ENSF 381
Full Stack Web Development

Lecture 28: Flask

Slides: Ahmad Abdellatif, PhD

Instructor: Novarun Deb, PhD

UNIVERSITY OF CALGARY

Outline

- Introduction to Flask.
- Key benefits.
- Routes.
- Dynamic routes.
- HTTP methods.

What is Flask?

- A lightweight, versatile and extensible web framework for Python.

- Provides essential features and functionality for building web applications, while extensions provide the rest.

- To use Flask, we need to install it by:

```
pip install Flask
```

Key features

- **Routing**: allows you to define URL patterns and associate them with functions, making it easy to create routes for different parts of your application.

- **Extension Ecosystem:** Flask has a rich ecosystem of extensions that provide additional functionality and integration with third-party services.

- **Web Development Server**: Flask comes with a built-in development server, making it convenient for testing and development.

- **Request and Response Handling**: Flask simplifies handling HTTP requests and responses. You can easily access request data, cookies, and form submissions, and construct responses with various content types.

Routing

- In web development, routing refers to the mechanism of mapping URLs to specific functions that handle the requested resource.
- The Flask application knows what code it needs to run for each URL requested.
- Allows breaking down the application into smaller, manageable components.
- Improves code readability and maintainability.

Flask – example

```python
# Import the Flask class
from flask import Flask

# Create an instance of the Flask class
app = Flask(__name__)

# Define a route and the corresponding function
@app.route('/')
def index():
    return 'Hello, World!'

# Run the application if this script is executed
if __name__ == '__main__':
    app.run()
```

1. app instance: create an instance of the Flask class, typically passing __name__ as an argument.

2. @app.route() decorator: associate a URL route with a Python function that corresponds to the route.

Flask – example

Running the app: run() method runs the Flask application, and it has four optional parameters:

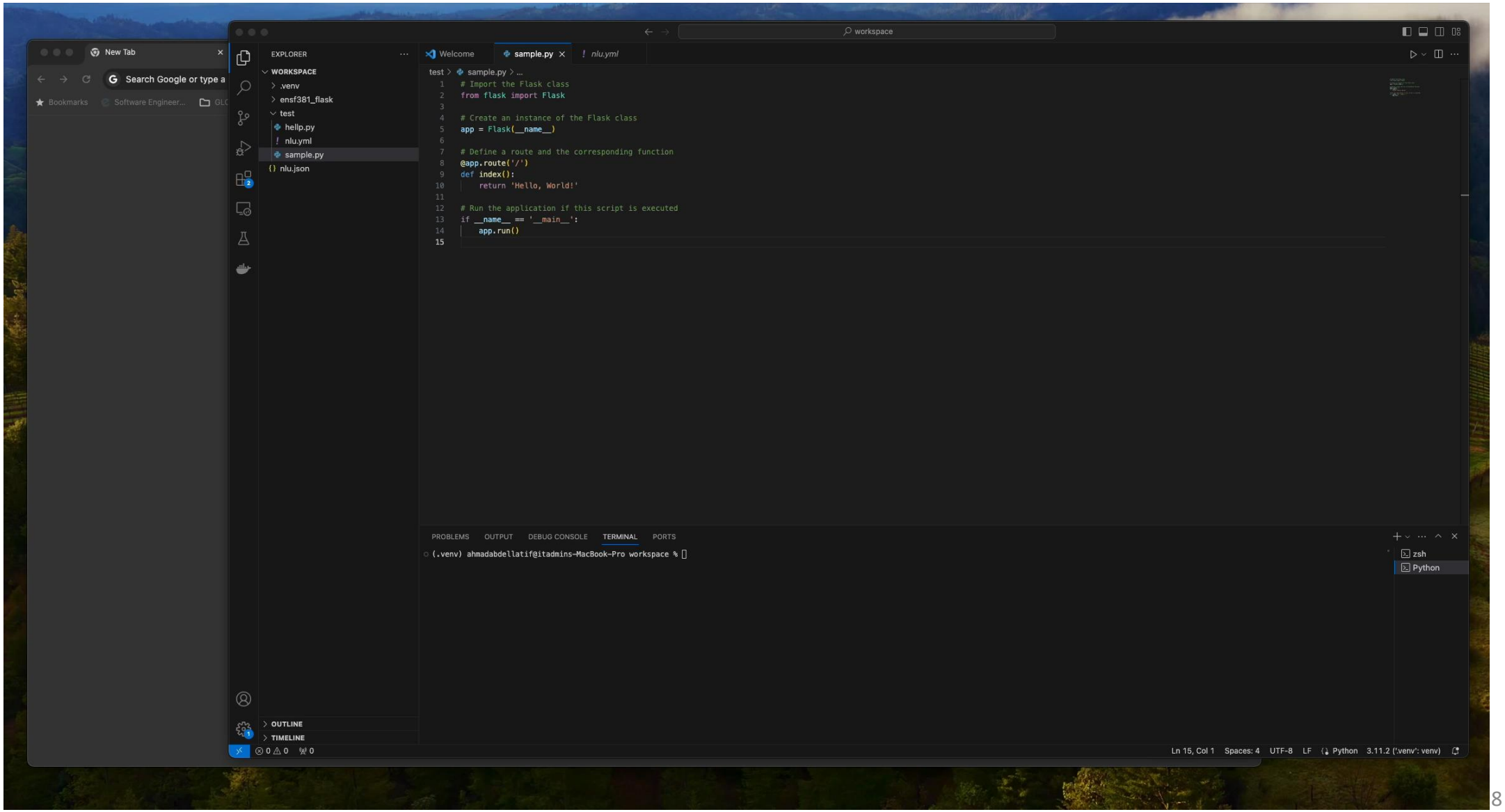
1. **host**: specifies the hostname to listen on. Defaults to 127.0.0.1 (localhost).

