TX00FL42-3001

# SETTING UP A BACKEND SERVICE WITH AN INTERFACE

MUATH OTHMAN

## LET’S CHECK HOMEWORKS!

## LEARNING OBJECTIVES

- Understand how to set up a Flask-based backend service in Python
- Create and test HTTP endpoints to handle requests and provide responses
- Implement JSON responses and error handling to ensure robust and user-friendly API design.

In this module you will learn to implement a backend in Python. This way you can build a web service so that the HTML, CSS or JavaScript user interface (UI) communicates with the HTTP endpoints provided by the backend written in Python.

The user of a backend does not necessarily have to be a browser. With the approach presented here, the backend service can be used programmatically from any service with any programming language thanks to the HTTP connection protocol.

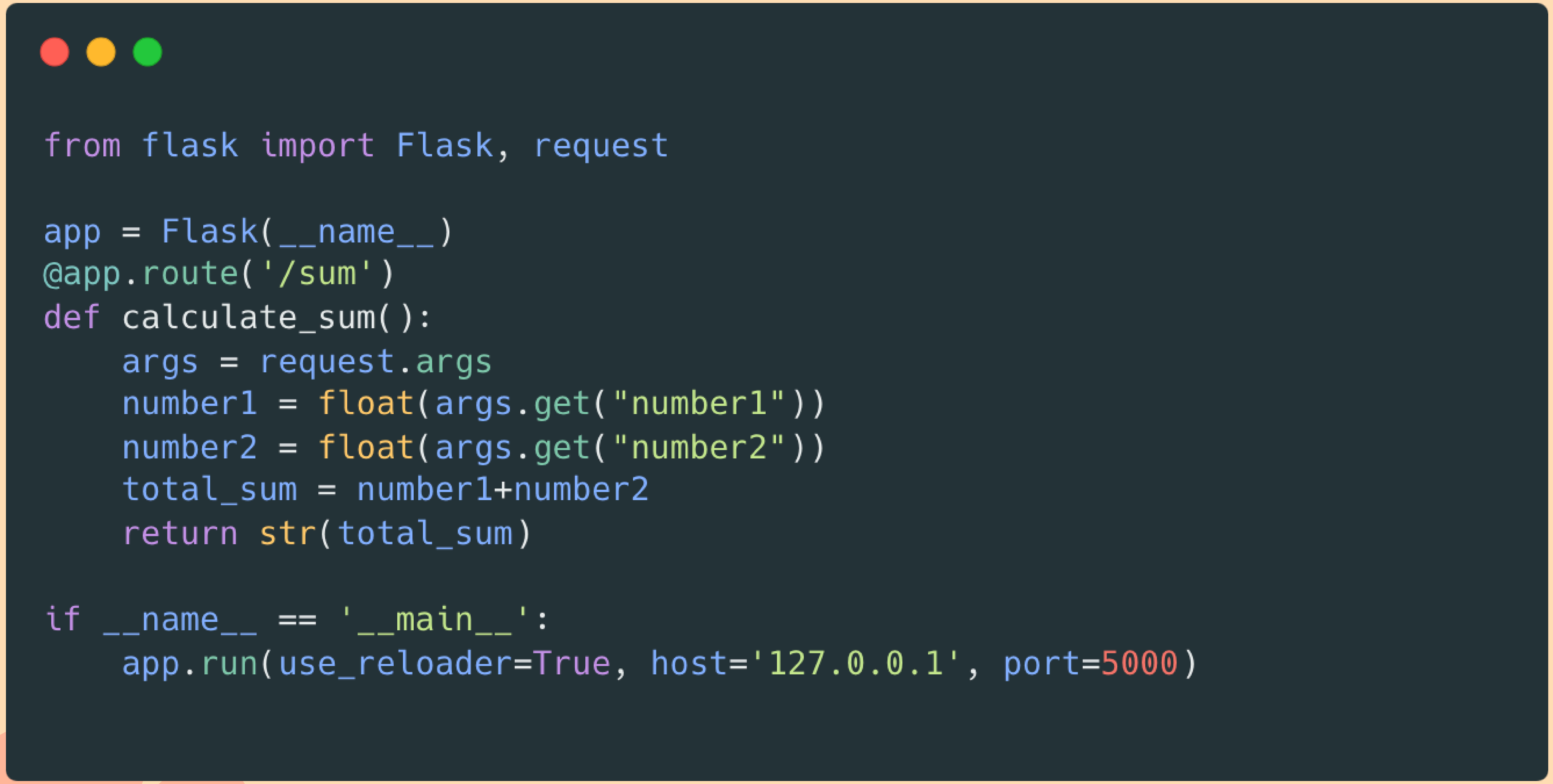
