FastAPI Slide Report

Lê Việt Hưng - AI Intern

Content

1 FastAPI Swagger

2 FastAPI authentication JWT

3 Log

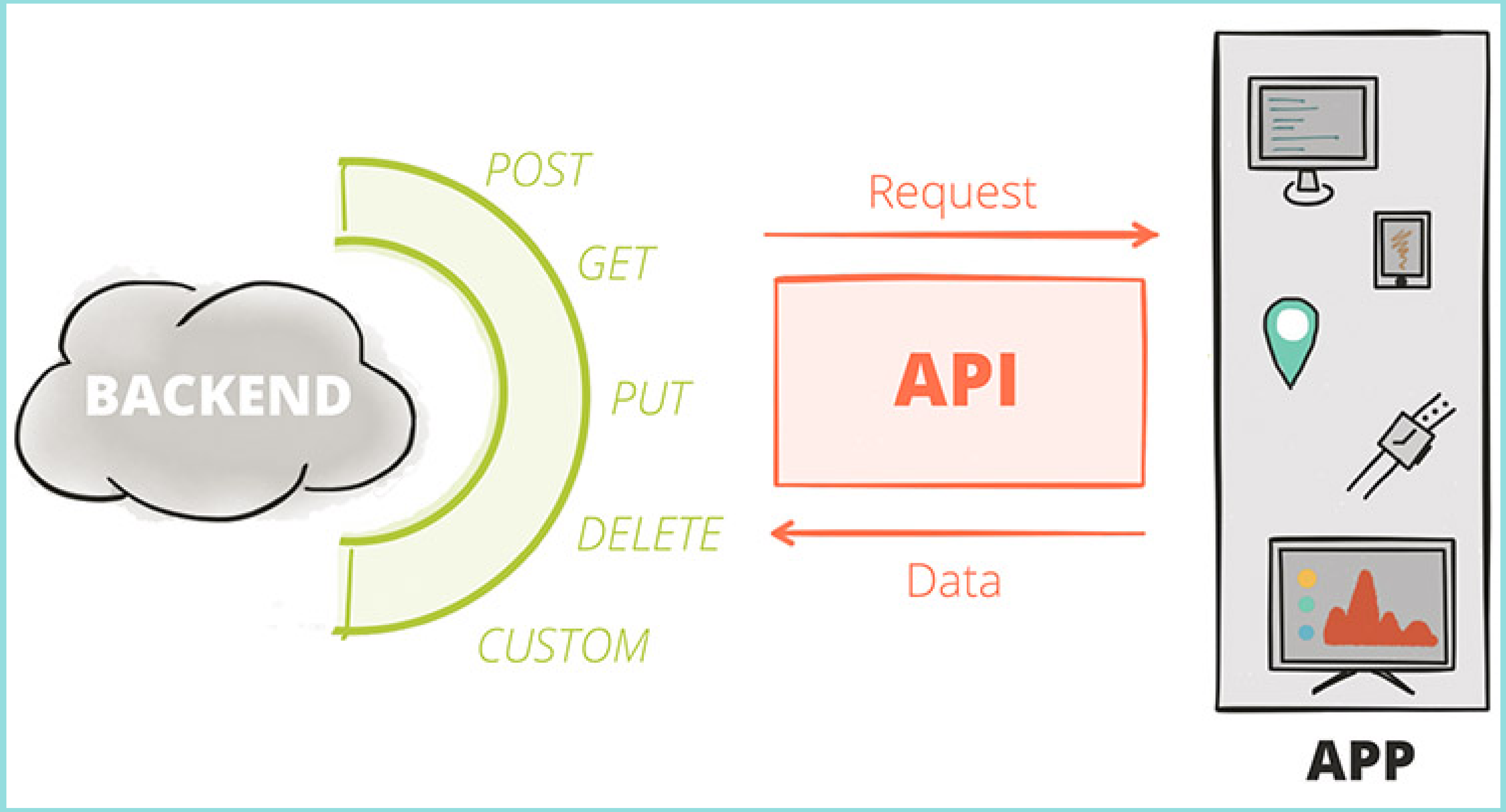
4 Uvicorn

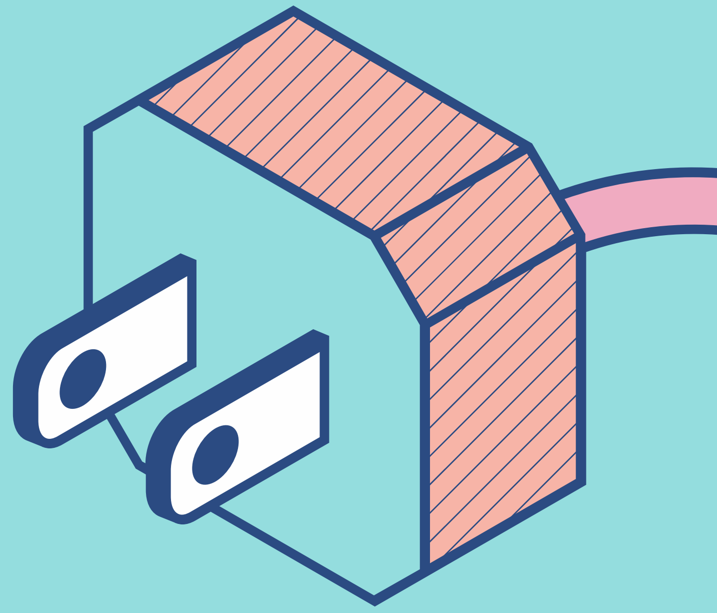
5 Pylint

6 Pytest

1.FastAPI Swagger

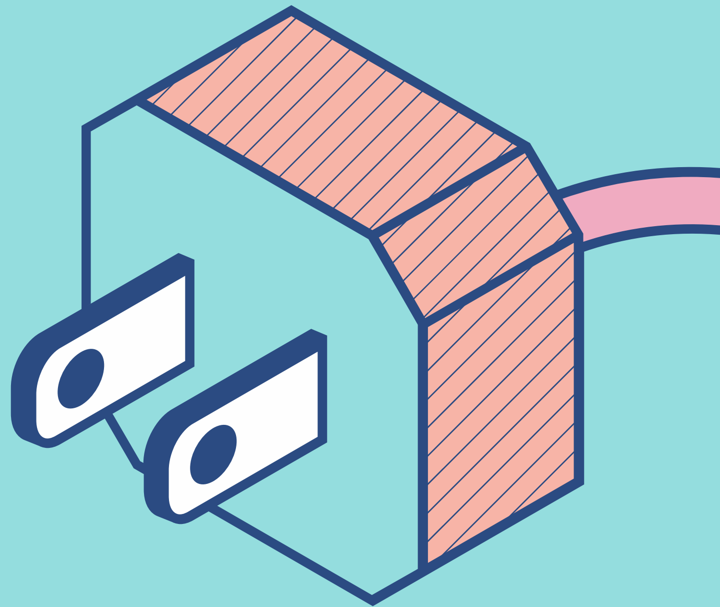
FastAPI is a web framework used to build high-performance APIs with ease, simplicity, and strong support for product development.





1.FastAPI Swagger

Swagger is a set of open-source tools and specifications for designing, building, and documenting APIs.