2. **port**: Specifies the port on which the server will listen. The default is 5000, but you can change it if needed.

3. **debug**: when set to True, enables the development mode, which provides more detailed error messages and automatically reloads the server on code changes. The default is false.

4. **options**: allows passing additional configuration options to the underlying server.

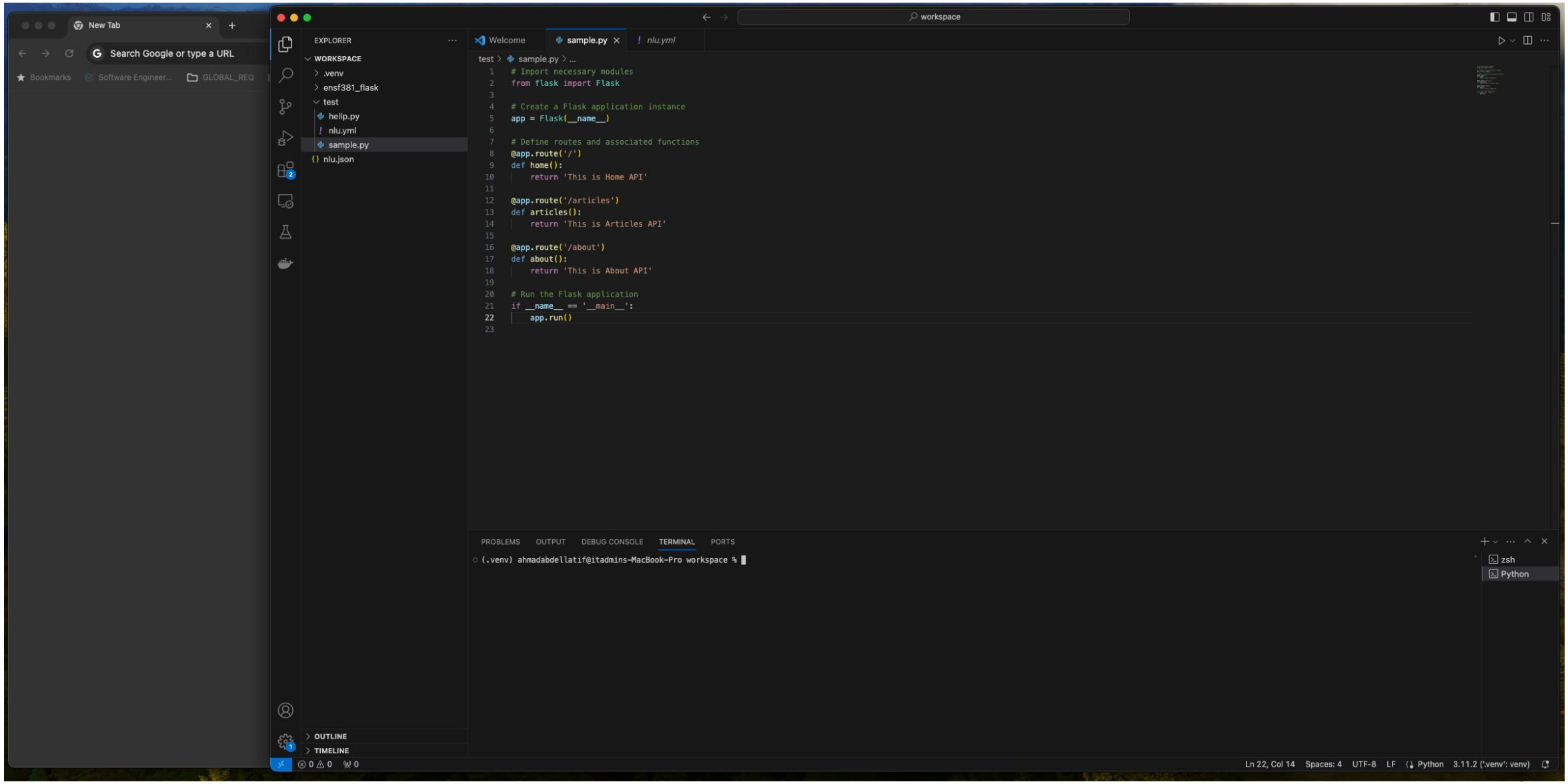
Flask – example



Routing - example

```python
from flask import Flask

app = Flask(__name__)

# Define routes and associated functions
@app.route('/')
def home():
  return 'This is Home API'

@app.route('/articles')
def articles():
  return 'This is Articles API'

@app.route('/about')
def about():
  return 'This is About API'

# Run the Flask application
if __name__ == '__main__':
  app.run()
```

