

Developing an API for a Distributed Environment

In this session, you will create a RESTful API which can be used to create and delete user records. Responses to the questions should be recorded in your e-portfolio.

You are advised to use these techniques to create an API for your team's submission in Unit 11 and be prepared to demonstrate it during next week's seminar (Unit 10). Remember that you can arrange a session with the tutor during office hours for more support, if required.

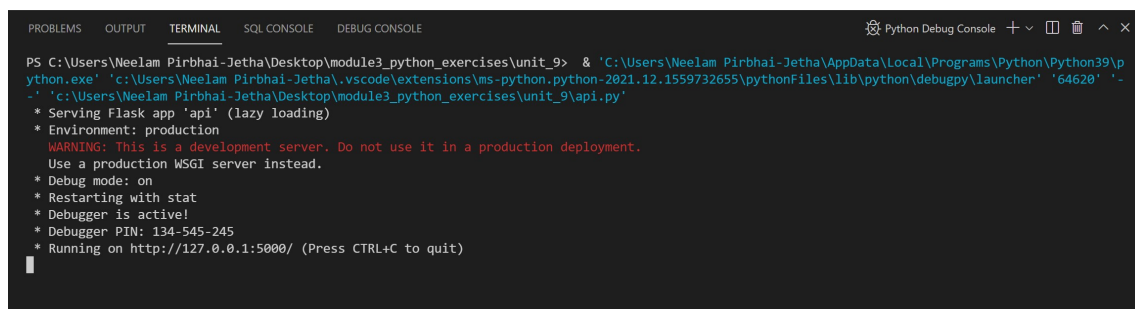
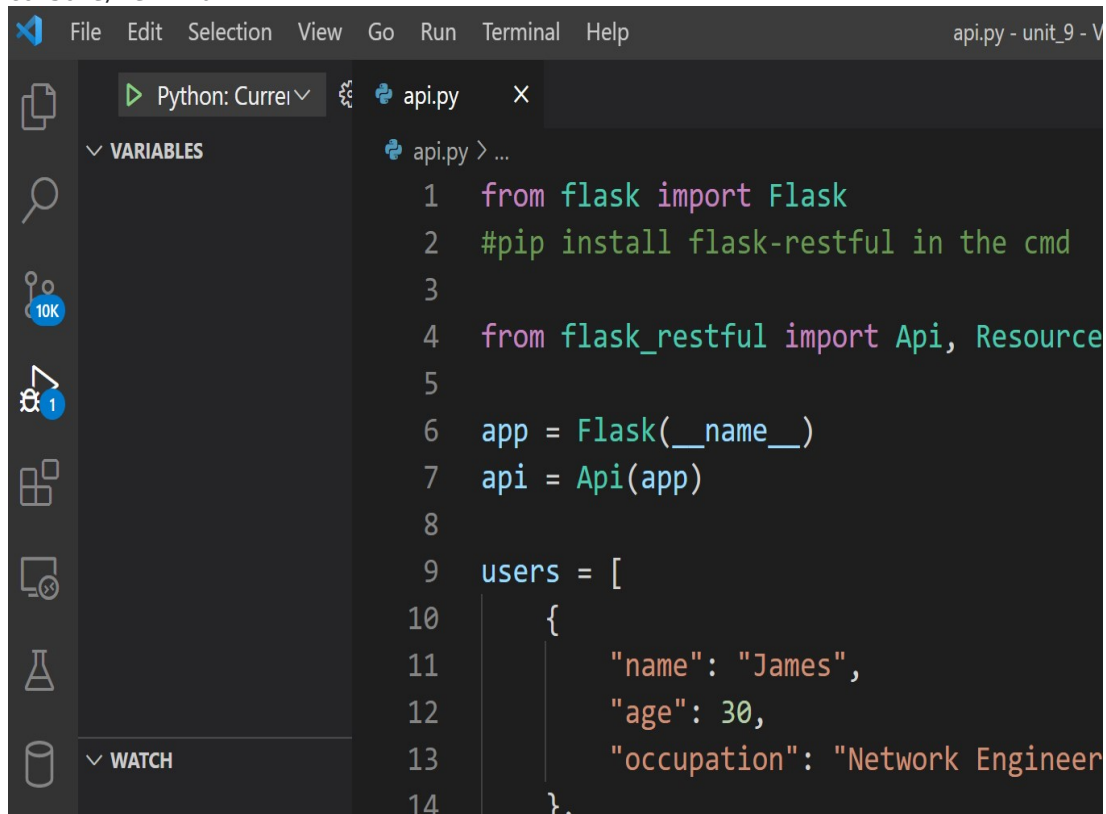
Using the [Jupyter Notebook workspace](#), create a file named api.py and copy the following code into it (a copy is provided for upload to Codio/GitHub):

#source of code: <https://codeburst.io/this-is-how-easy-it-is-to-create-a-rest-api-8a25122ab1f3>

Question 1

Run the API.py code. Take a screenshot of the terminal output. What command did you use to compile and run the code?