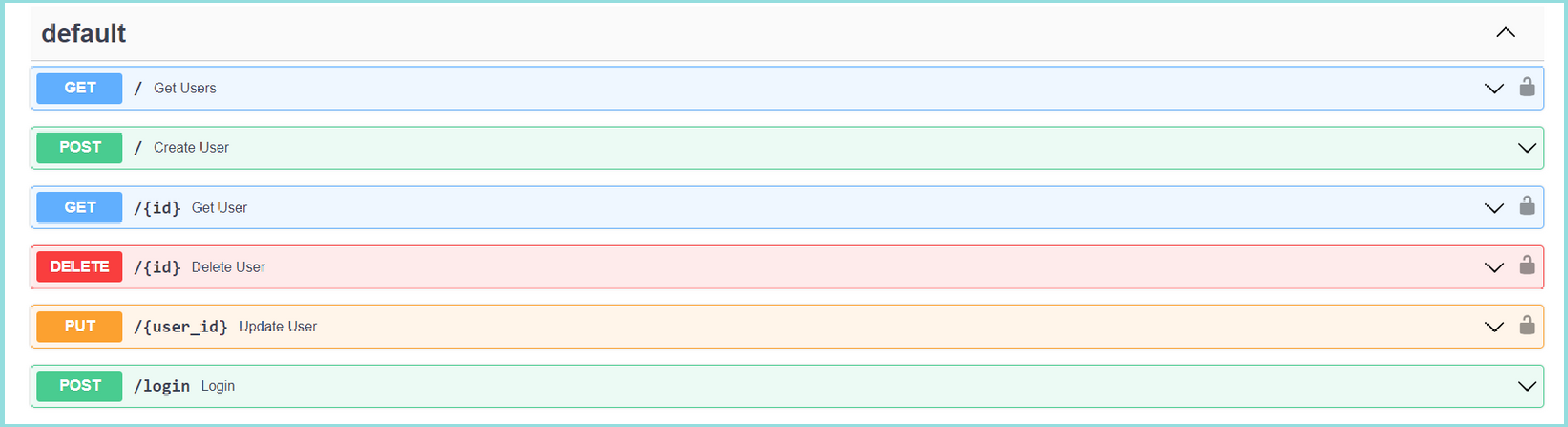


1.FastAPI Swagger

Key functions of Swagger:

- API Documentation
- Interactive API Exploration (Swagger UI)

Swagger UI:



default

GET / Get Users

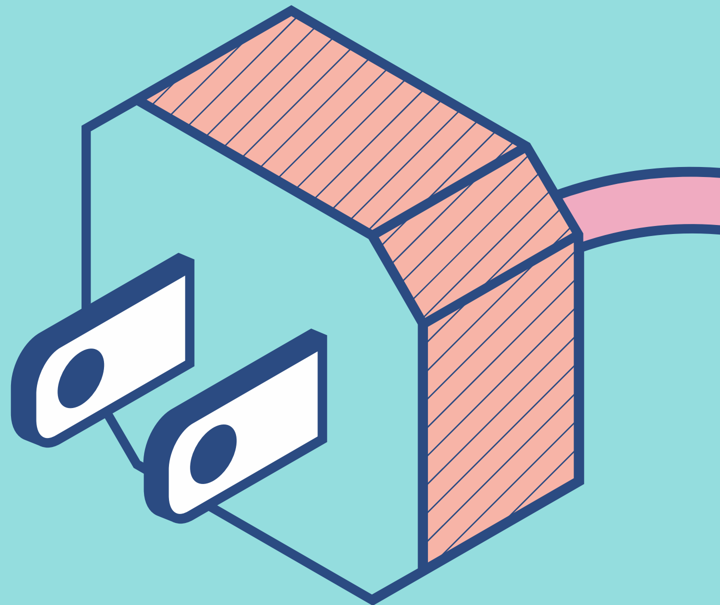
POST / Create User

GET /{id} Get User

DELETE /{id} Delete User

PUT /{user_id} Update User

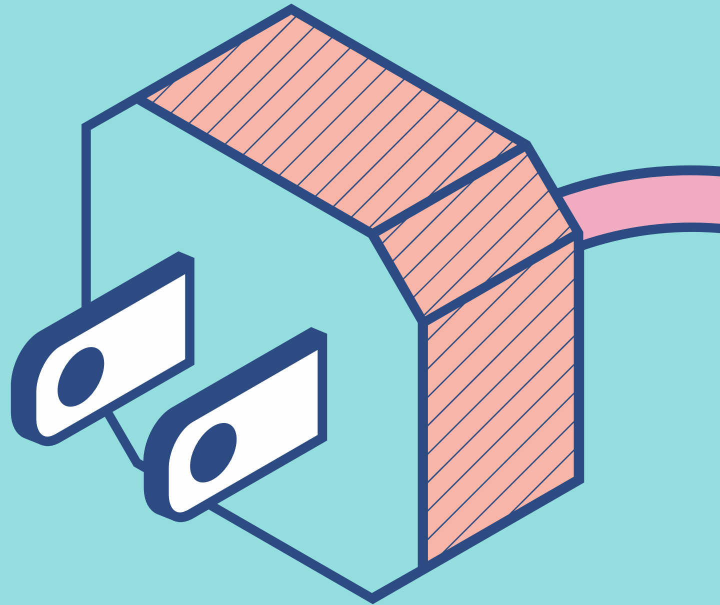
POST /login Login



2.FastAPI authentication JWT

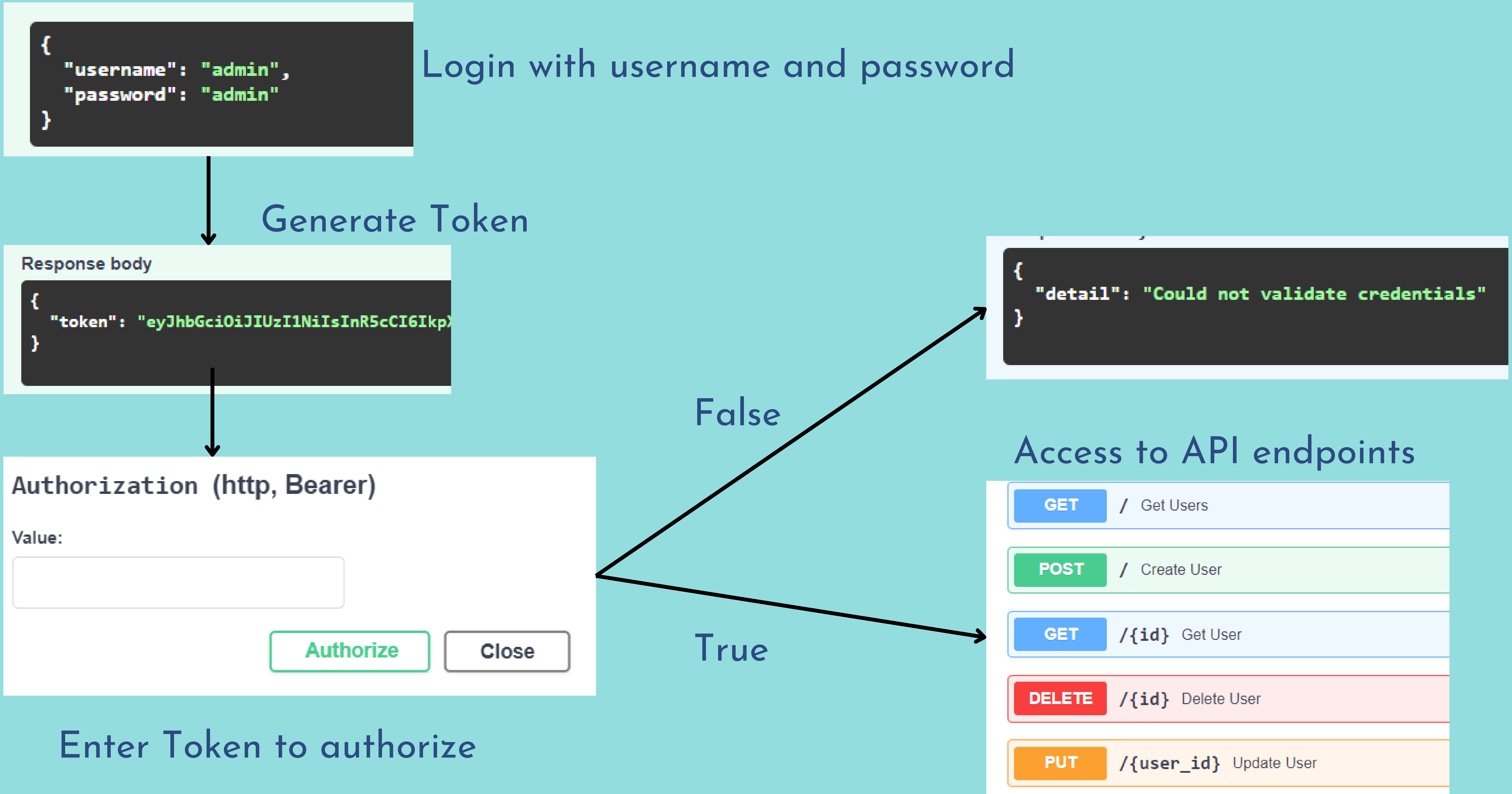
JWT (JSON Web Token) is a URL-safe means of representing claims to be transferred between two parties. These claims can be used for various purposes, including authentication and data exchange. JWTs are commonly used in **authentication** and authorization processes for securing web applications and APIs.