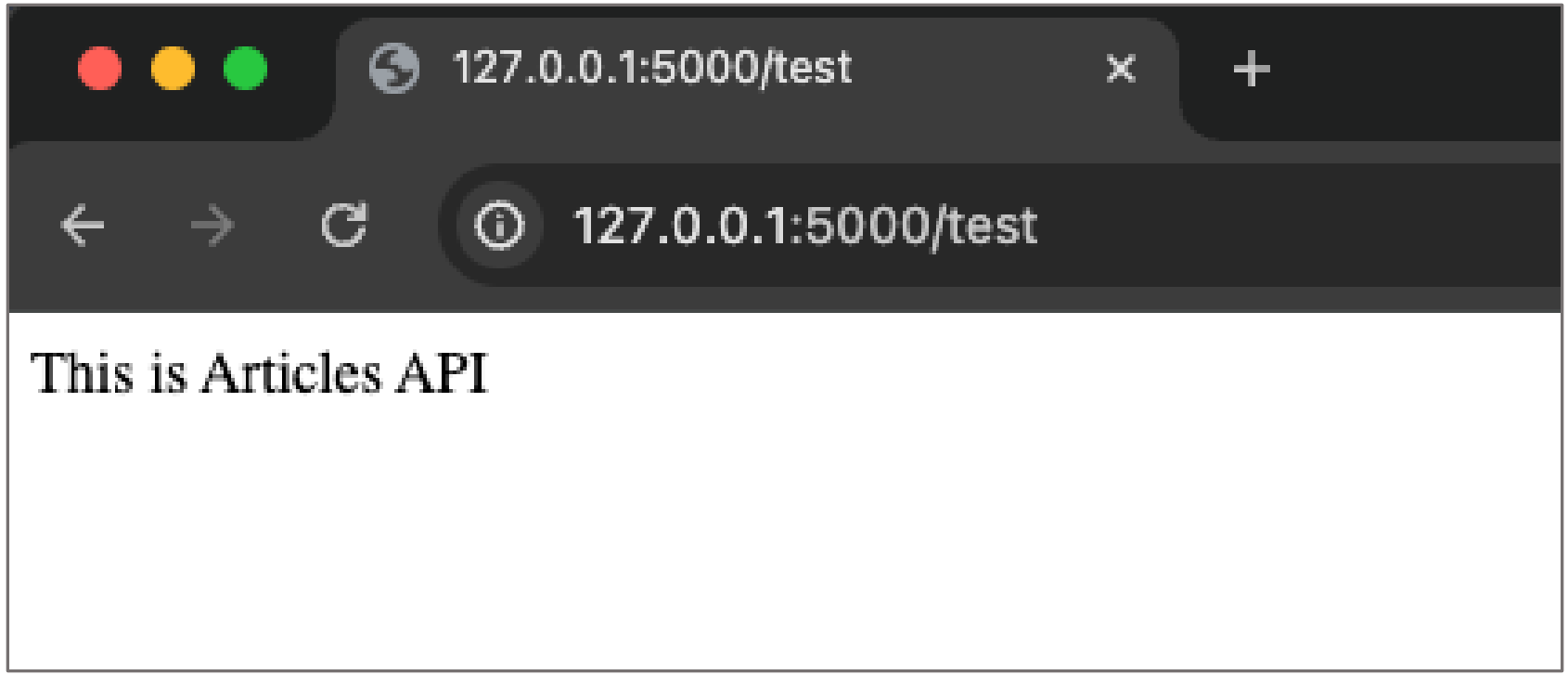
Routing - example



What is the output of “http://127.0.0.1:5000/test”?

```python
from flask import Flask

app = Flask(__name__)

# Define routes and associated functions
@app.route('/')
def home():
  return 'This is Home API'

@app.route('/test')