# WHAT IS

# Flask

web development,
one drop at a time

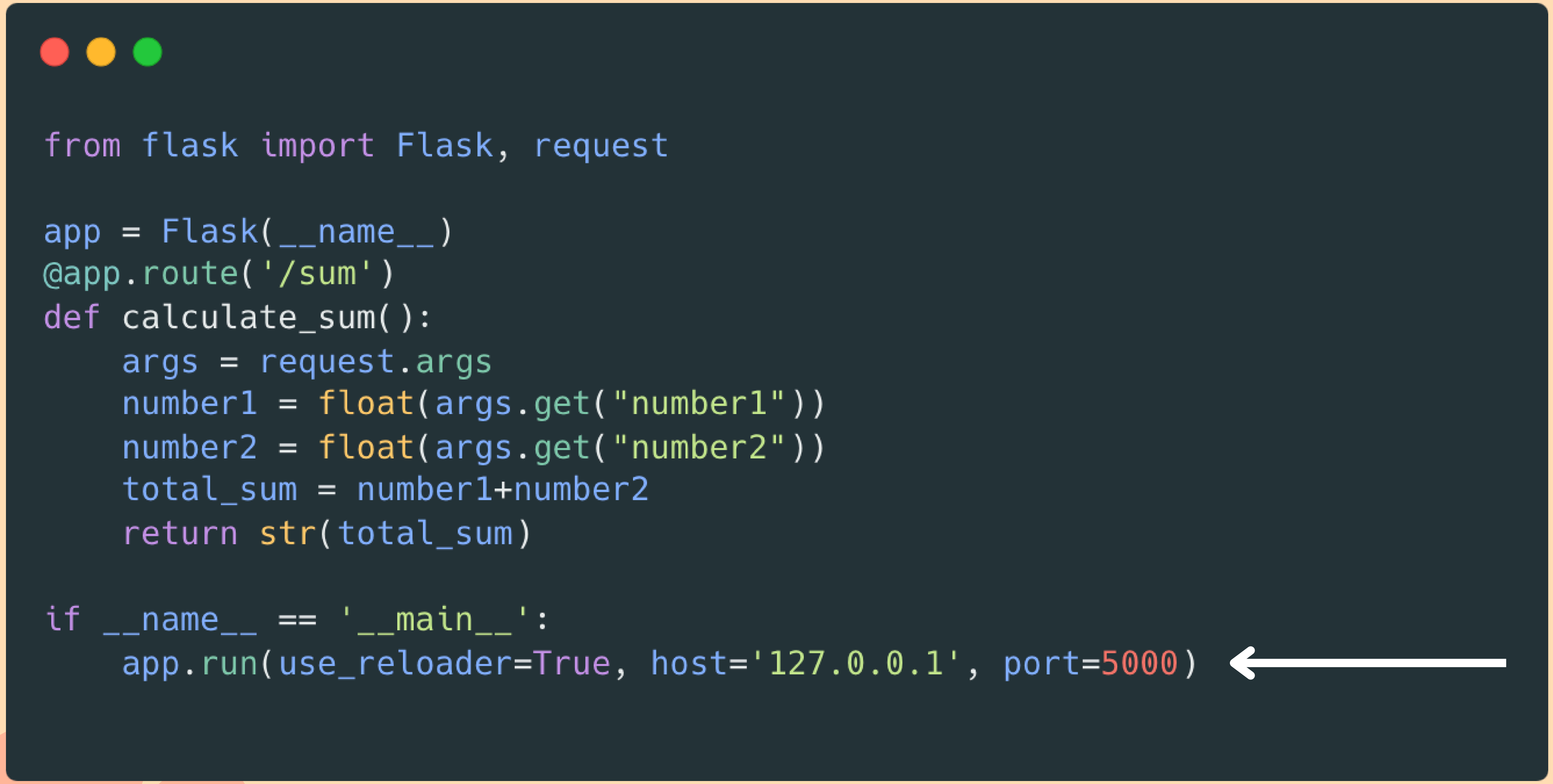
Flask is a lightweight and flexible web framework for Python, designed to make it easy to create web applications and backend services. It is often referred to as a "micro-framework" because it provides only the core tools needed to build web applications, allowing developers to add additional functionality as needed using extensions.

# PROGRAMMING ENDPOINTS



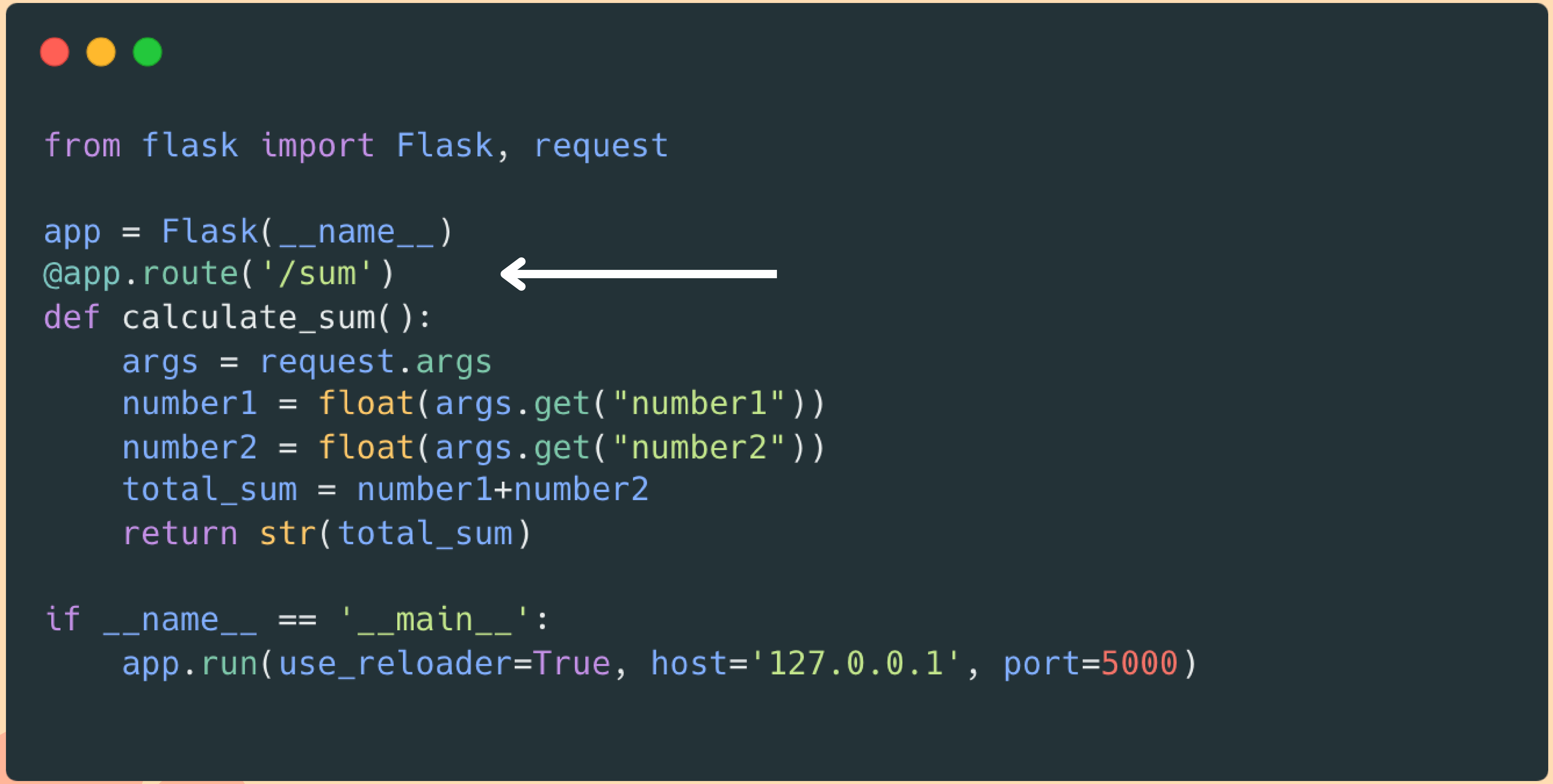
```python
from flask import Flask, request

app = Flask(__name__)
@app.route('/sum')
def calculate_sum():
    args = request.args
    number1 = float(args.get("number1"))
    number2 = float(args.get("number2"))
    total_sum = number1+number2
    return str(total_sum)

if __name__ == '__main__':
    app.run(use_reloader=True, host='127.0.0.1', port=5000)
```

