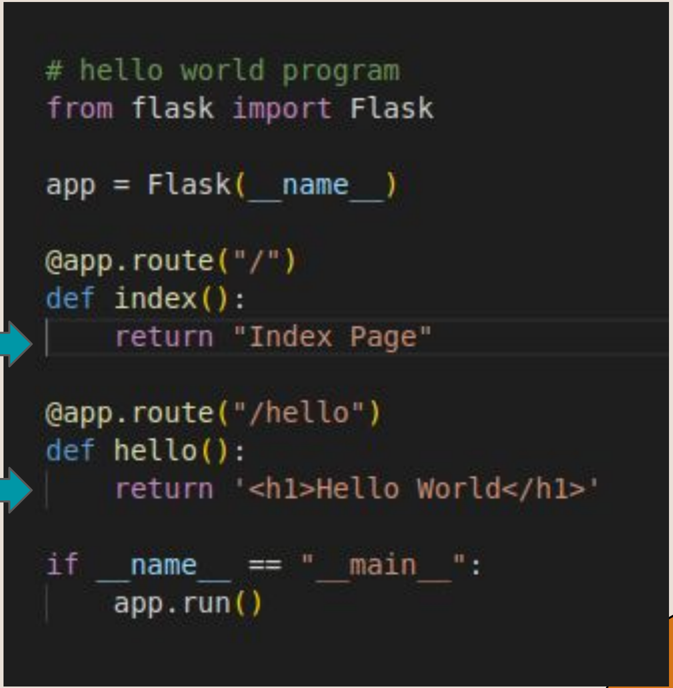
1. Import Flask class
2. Create an instance of the class
3. Use route() decorator to tell which URL triggers functions



```python
# hello world program
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "Index Page"

@app.route("/hello")
def hello():
    return '<h1>Hello World</h1>'

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

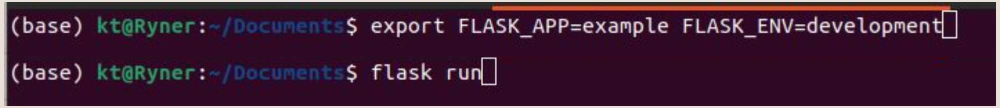
Flask: First Program

1. Import Flask class
2. Create an instance of the class
3. Use route() decorator to tell which URL triggers functions
4. Function returns message to display in browser

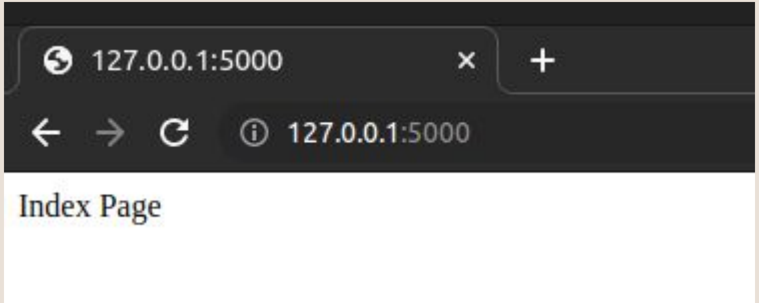


```python
# hello world program
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "Index Page"

@app.route("/hello")
def hello():
    return '<h1>Hello World</h1>'