def articles():
  return 'This is Articles API'

@app.route('/about')
def about():
  return 'This is About API'

# Run the Flask application
if __name__ == '__main__':
  app.run()
```

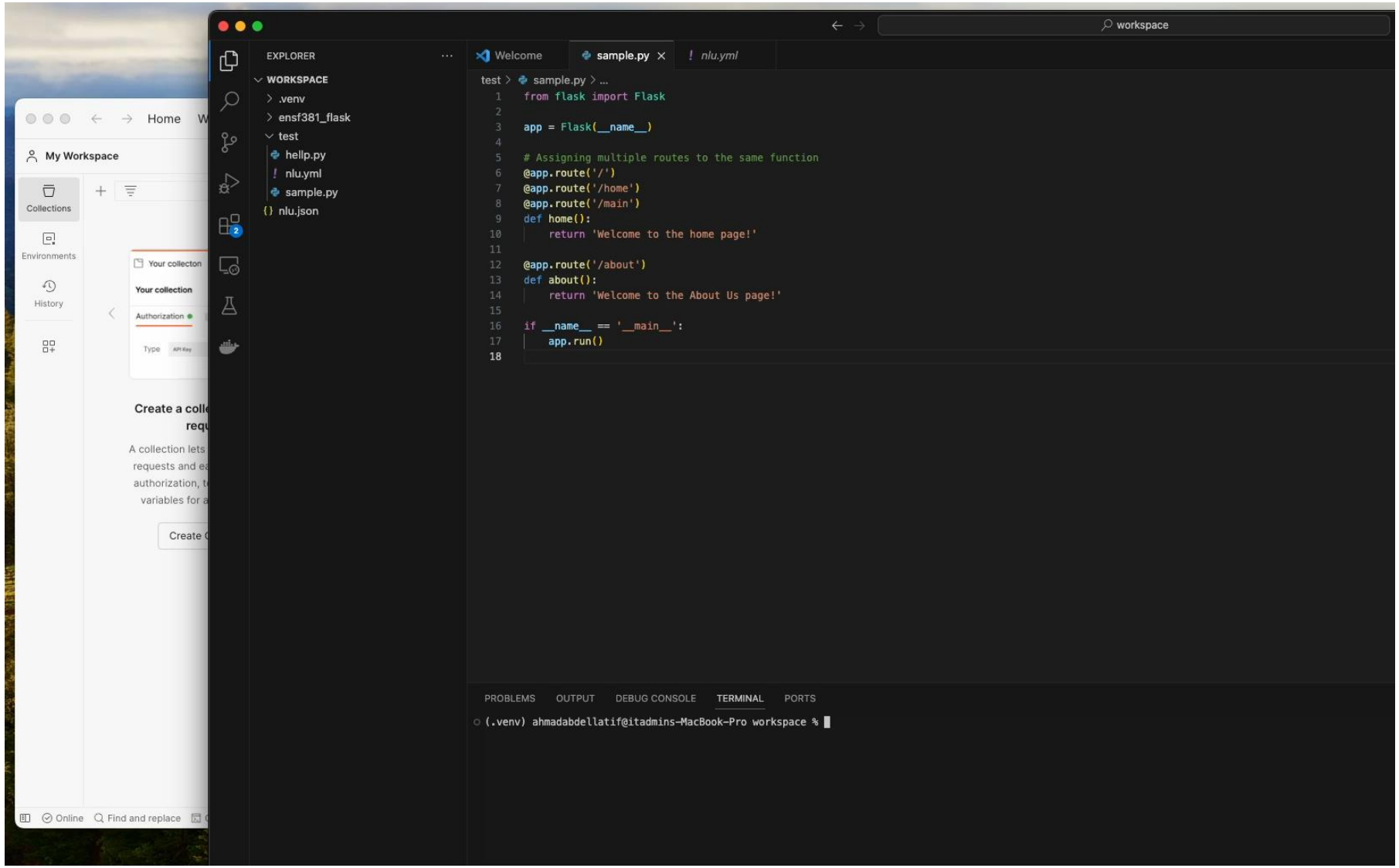
What is the output of “http://127.0.0.1:5000/test”?