FastAPI's authentication JWT simplifies the process of securing your API endpoints by integrating JWT-based authentication.



2.FastAPI authentication JWT

Example: Login using authentication JWT



```
{
  "username": "admin",
  "password": "admin"
}
```

Login with username and password

Generate Token

Response body

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp
}
```

Authorization (http, Bearer)

Value:

Enter Token to authorize

False

```
{
  "detail": "Could not validate credentials"
}
```

True

Access to API endpoints

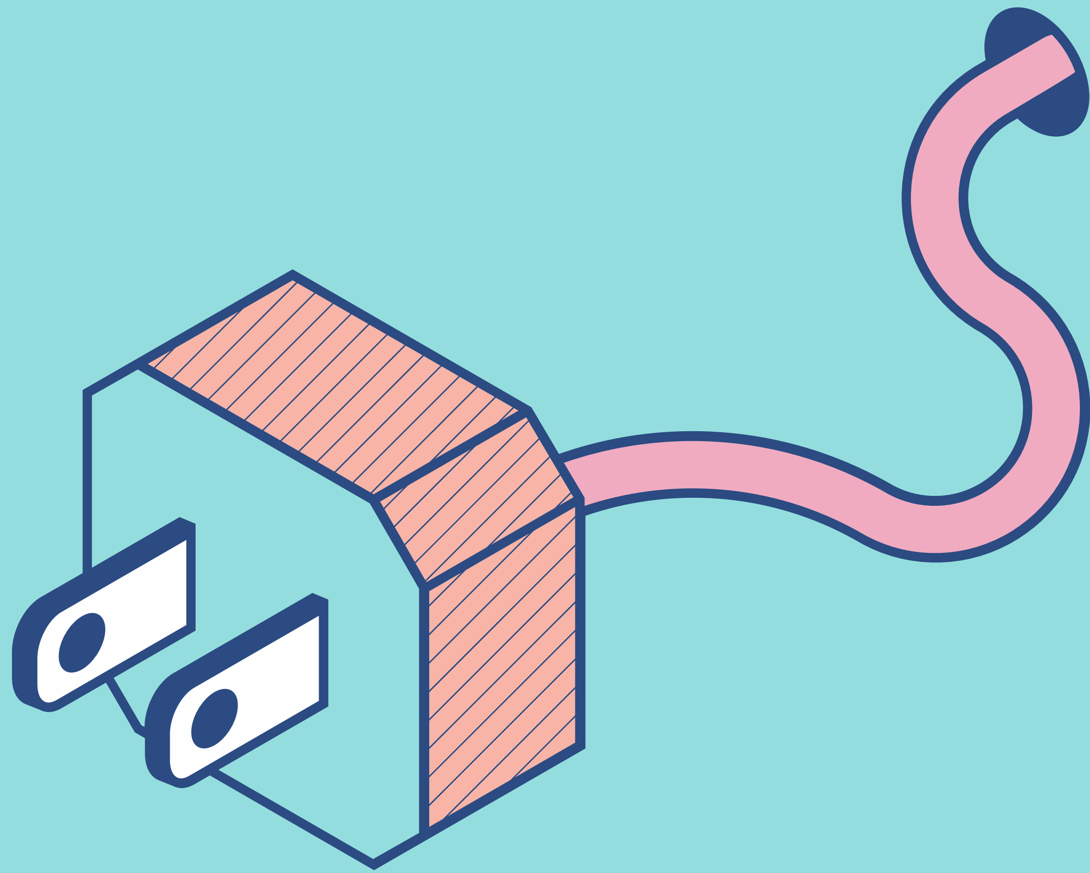
- GET / Get Users
- POST / Create User
- GET /{id} Get User
- DELETE /{id} Delete User
- PUT /{user_id} Update User

3.Logger

Logger is a tool that allows you to capture and manage log messages generated by your application.

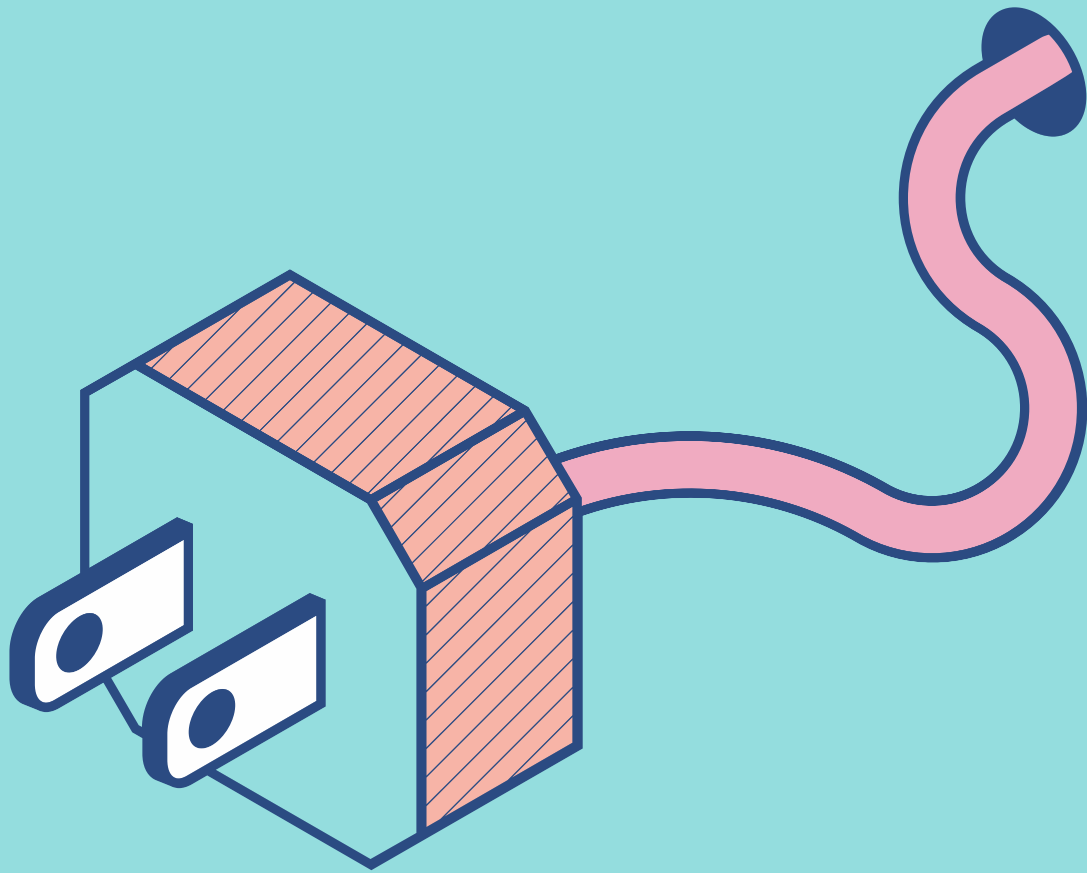
Log messages are categorized by four types:

- Debug
- Information
- Error



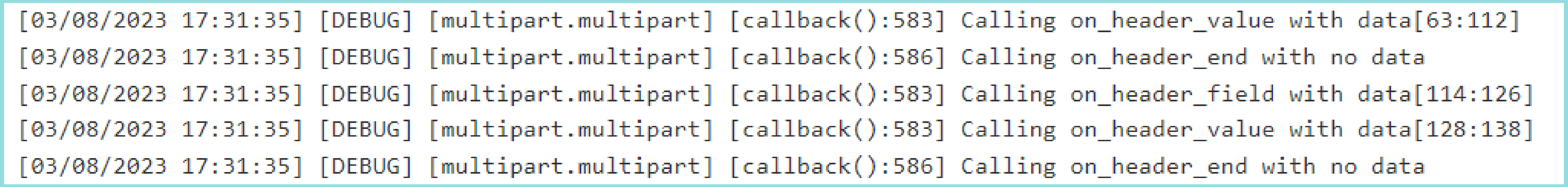
3.Logger