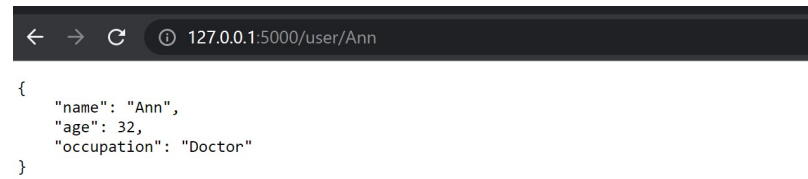
Visual studio code was used and when the codes were run, they appeared in the console/Terminal



Question 2

Run the following command at the terminal prompt: <http://127.0.0.1:5000/user/Ann>

What happens when this command is run, and why?



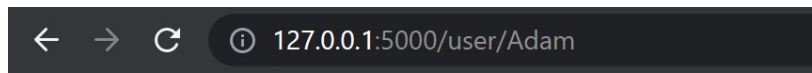
The screenshot shows a web browser window with the address bar displaying `127.0.0.1:5000/user/Ann`. The browser content area displays a JSON object representing a user record.

```
{  
  "name": "Ann",  
  "age": 32,  
  "occupation": "Doctor"  
}
```

Question 3

Run the following command at the terminal prompt:
w3m <http://127.0.0.1:5000/user/Adam>

What happens when this command is run, and why?



"User not found"

Question 4

What capability is achieved by the flask library?

Flask is often referred to as a microframework. It is designed to keep the core of the application simple and scalable. Instead of an abstraction layer for database support, Flask supports extensions to add such capabilities to the application.

Sources: <https://pythonbasics.org/what-is-flask-python/>

Architecture Evolution Activity

Based on your reading this week, could you write a section that might be appended to this paper, Salah et al, 2016, which would present the next phase of evolution history, from microservices to the technologies which are commonly in use today?

Remember to record your results, ideas and team discussions in your e-portfolio.

Check more info before writing next phase of evolution

-Flask Architecture

-Flask API

Additional Notes in Google Scholar

“Representational State Transfer (REST) is a software architectural style for web services that provides a standard for data communication between different kinds of systems.

REST is a standard for exchanging data over the Web for the sake of interoperability between computer systems.

Web services which conform to the REST architectural style are called RESTful web services which allow requesting systems to access and manipulate the data using a uniform and predefined set of stateless operations.

Since its inception in 2000 by Roy Feilding, RESTful architecture has grown a lot and has been implemented in millions of systems since then. REST has now become one of the most important technologies for web-based applications and is likely to grow even more with its integration in mobile and IoT-based applications as well.

Every major development language has frameworks for building REST web services. REST principles are what makes it popular and heavily used. REST is stateless, making it straightforward for any kind of system to use and also making it possible for each request to be served by a different system.

REST enables us to distinguish between the client and the server, letting us implement the client and the server independently. The most important feature of REST is its statelessness, which simply means that neither the client nor the server has to know the state of each other to be able to communicate. In this way, both the client and the server can understand any message received without seeing the previous message.”

Relan, K. (2019) *Building REST APIs with Flask Create Python Web Services with MySQL*. New Delhi (India): APress. Available from: <https://edu.anarcho->

[copy.org/Programming%20Languages/Python/Building%20REST%20APIs%20with%20Flask.pdf](https://www.fullstackpython.com/Programming%20Languages/Python/Building%20REST%20APIs%20with%20Flask.pdf)

Microservices

Microservices are an application architecture style where independent, self-contained programs with a single purpose each can communicate with each other over a network. Typically, these microservices are able to be deployed independently because they have strong separation of responsibilities via a well-defined specification with significant backwards compatibility to avoid sudden dependency breakage.

However, microservices have more substance because they are typically based on RESTful APIs that are far easier for actual software developers to use compared with the previous complicated XML-based schemas thrown around by enterprise software companies. In addition, successful applications begin with a monolith-first approach using a single, shared application codebase and deployment. Only after the application proves its usefulness is it then broken down into microservice components to ease further development and deployment. This approach is called the "monolith-first" or "MonolithFirst" pattern.

Sources:

<https://www.fullstackpython.com/microservices.html>