## LAUNCHES THE BACKEND SERVICE



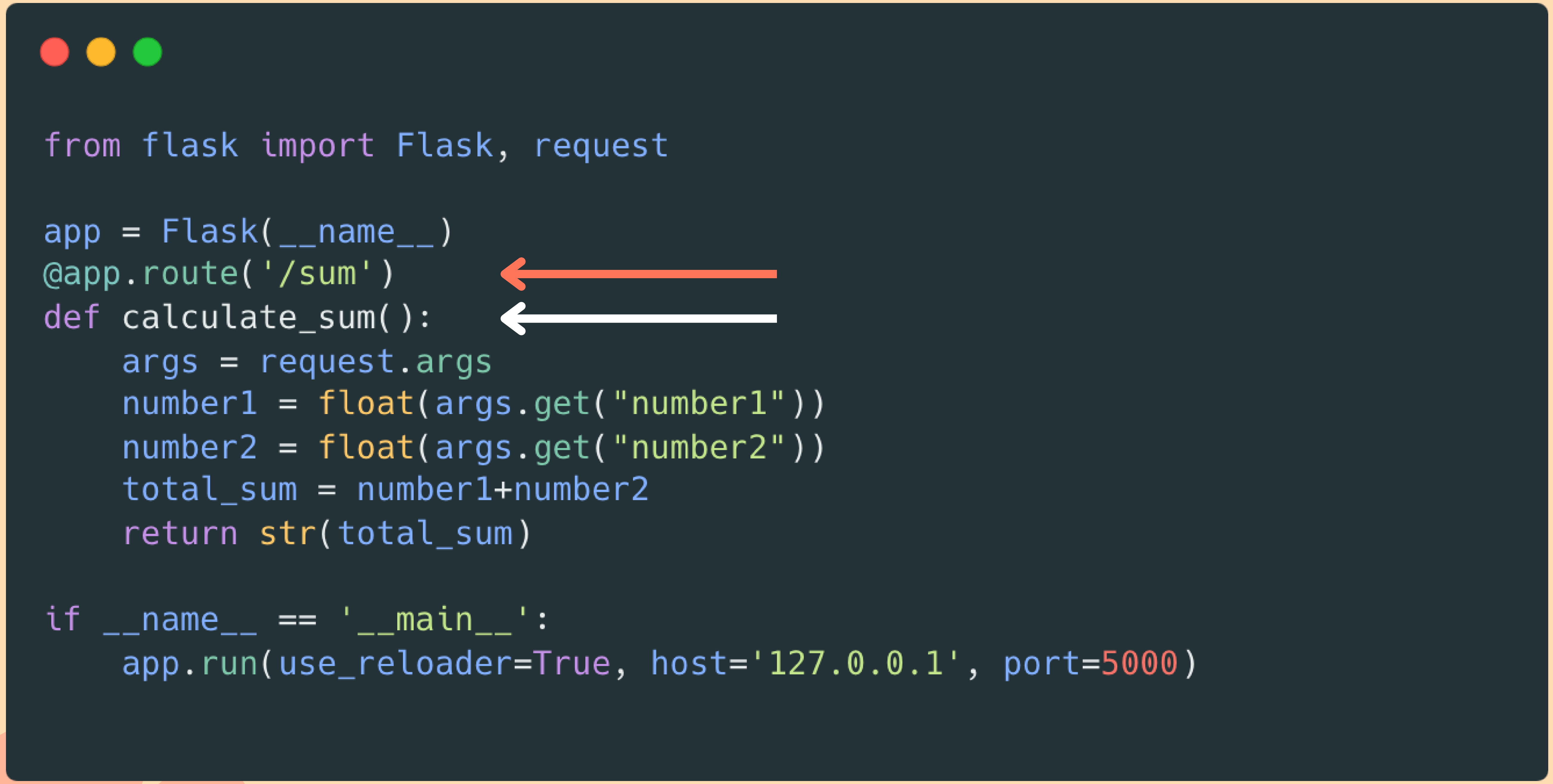
```
from flask import Flask, request

app = Flask(__name__)
@app.route('/sum')
def calculate_sum():
    args = request.args
    number1 = float(args.get("number1"))
    number2 = float(args.get("number2"))
    total_sum = number1+number2
    return str(total_sum)

if __name__ == '__main__':
    app.run(use_reloader=True, host='127.0.0.1', port=5000)
```

# DEFINING ENDPOINTS



```python
from flask import Flask, request

app = Flask(__name__)
@app.route('/sum')
def calculate_sum():
    args = request.args
    number1 = float(args.get("number1"))
    number2 = float(args.get("number2"))
    total_sum = number1+number2
    return str(total_sum)

if __name__ == '__main__':
    app.run(use_reloader=True, host='127.0.0.1', port=5000)
```