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

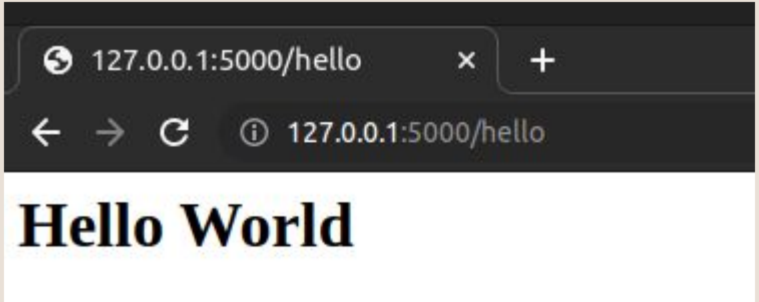
Flask: Running Dev Environment



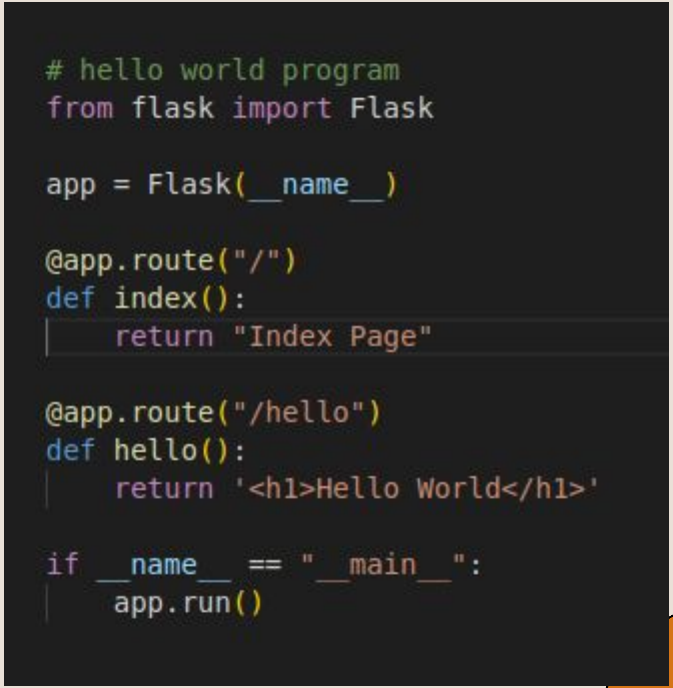
```
(base) kt@Ryner:~/Documents$ export FLASK_APP=example FLASK_ENV=development
(base) kt@Ryner:~/Documents$ flask run
```