Assigning multiple routes to the same function

```
from flask import Flask

app = Flask(__name__)

# Assigning multiple routes to the same function
@app.route('/')
@app.route('/home')
@app.route('/main')
def home():
  return 'Welcome to the home page!'

@app.route('/about')
def about():
  return 'Welcome to the About Us page!'

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

Assigning multiple routes to the same function



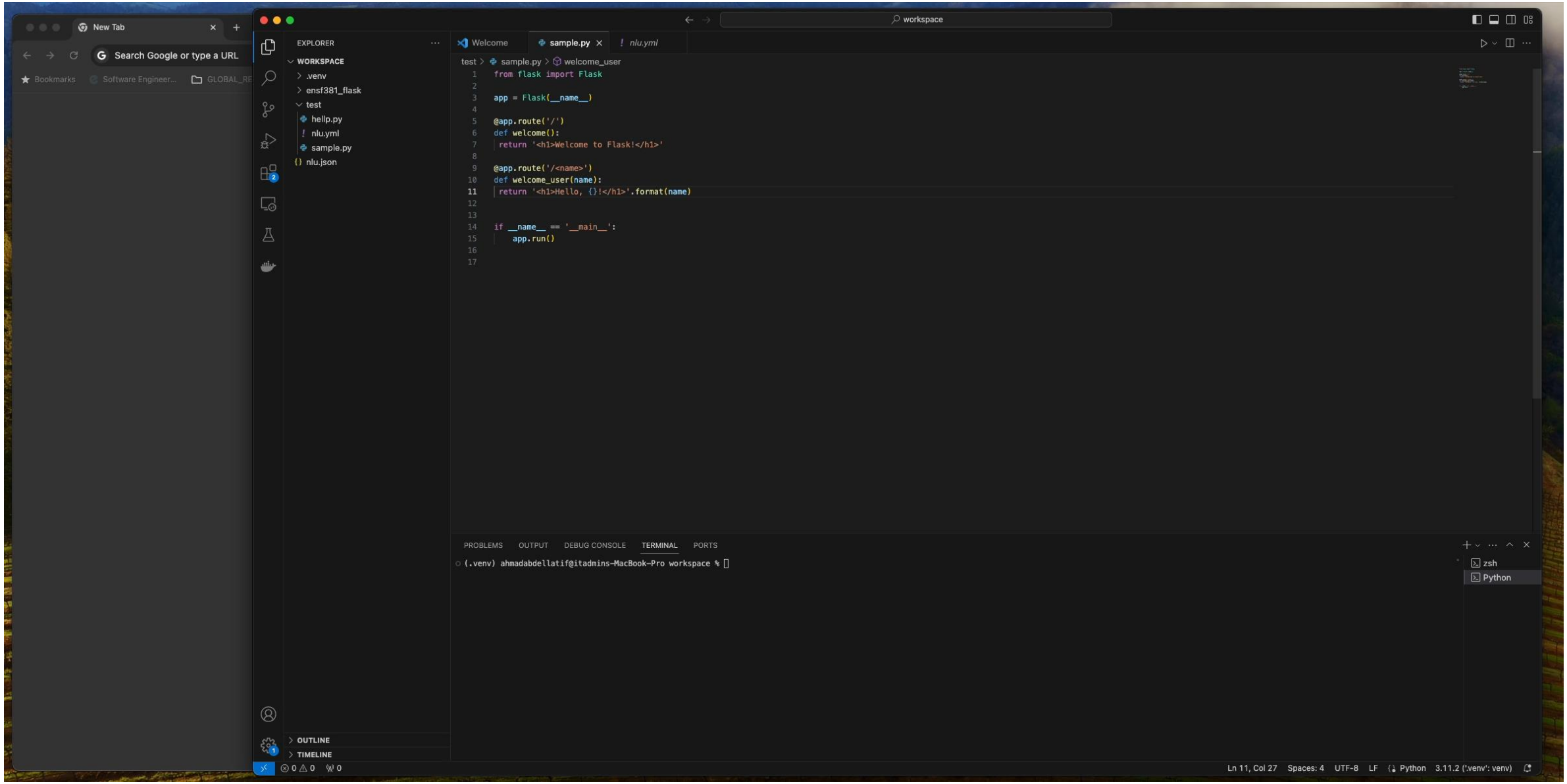
Dynamic routes

- A flexible URLs that can handle variable components.
- Instead of defining a fixed set of routes, you can use dynamic route parameters to capture values from the URL.
- Useful when building web applications with variable data, such as user profiles, product pages, or blog posts.
- The variable parts are enclosed in **< >** brackets and act as placeholders for the values that will be extracted from the actual URL during runtime.

Dynamic routes - example

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def welcome():
 return '<h1>Welcome to Flask!</h1>'

@app.route('/<name>')
def welcome_user(name):
 return '<h1>Hello, {}!</h1>'.format(name)


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

Dynamic routes - example



Dynamic routes – example 2