- **Middleware:** allows you to intercept and process requests and responses. It's often useful to log information about these interactions

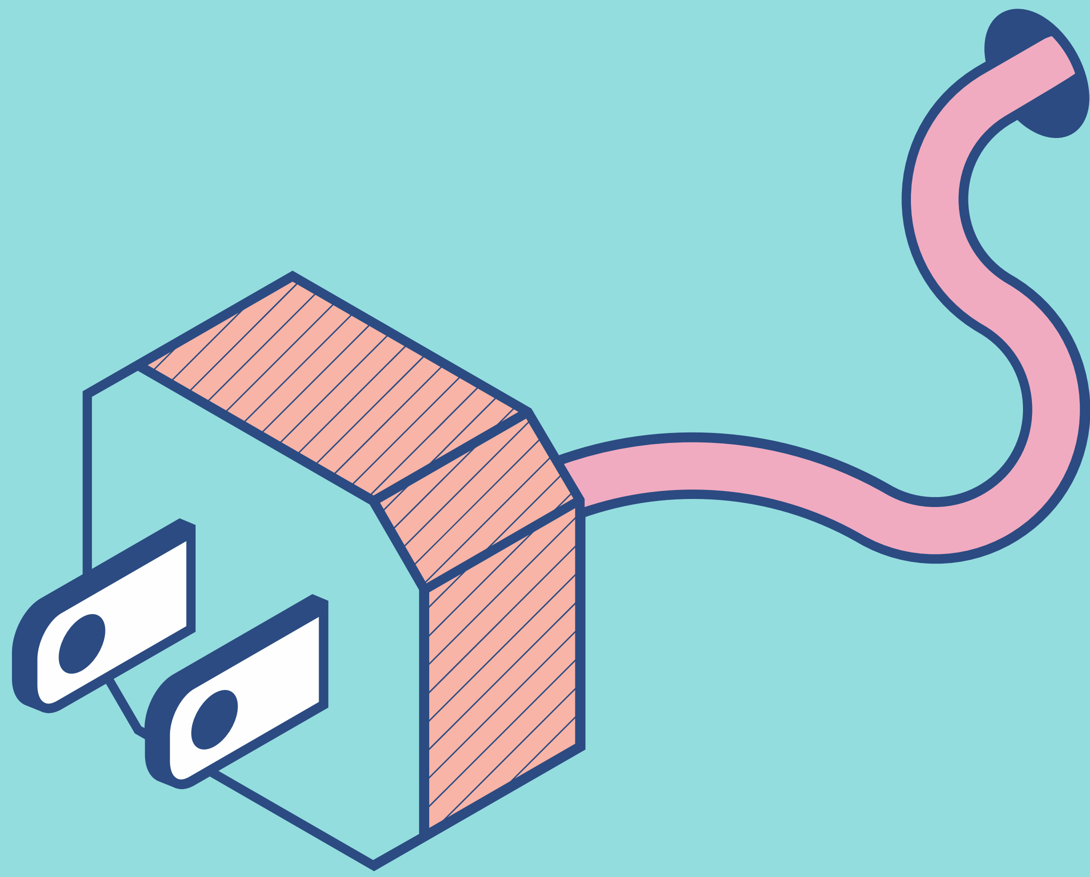


3.Logger

- **Debug:** diagnosing issues and understanding the flow of your application's execution
- **Example:**



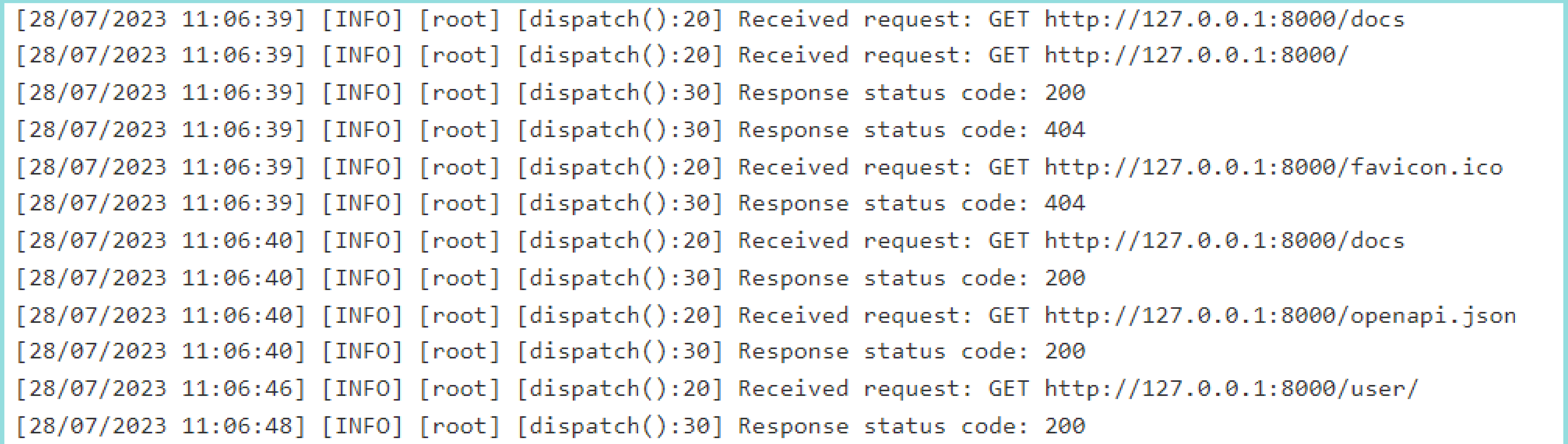
```
[03/08/2023 17:31:35] [DEBUG] [multipart.multipart] [callback():583] Calling on_header_value with data[63:112]
[03/08/2023 17:31:35] [DEBUG] [multipart.multipart] [callback():586] Calling on_header_end with no data
[03/08/2023 17:31:35] [DEBUG] [multipart.multipart] [callback():583] Calling on_header_field with data[114:126]
[03/08/2023 17:31:35] [DEBUG] [multipart.multipart] [callback():583] Calling on_header_value with data[128:138]
[03/08/2023 17:31:35] [DEBUG] [multipart.multipart] [callback():586] Calling on_header_end with no data
```



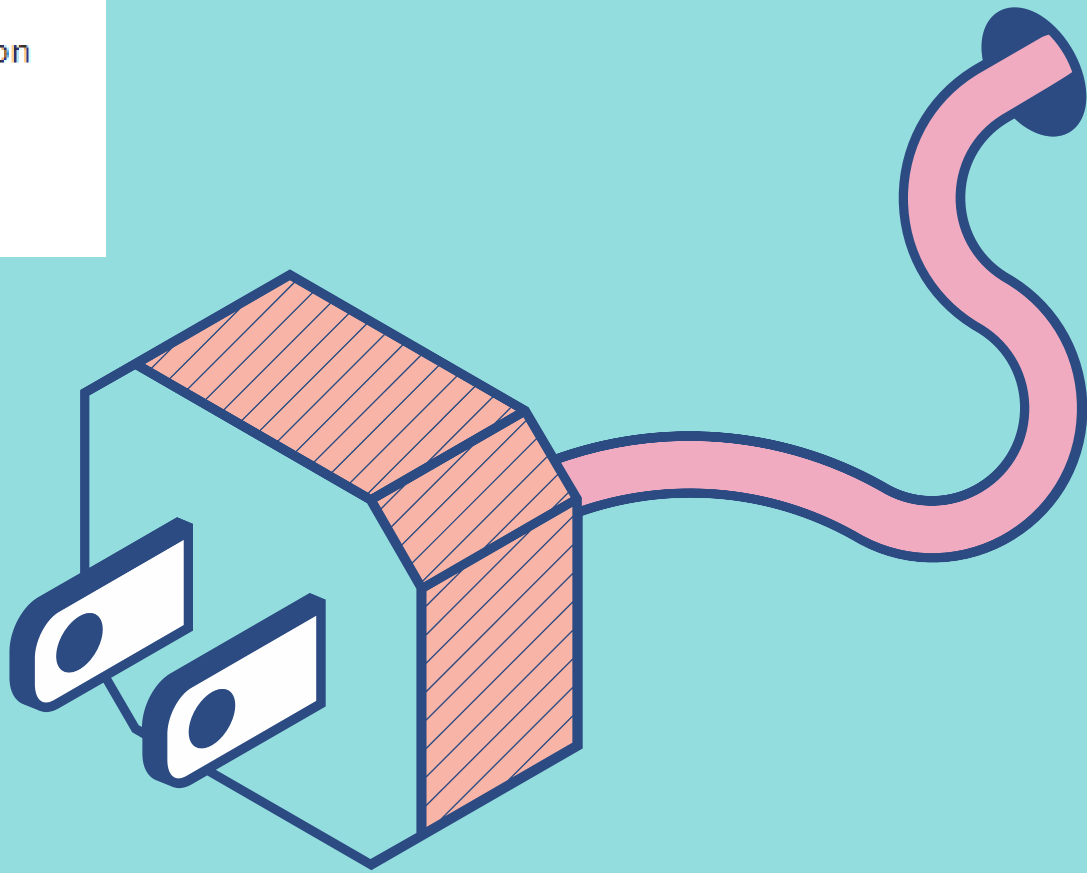
3.Logger

- **Infor:** provide high-level insights into the behavior of your application

- **Example:**

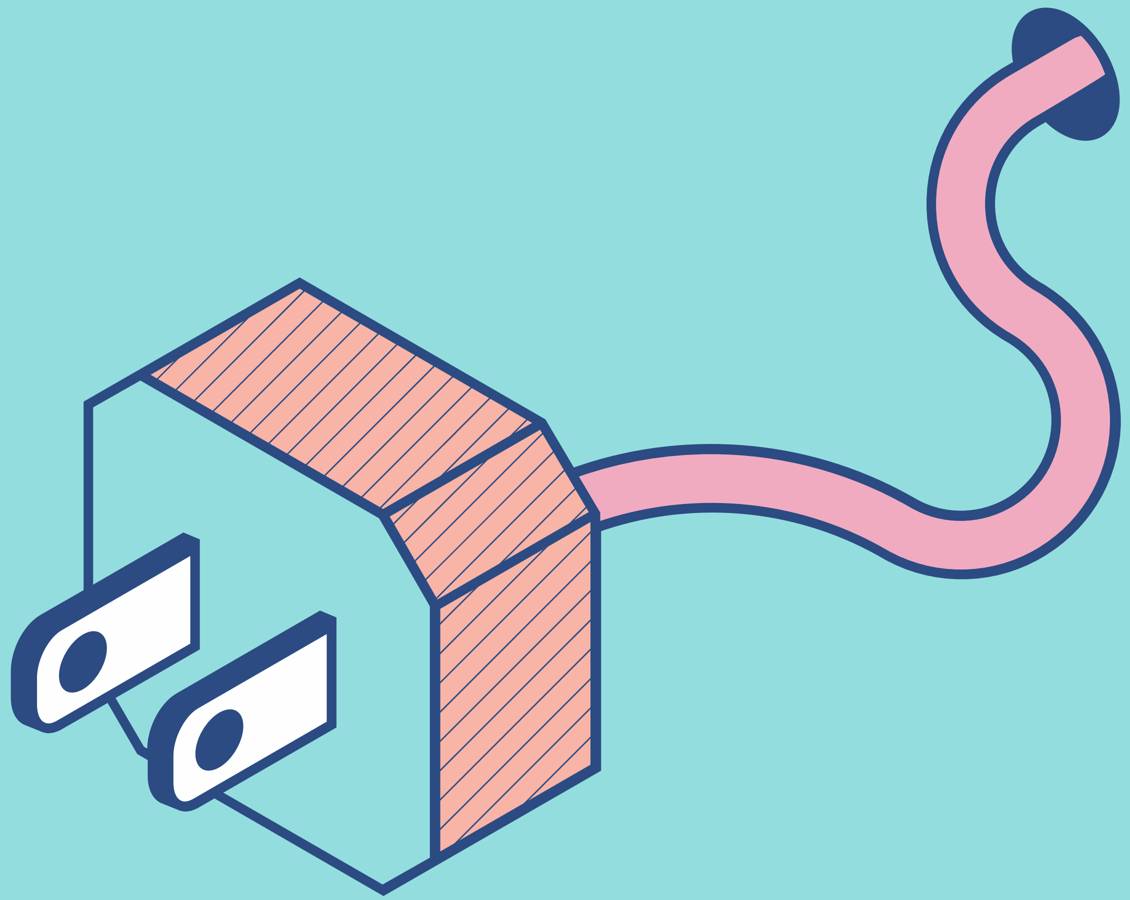