Index Page

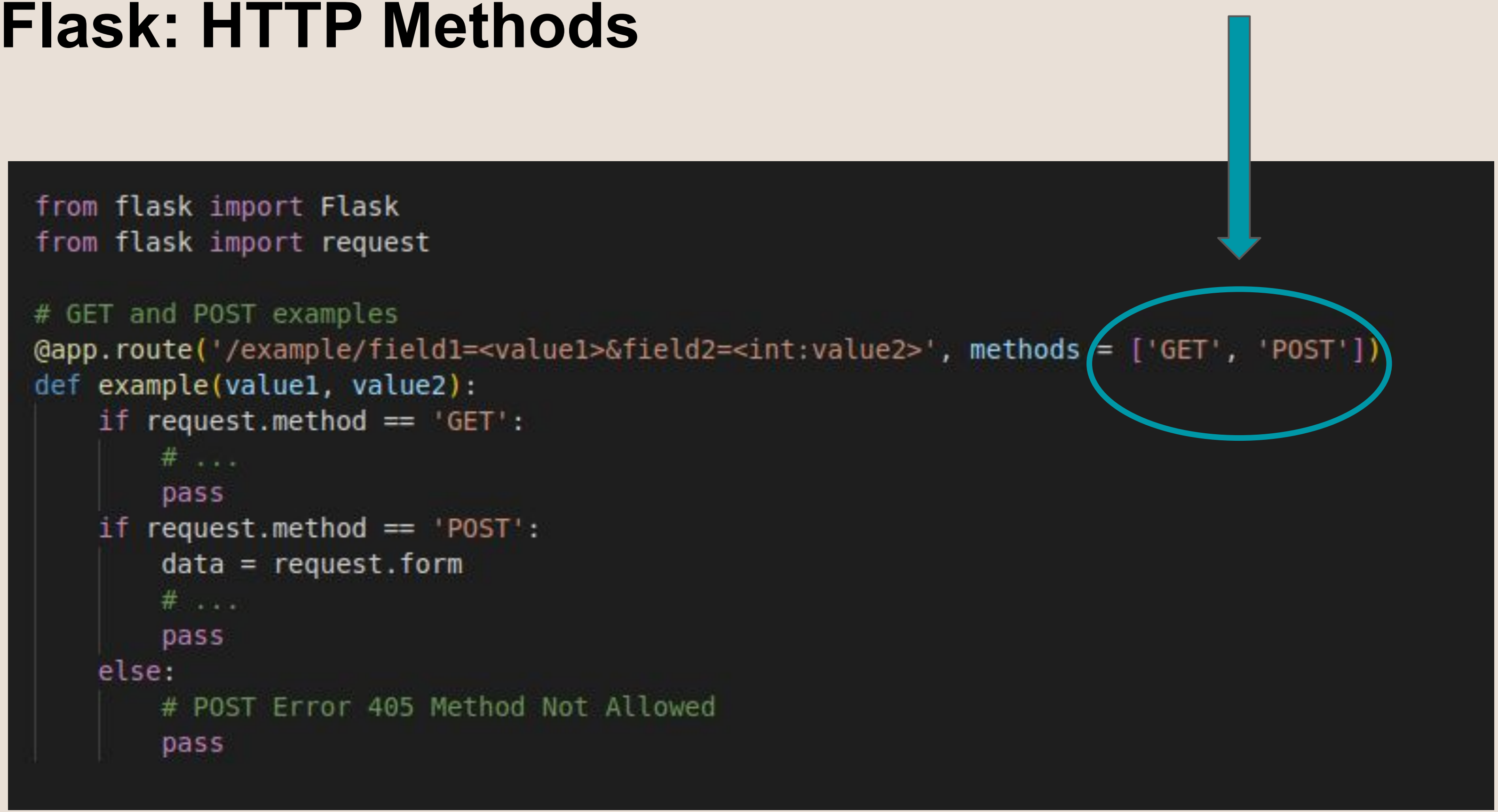


Hello World



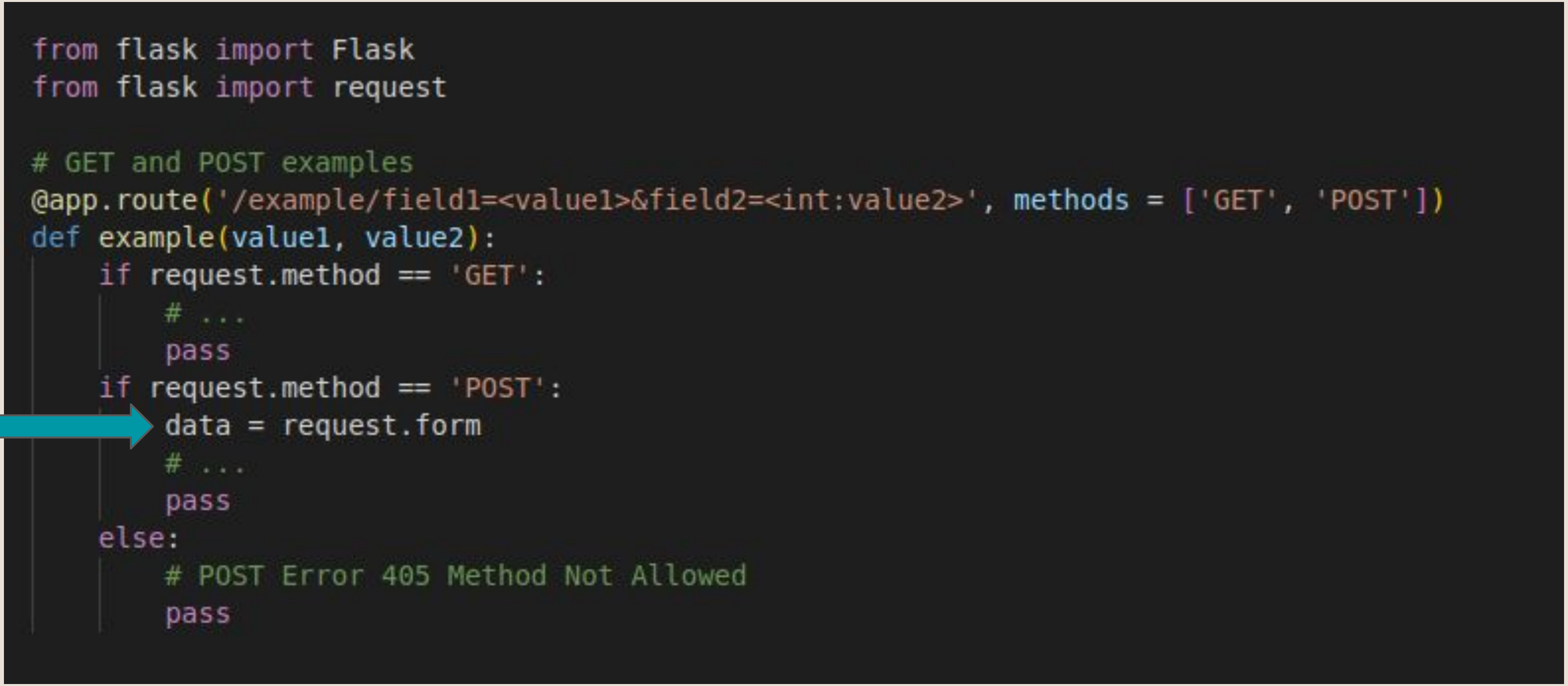
```python
# hello world program
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "Index Page"

@app.route("/hello")
def hello():
    return '<h1>Hello World</h1>'

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

Flask: HTTP Methods



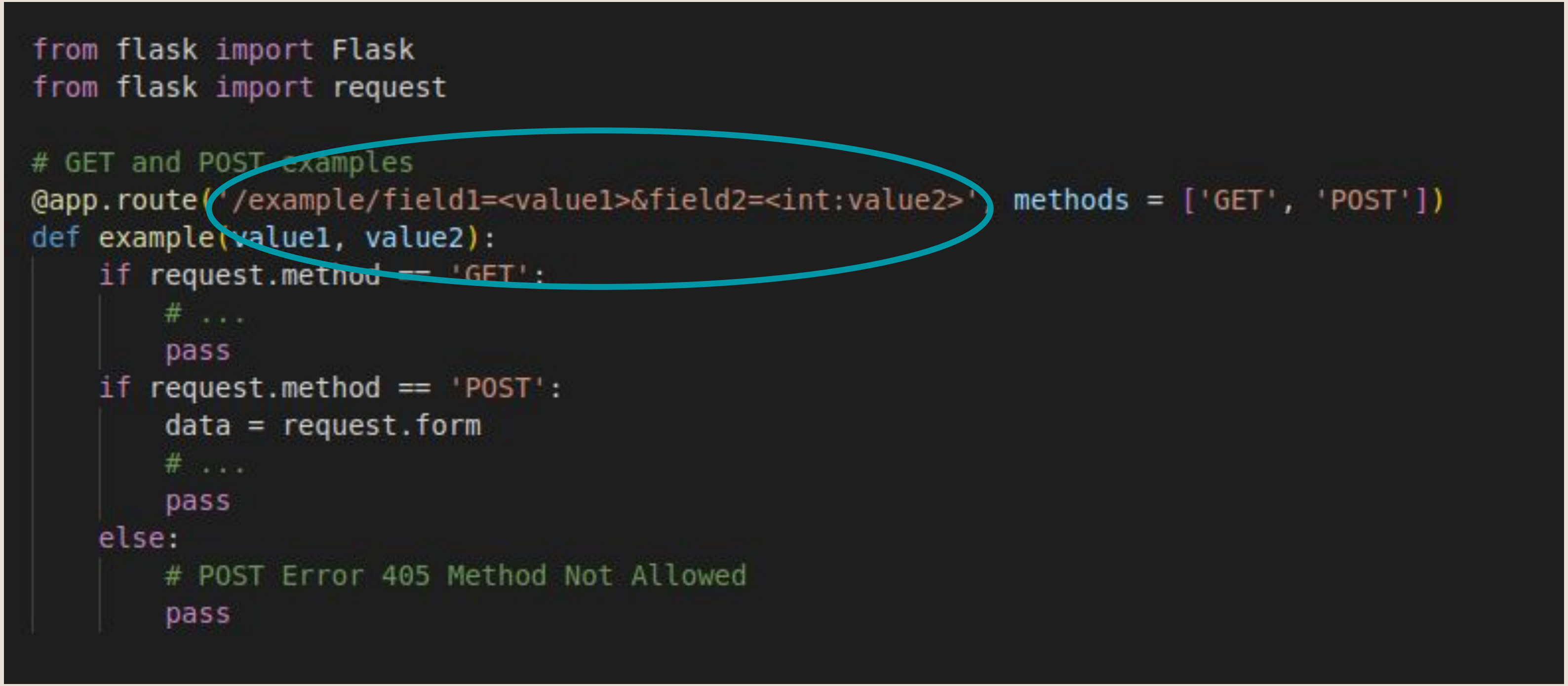
```python
from flask import Flask
from flask import request