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def welcome():
  return '<h1>Welcome to Flask!</h1>'

@app.route('/<name>')
def welcome_user(name):
return '<h1>Hello, {}!</h1>'.format(name)

@app.route('/user/<name>')
def user_point(name):
return '<h1>Hello {}, from USER API!</h1>'.format(name)

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

Dynamic routes – example 2

Recap: HTTP methods

- GET: retrieve data from a specified resource.
  - Example: fetching a webpage, image, or any resource without modifying the server's state.
- POST: submit data to be processed to a specified resource.
  - Example: submitting a form, uploading a file, or making a request that results in the creation of a new resource on the server.
- PUT: update a resource or create a new resource if it does not exist.
  - Example: updating user profile information, uploading a new version of a file.
- DELETE: delete a specified resource.
  - Example: deleting a user account, removing a file from a server.

HTTP Methods

- Flask supports several HTTP methods that are commonly used in web development.

- Each HTTP method corresponds to a different type of operation you can perform on a resource.

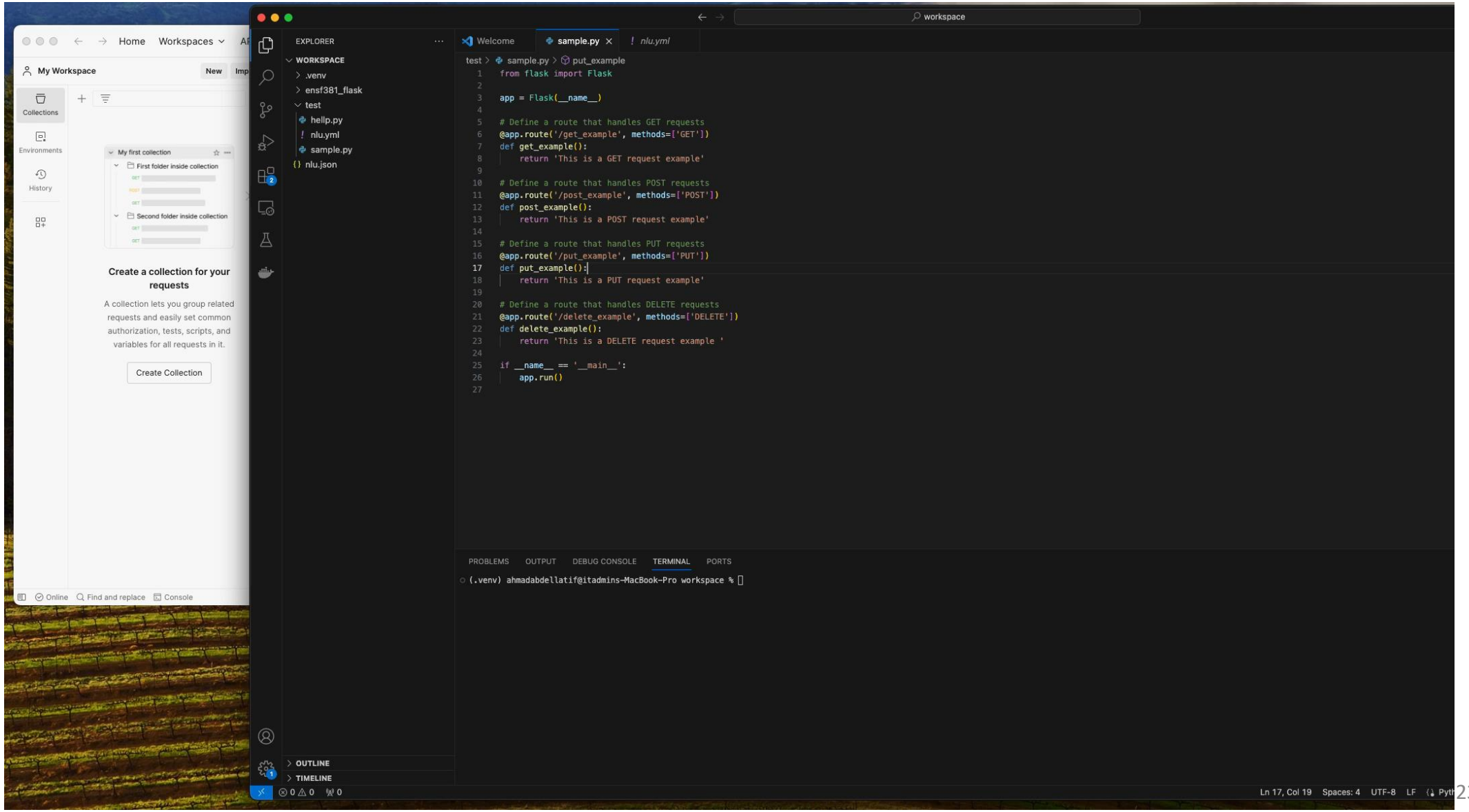
```
from flask import Flask