```
[28/07/2023 11:06:39] [INFO] [root] [dispatch():20] Received request: GET http://127.0.0.1:8000/docs
[28/07/2023 11:06:39] [INFO] [root] [dispatch():20] Received request: GET http://127.0.0.1:8000/
[28/07/2023 11:06:39] [INFO] [root] [dispatch():30] Response status code: 200
[28/07/2023 11:06:39] [INFO] [root] [dispatch():30] Response status code: 404
[28/07/2023 11:06:39] [INFO] [root] [dispatch():20] Received request: GET http://127.0.0.1:8000/favicon.ico
[28/07/2023 11:06:39] [INFO] [root] [dispatch():30] Response status code: 404
[28/07/2023 11:06:40] [INFO] [root] [dispatch():20] Received request: GET http://127.0.0.1:8000/docs
[28/07/2023 11:06:40] [INFO] [root] [dispatch():30] Response status code: 200
[28/07/2023 11:06:40] [INFO] [root] [dispatch():20] Received request: GET http://127.0.0.1:8000/openapi.json
[28/07/2023 11:06:40] [INFO] [root] [dispatch():30] Response status code: 200
[28/07/2023 11:06:46] [INFO] [root] [dispatch():20] Received request: GET http://127.0.0.1:8000/user/
[28/07/2023 11:06:48] [INFO] [root] [dispatch():30] Response status code: 200
```



3.Logger

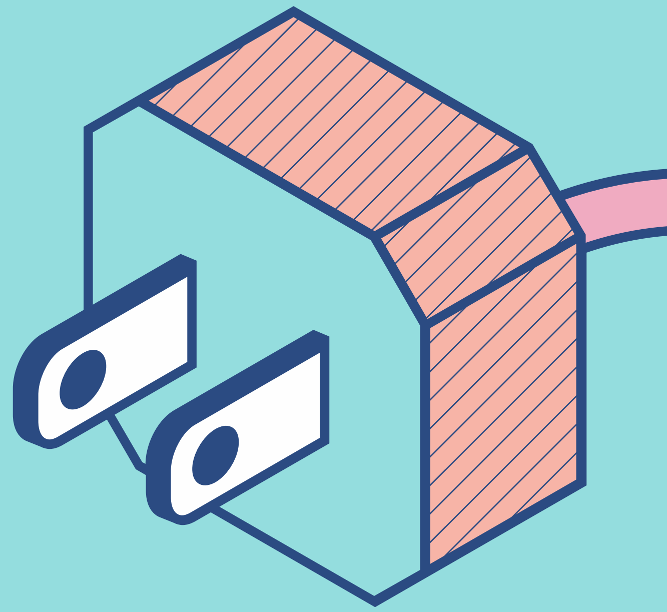