# GET and POST examples
@app.route('/example/field1=<value1>&field2=<int:value2>', methods = ['GET', 'POST'])
def example(value1, value2):
    if request.method == 'GET':
        # ...
        pass
    if request.method == 'POST':
        data = request.form
        # ...
        pass
    else:
        # POST Error 405 Method Not Allowed
        pass
```

Flask: HTTP Methods



```python
from flask import Flask
from flask import request

# GET and POST examples
@app.route('/example/field1=<value1>&field2=<int:value2>', methods = ['GET', 'POST'])
def example(value1, value2):
    if request.method == 'GET':
        # ...
        pass
    if request.method == 'POST':
        data = request.form
        # ...
        pass
    else:
        # POST Error 405 Method Not Allowed
        pass
```

Flask: HTTP Methods



```
from flask import Flask
from flask import request

# GET and POST examples
@app.route('/example/field1=<value1>&field2=<int:value2>', methods = ['GET', 'POST'])
def example(value1, value2):
    if request.method == 'GET':
        # ...
        pass
    if request.method == 'POST':
        data = request.form
        # ...
        pass
    else:
        # POST Error 405 Method Not Allowed
        pass
```

Summary

- HTTP protocols allows you to send and receive data from the users
- Web application might try multiple different methods at identical URLs
- Flask is a micro framework for HTTP protocols in python.
- Dynamic routing allows for the use of variable names in the URL