app = Flask(__name__)

@app.route('/home', methods=['GET'])
def home():
    return 'This is the home page'
```

Specify the HTTP methods:
GET, POST, PUT, and DELETE

HTTP Methods - example

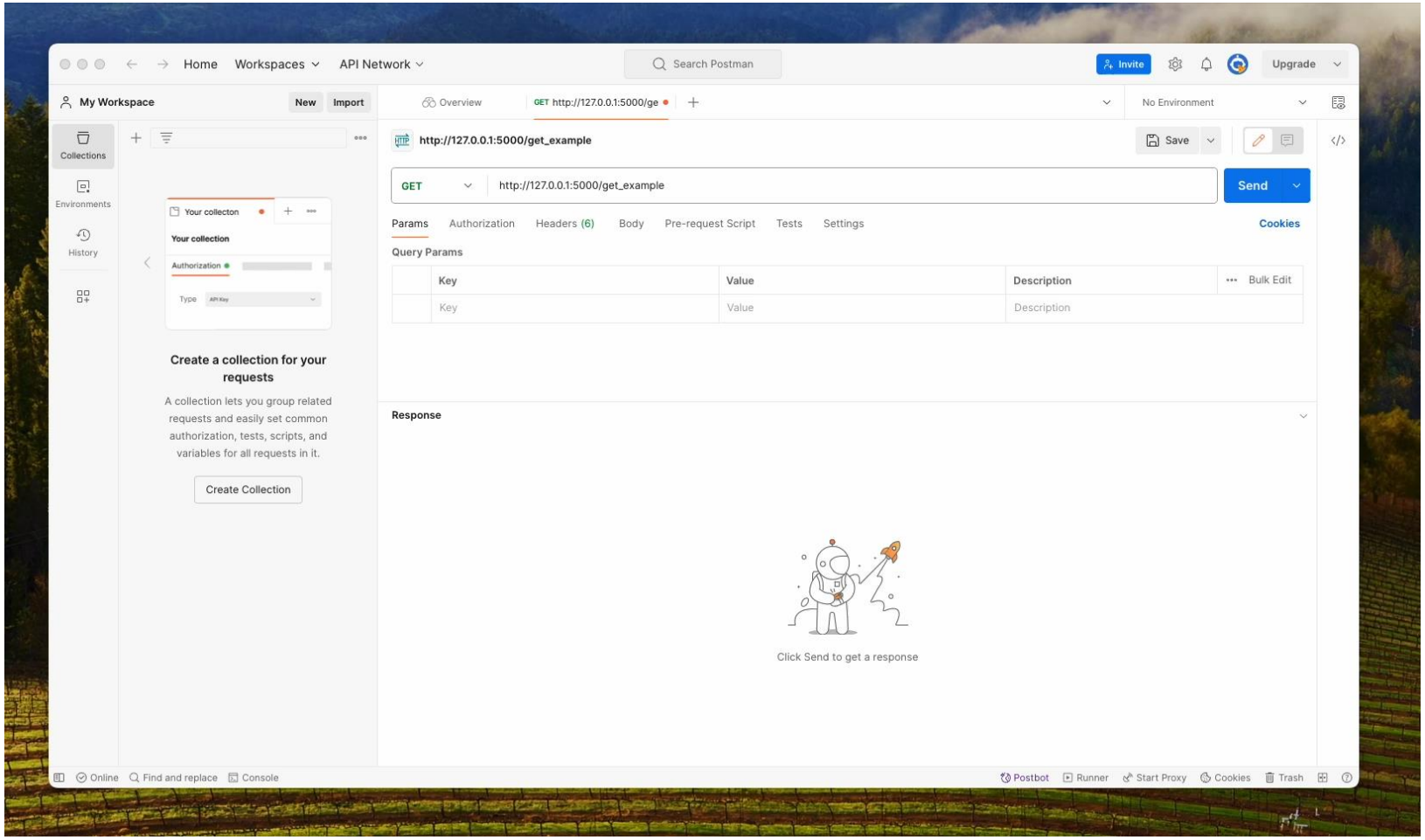
```
from flask import Flask

app = Flask(__name__)

# Define a route that handles GET requests
@app.route('/get_example', methods=['GET'])
def get_example():
  return 'This is a GET request example'

# Define a route that handles POST requests
@app.route('/post_example', methods=['POST'])
def post_example():
  return 'This is a POST request example'

# Define a route that handles PUT requests
@app.route('/put_example', methods=['PUT'])
def put_example():
  return 'This is a PUT request example'

# Define a route that handles DELETE requests
@app.route('/delete_example', methods=['DELETE'])
def delete_example():
  return 'This is a DELETE request example '

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

HTTP Methods - example



Question

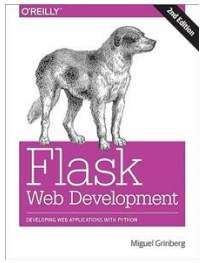
What would be the result when using the HTTP POST method on 'http://127.0.0.1:5000/get_example'?



Questions

References

- Flask Web Development: Developing Web Applications with Python.



- https://flask.palletsprojects.com/en/3.0.x/

- https://www.tutorialspoint.com/flask/index.htm

Final Exam

Date: April 17th, 2025 (Thursday)

Time: 5:00PM - 7:30PM

Location: Aux Gym Exam