- **Error:** helps you identify and address issues within your application



4.Uvicorn

Uvicorn is an ASGI (Asynchronous Server Gateway Interface) web server for Python.

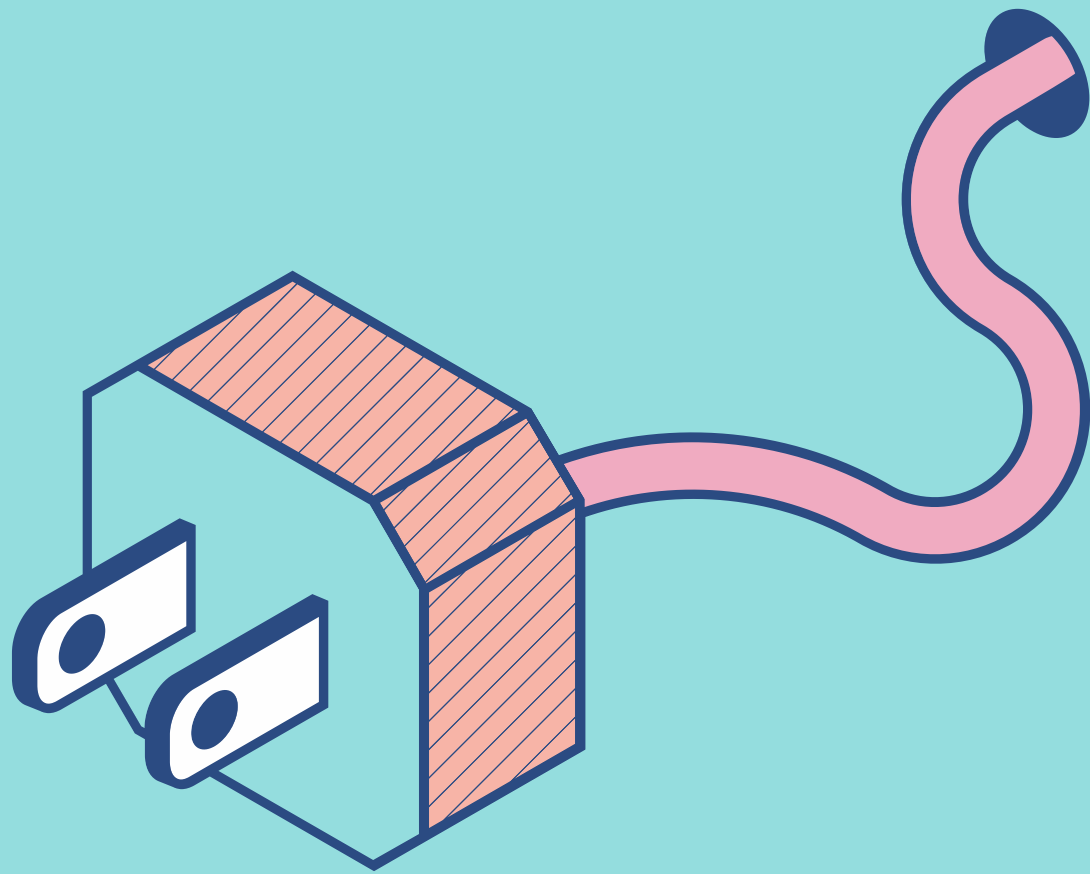
Uvicorn specifically focuses on serving ASGI applications and is optimized for performance.



4.Uvicorn

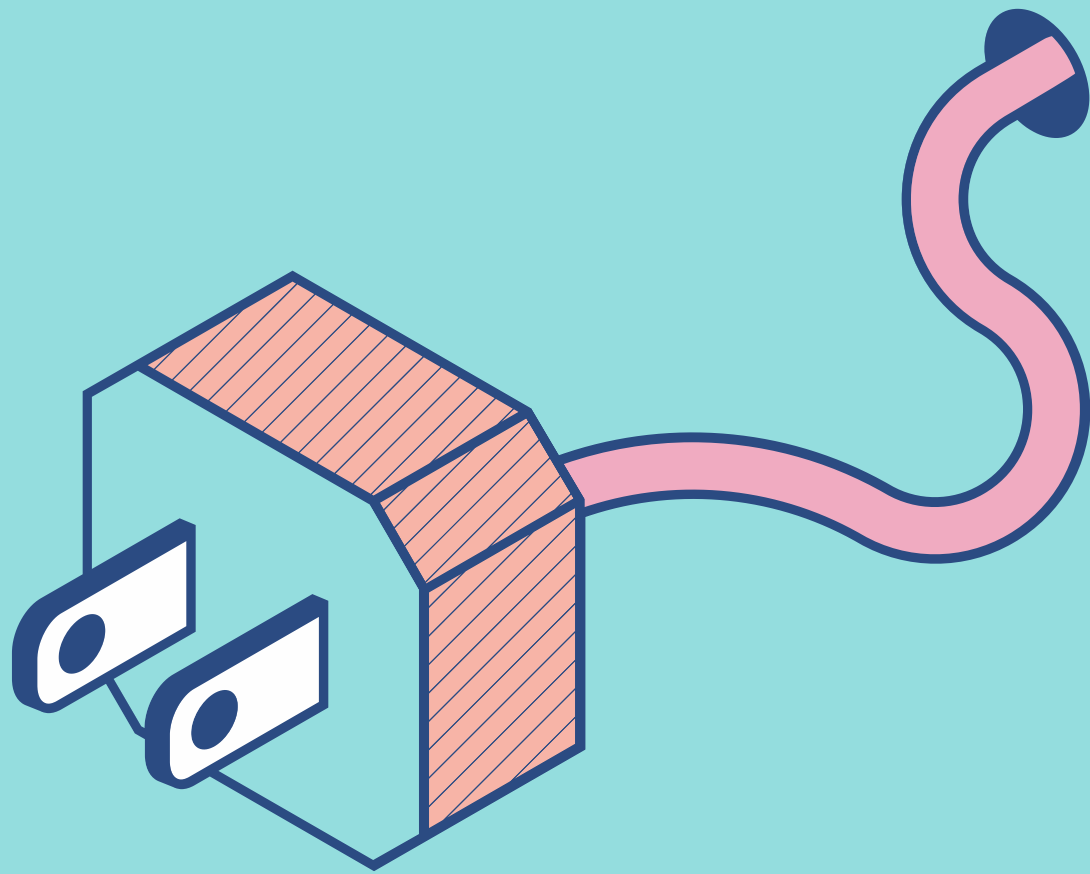
Uvicorn's performance:

- A large number of simultaneous connections
- Be well-suited for applications that require real-time communication,
- Long-lived connections
- Asynchronous processing



5.Pylint

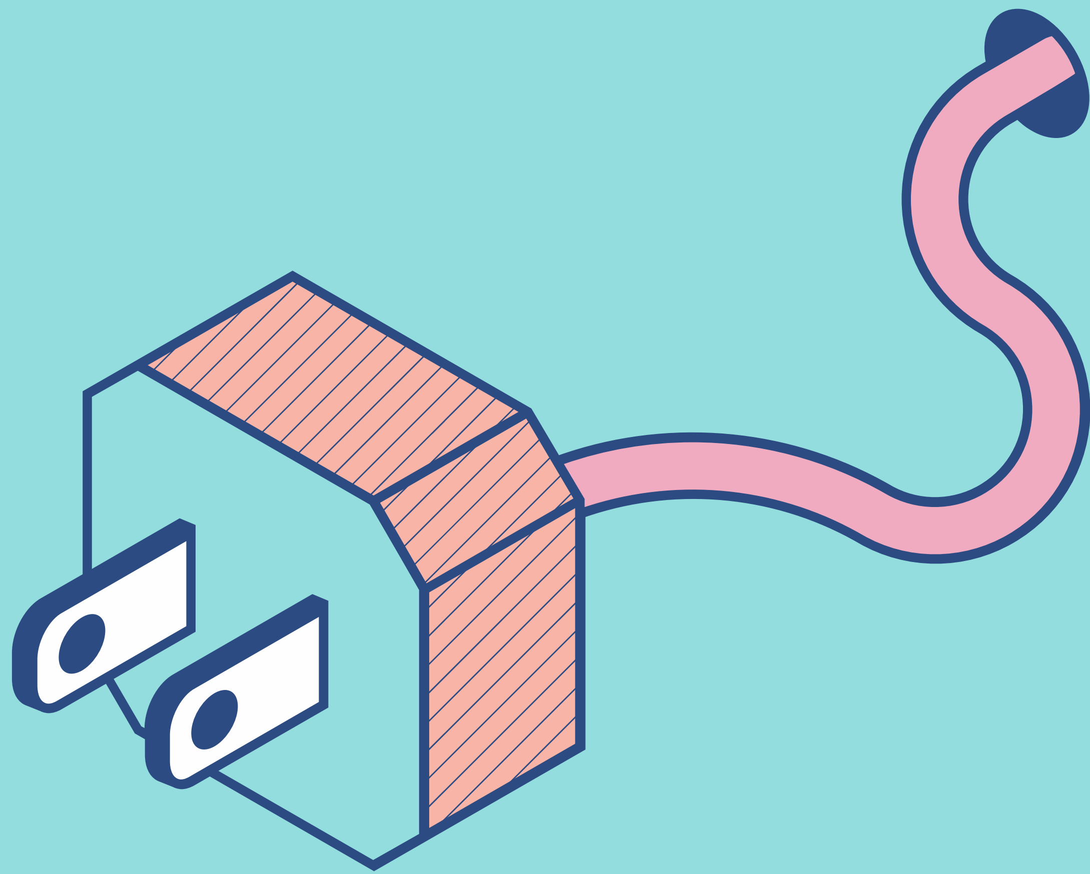
Pylint is a widely used static code analysis tool for the Python programming language. It's designed to help developers identify and correct various coding issues, style inconsistencies, and potential errors in their Python code



5.Pylint

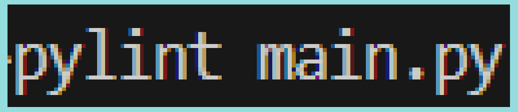
Pylint checks for:

- Coding Style
- Code complexity
- Potential errors
- Code duplication
- Documentation
- Consistency

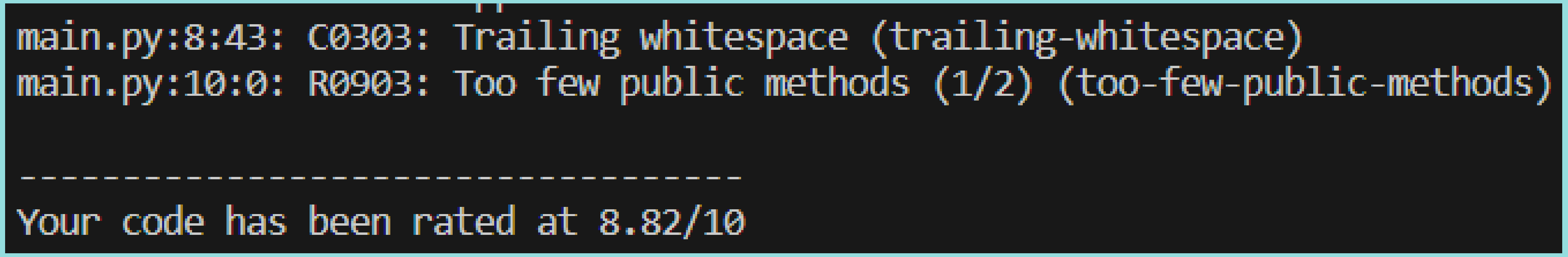


5.Pylint

Example: Rate the python code main.py on scaled rate 10

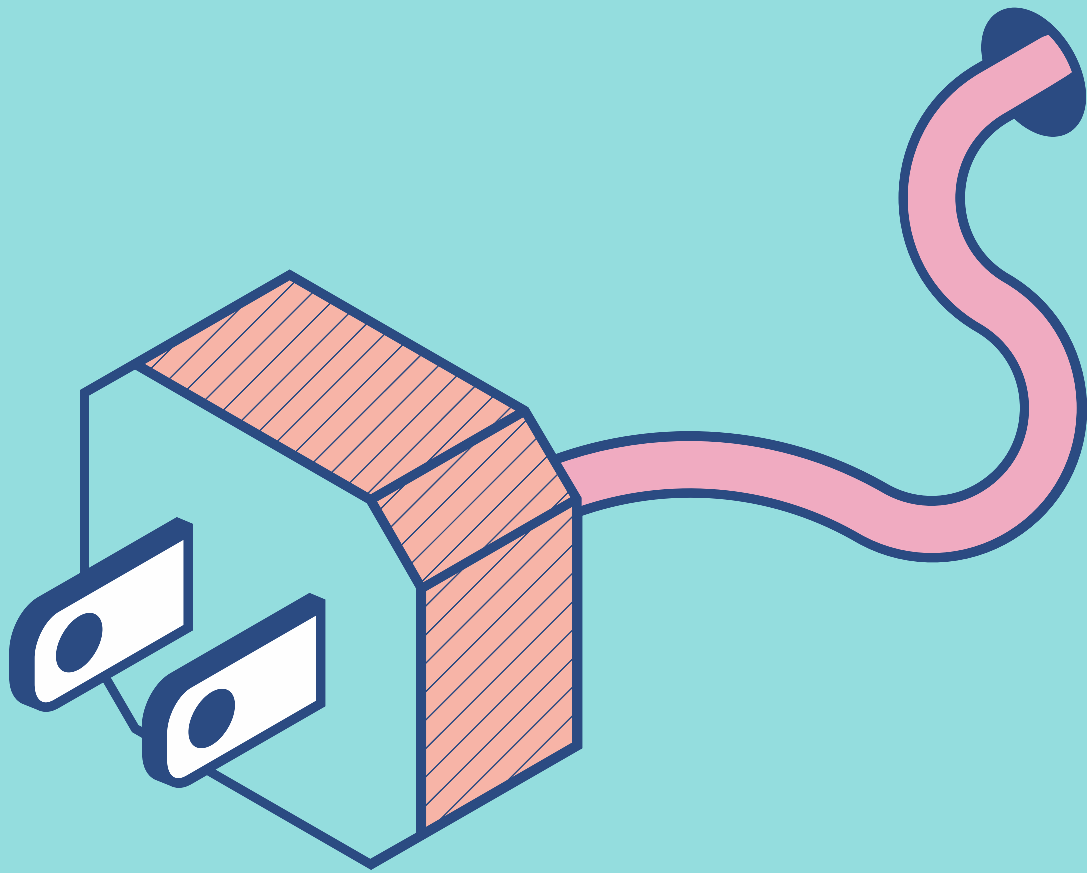