# DEFINING ENDPOINTS



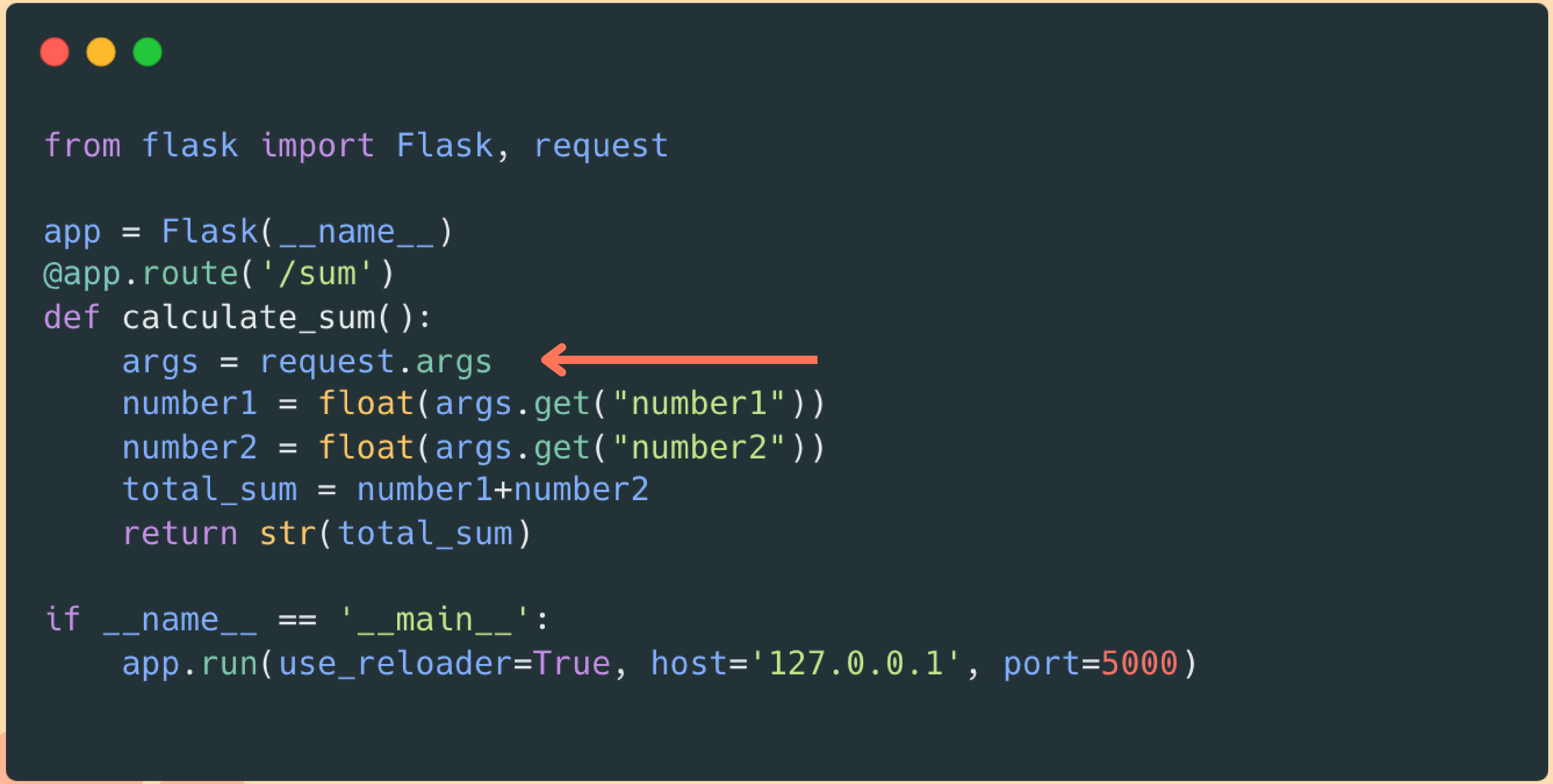
```python
from flask import Flask, request

app = Flask(__name__)
@app.route('/sum')
def calculate_sum():
    args = request.args
    number1 = float(args.get("number1"))
    number2 = float(args.get("number2"))
    total_sum = number1+number2
    return str(total_sum)

if __name__ == '__main__':
    app.run(use_reloader=True, host='127.0.0.1', port=5000)
```

## DEFINING ENDPOINTS

**http://127.0.0.1:5000/sum**

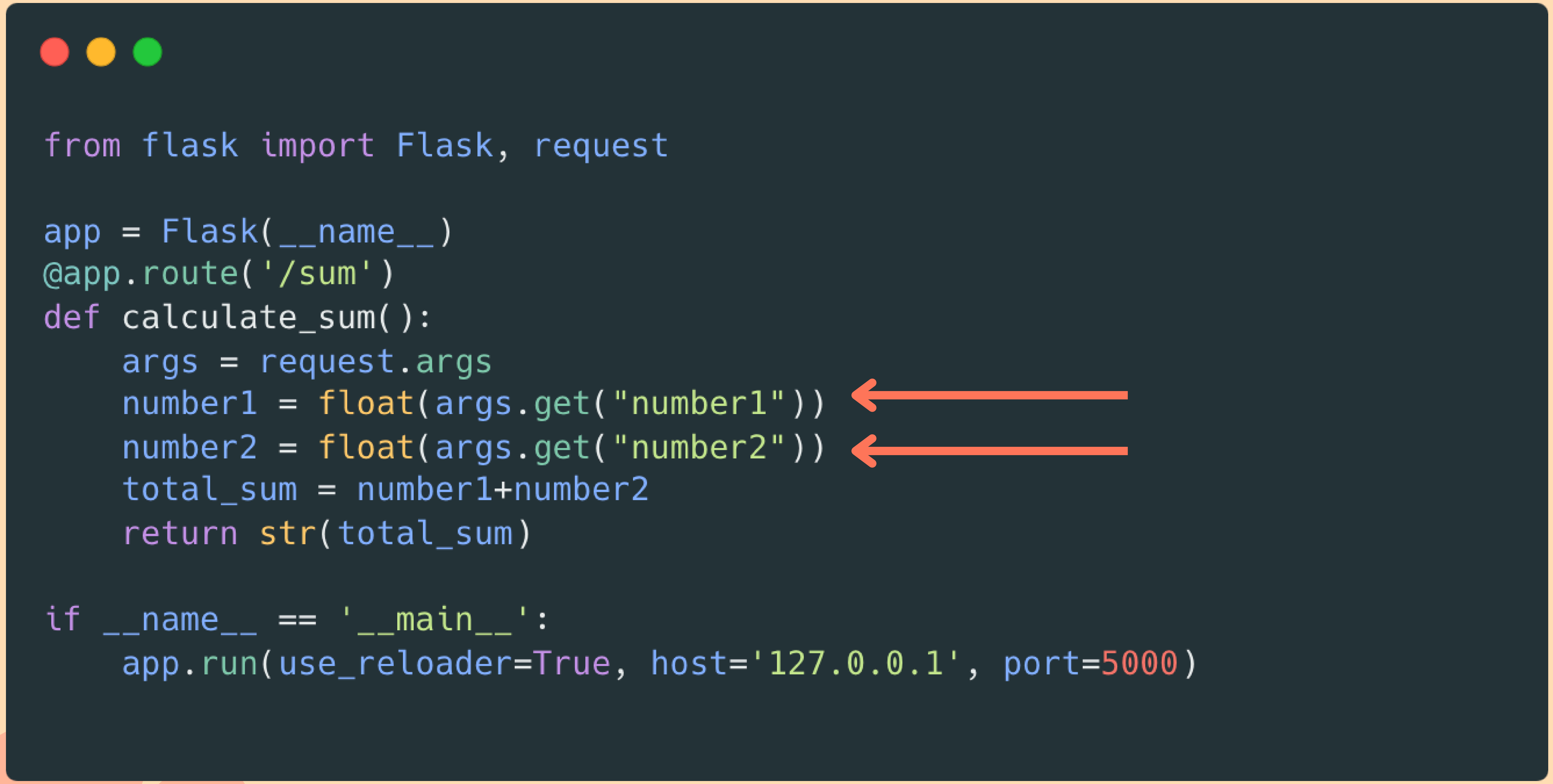
## DEMO

## PARAMETERS OF THE GET REQUEST



```python
from flask import Flask, request

app = Flask(__name__)
@app.route('/sum')
def calculate_sum():
    args = request.args
    number1 = float(args.get("number1"))
    number2 = float(args.get("number2"))
    total_sum = number1+number2
    return str(total_sum)

if __name__ == '__main__':
    app.run(use_reloader=True, host='127.0.0.1', port=5000)
```

## PARAMETERS OF THE GET REQUEST



```python
from flask import Flask, request

app = Flask(__name__)
@app.route('/sum')
def calculate_sum():
    args = request.args
    number1 = float(args.get("number1"))
    number2 = float(args.get("number2"))
    total_sum = number1+number2
    return str(total_sum)

if __name__ == '__main__':
    app.run(use_reloader=True, host='127.0.0.1', port=5000)
```