```
pylint main.py
```



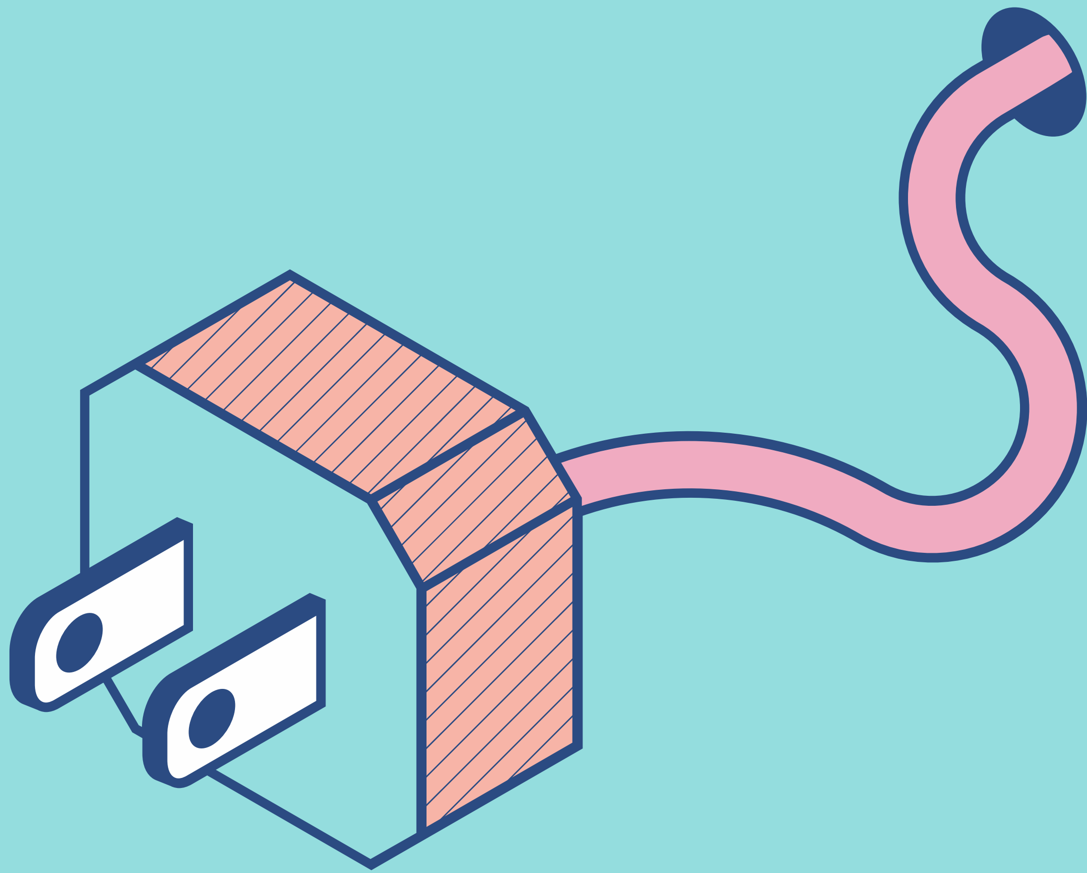
```
main.py:8:43: C0303: Trailing whitespace (trailing-whitespace)
main.py:10:0: R0903: Too few public methods (1/2) (too-few-public-methods)