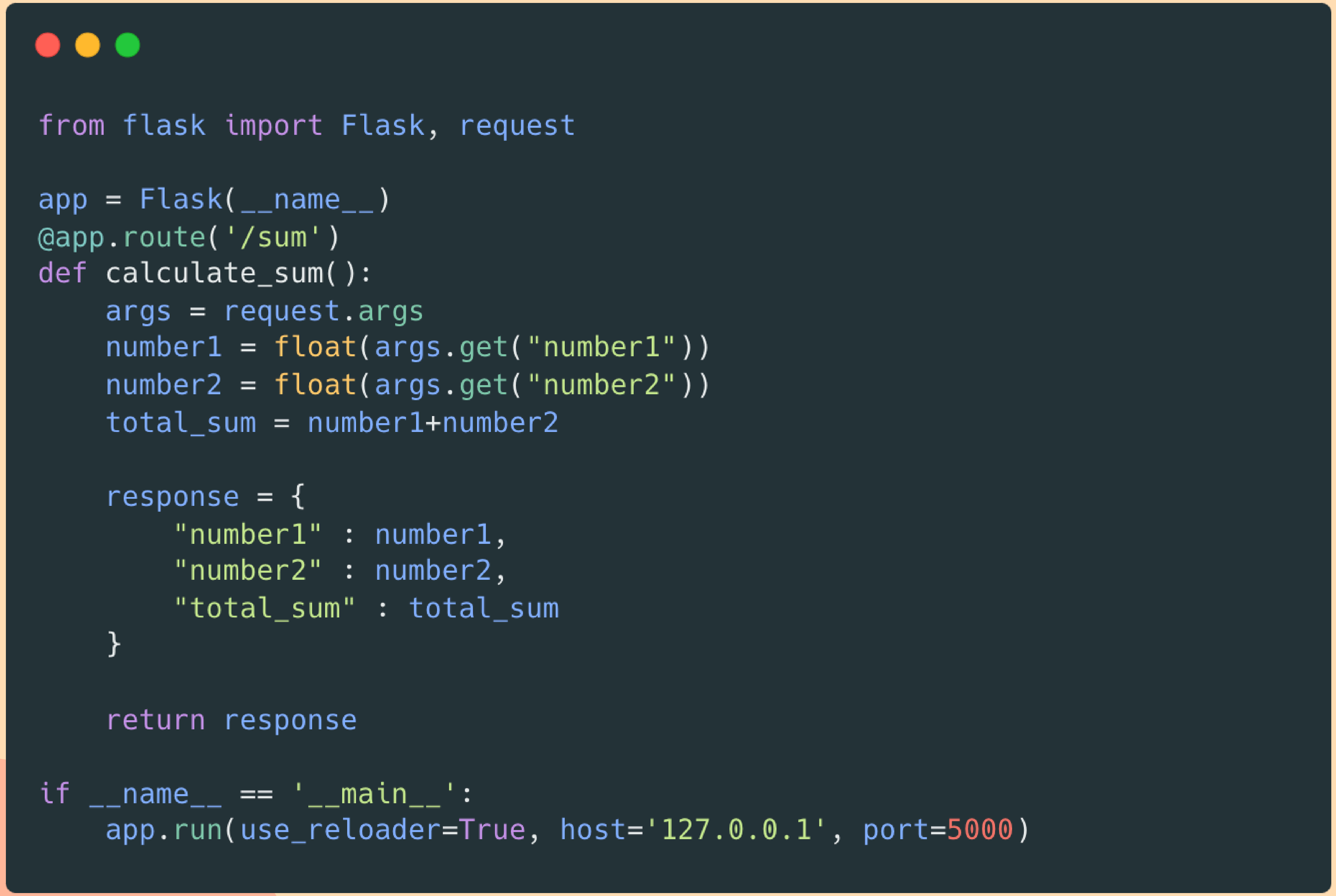
## PARAMETERS OF THE GET REQUEST

http://127.0.0.1:5000/sum?number1=13&number2=28

## DEMO

At this point the backend service technically works, but the format of the result is not optimal to be processed programmatically.

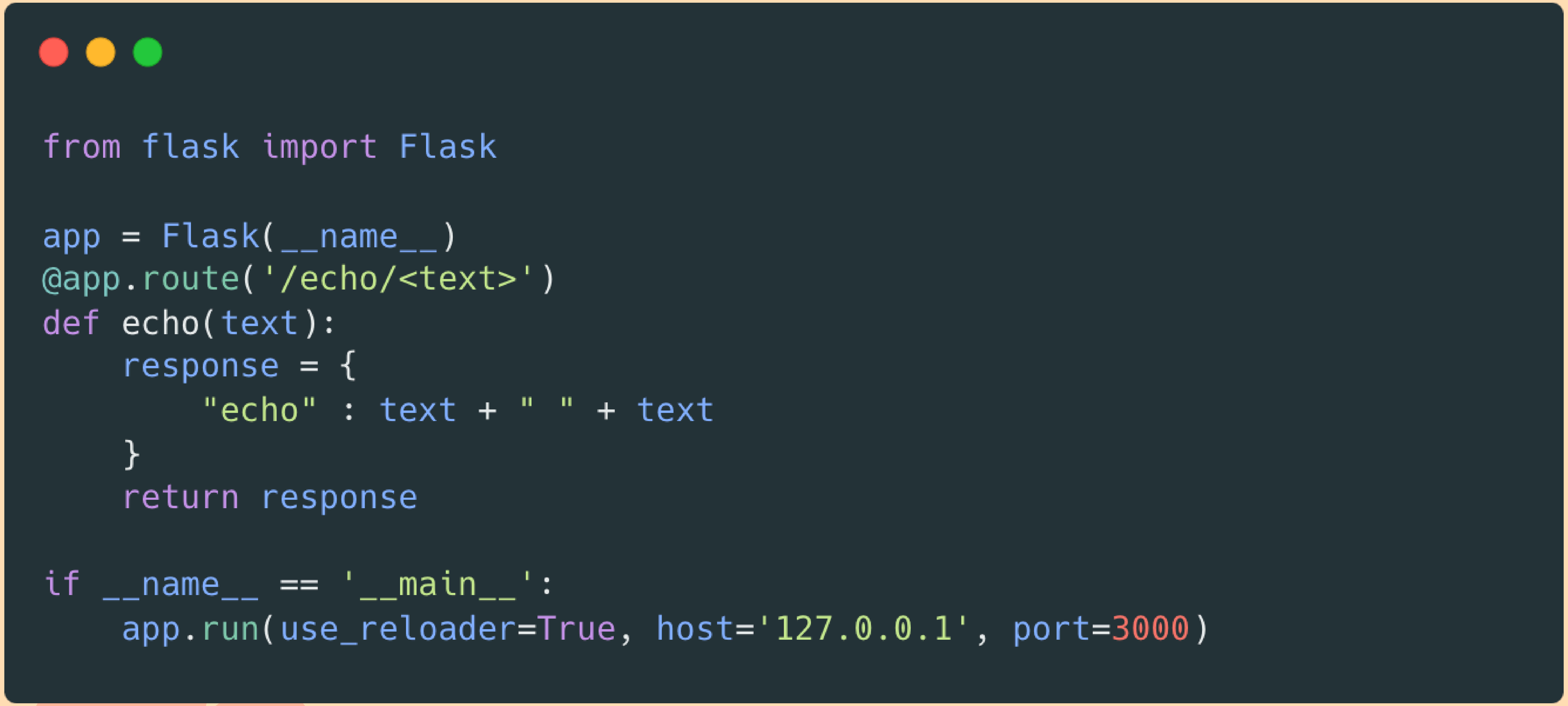
# GENERATING A JSON RESPONSE



```python
from flask import Flask, request

app = Flask(__name__)
@app.route('/sum')
def calculate_sum():
    args = request.args
    number1 = float(args.get("number1"))
    number2 = float(args.get("number2"))
    total_sum = number1+number2

    response = {
        "number1" : number1,
        "number2" : number2,
        "total_sum" : total_sum
    }

    return response

if __name__ == '__main__':
    app.run(use_reloader=True, host='127.0.0.1', port=5000)
```

## DEMO

# PARSING THE REQUEST



```python
from flask import Flask

app = Flask(__name__)
@app.route('/echo/<text>')
def echo(text):
    response = {
        "echo" : text + " " + text
    }
    return response

if __name__ == '__main__':
    app.run(use_reloader=True, host='127.0.0.1', port=3000)
```

## DEMO

# ERROR SCENARIO 1

Example: http://127.0.0.1:3000/dum/42/117

- Status Code: 404 (Not Found)

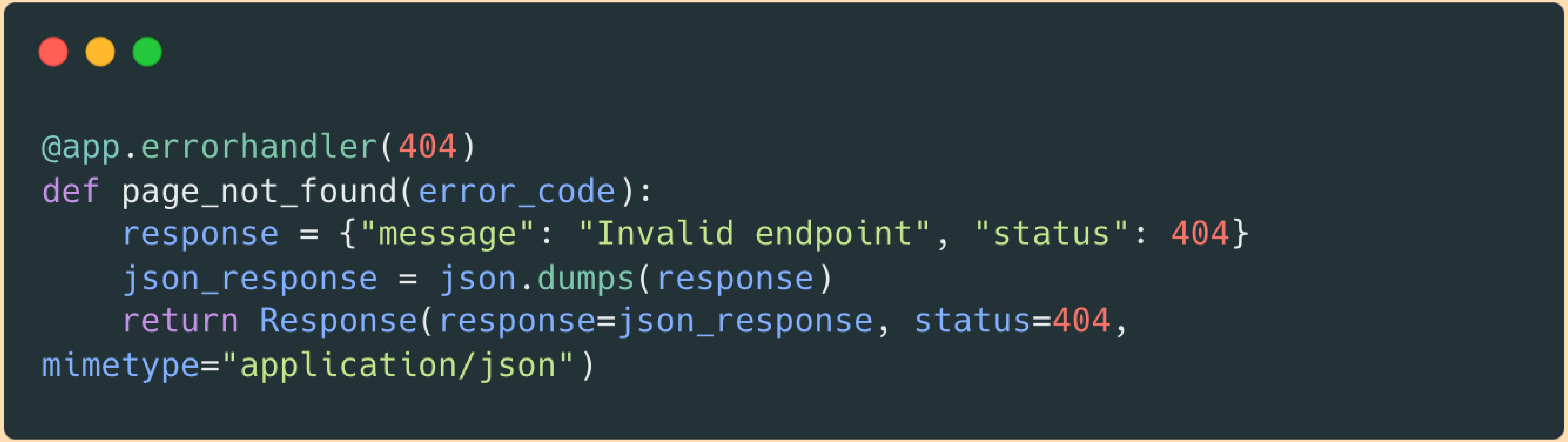
## ERROR SCENARIO 2

**Example: http://127.0.0.1:3000/sum/4t23/117**

- **Status Code: 400 (Bad Request)**

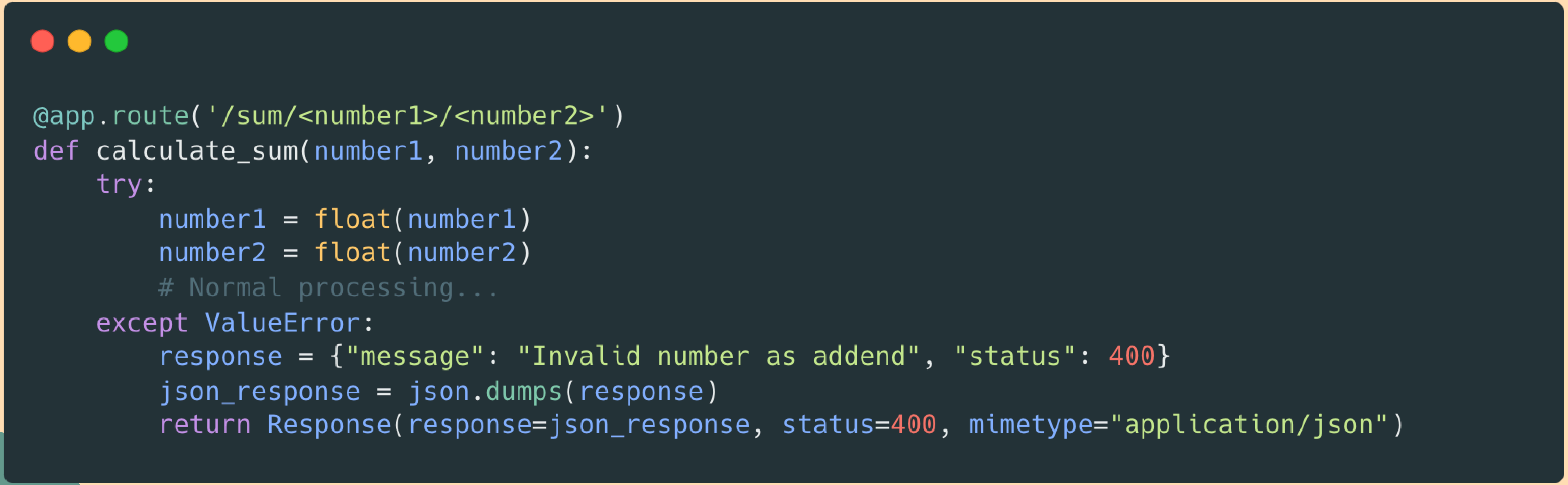
## WE CAN CREATE CUSTOMIZED ERROR RESPONSES

## CUSTOMIZING 404 ERRORS



```python
@app.errorhandler(404)
def page_not_found(error_code):
    response = {"message": "Invalid endpoint", "status": 404}
    json_response = json.dumps(response)
    return Response(response=json_response, status=404,
mimetype="application/json")
```