-----------------------------------
Your code has been rated at 8.82/10
```

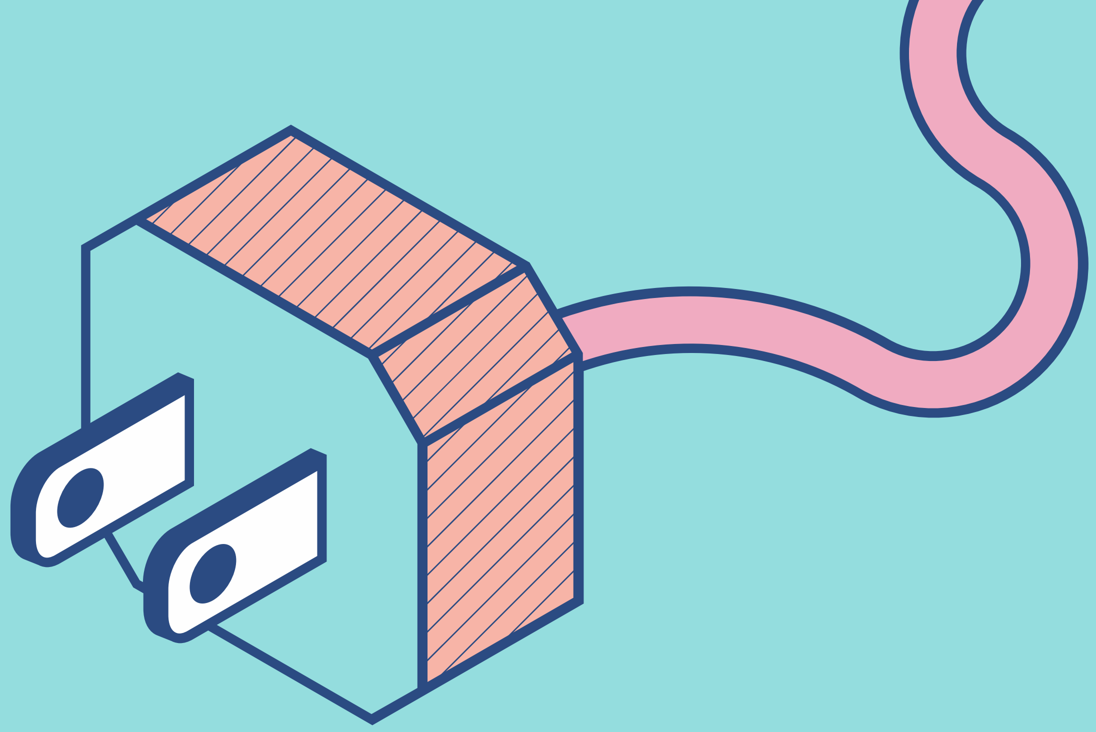


6.Pytest

PyTest is a testing framework that allows users to write test code using the Python programming language.

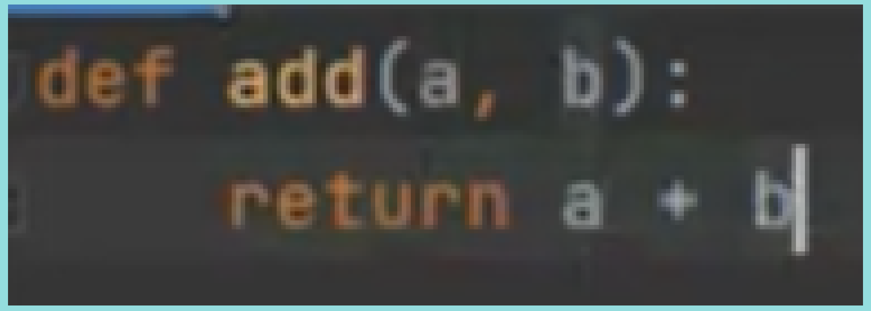


6.Pytest



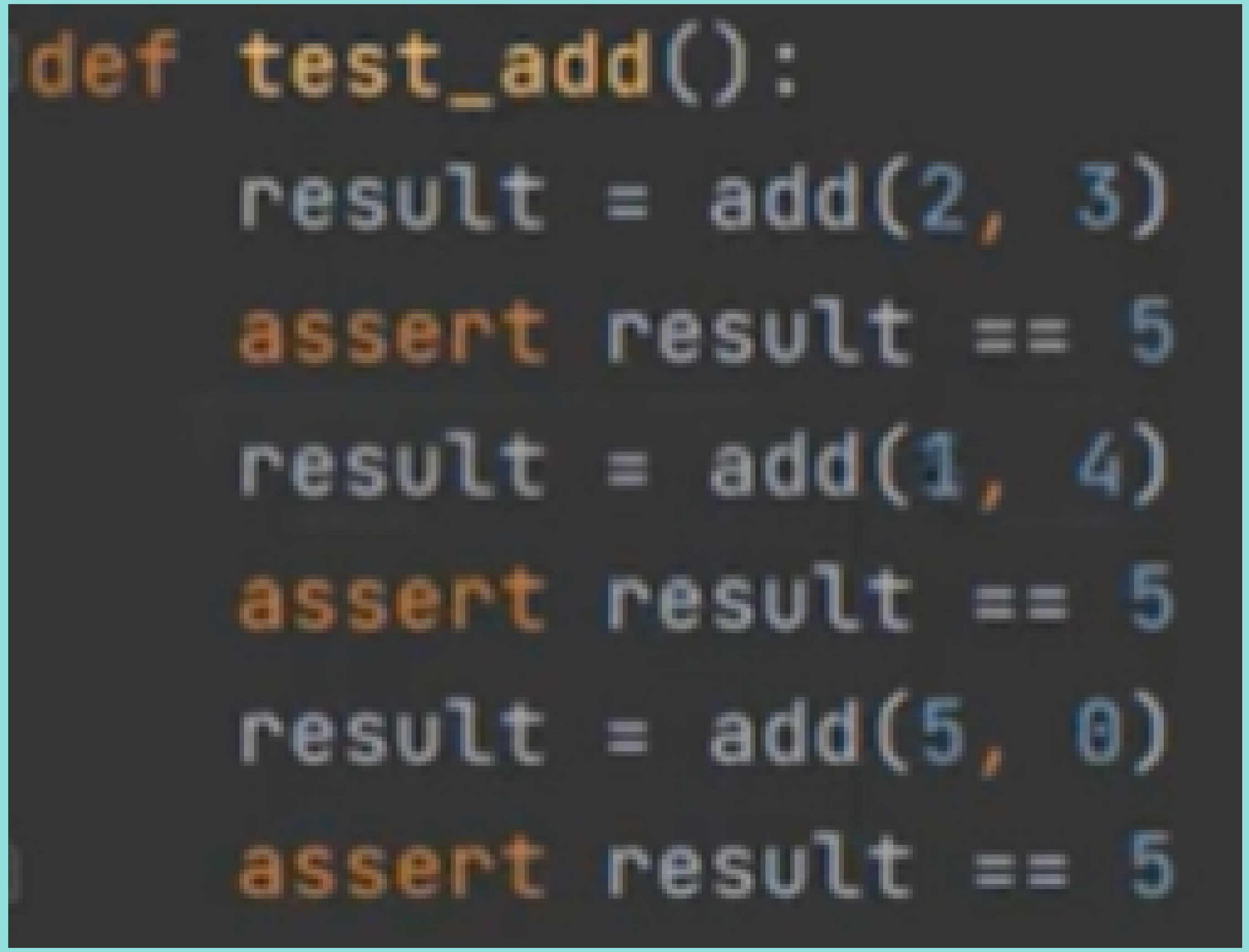
Example:

main.py



```
def add(a, b):
    return a + b
```

test_main.py



```
def test_add():
    result = add(2, 3)
    assert result == 5
    result = add(1, 4)
    assert result == 5
    result = add(5, 0)
    assert result == 5
```

Thank you!