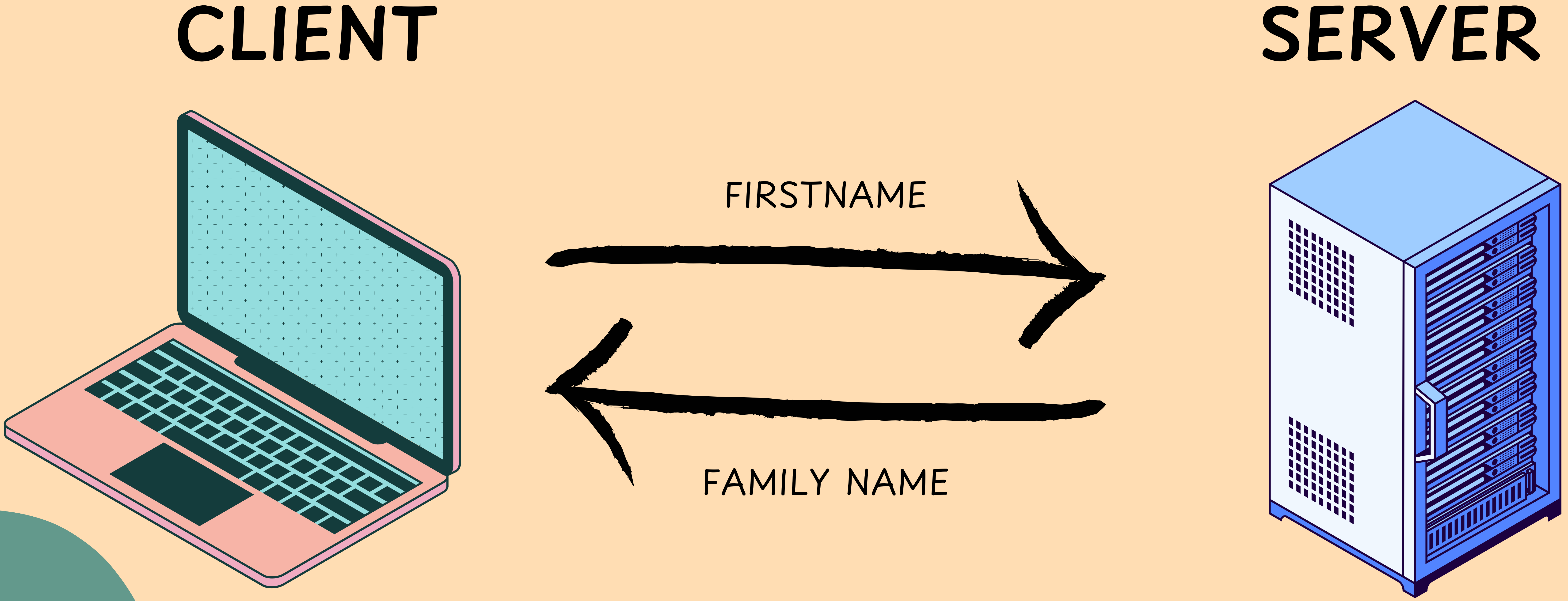
## HANDLING INVALID INPUTS



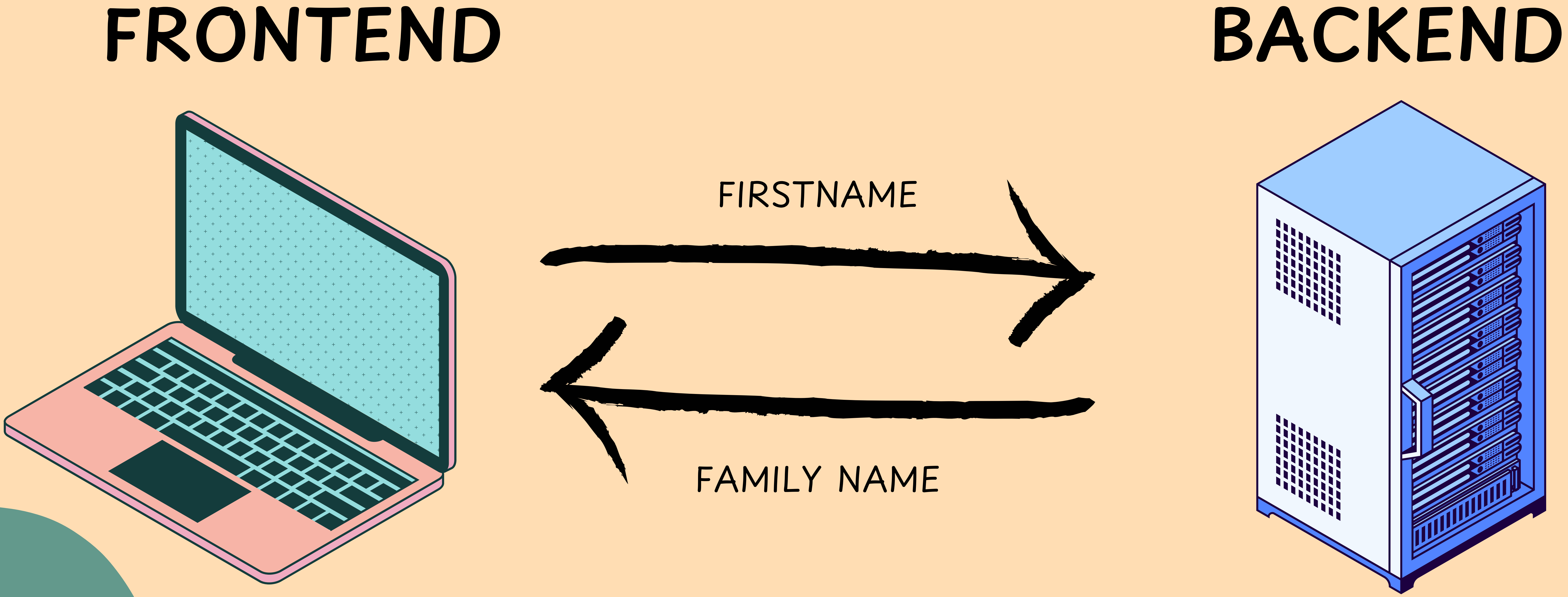
```python
@app.route('/sum/<number1>/<number2>')
def calculate_sum(number1, number2):
    try:
        number1 = float(number1)
        number2 = float(number2)
        # Normal processing...
    except ValueError:
        response = {"message": "Invalid number as addend", "status": 400}
        json_response = json.dumps(response)
        return Response(response=json_response, status=400, mimetype="application/json")
```

## LET’S BUILD BACKEND TOGETHER

## CLIENT - SERVER MODEL



# CLIENT - SERVER MODEL


FRONTEND
BACKEND
FIRSTNAME
FAMILY NAME