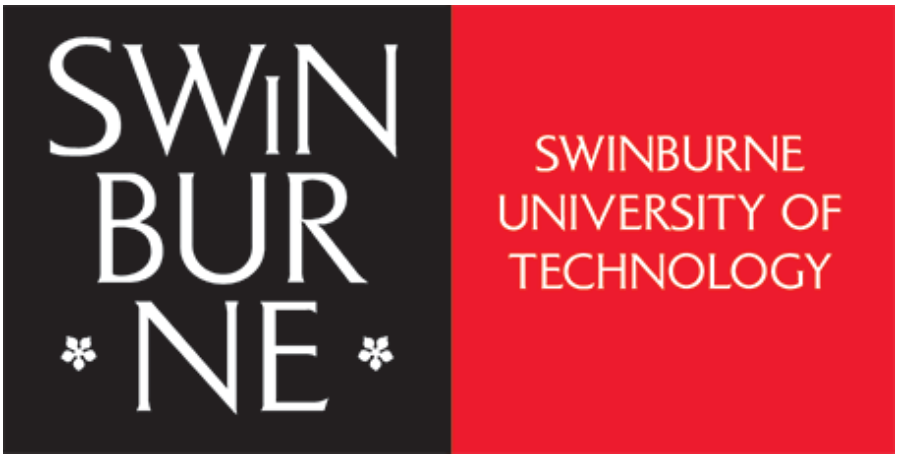


2023-COS30049-Computing Technology Innovation Project

Workshop Guide

Note: It's crucial to modify the distribution of time according to your specific requirements. You might also need to devise your own PowerPoint presentation in line with the guidelines of the workshop.

Workshop 05

Introduction of Anaconda and FastAPI

Objective: By the end of this workshop, students should have an understanding of the basic concepts of FastAPI and be able to develop HTTP requests on FastAPI. In addition, JSON objects will be introduced in this week's workshop material in order to let students pass and store values between different applications.

Workshop Structure:

1. Python Environment Introduction and Configuration（20 mins）:

Using pip and Anaconda simplifies Python configuration. Pip, the package installer, offers a vast repository of Python packages, accessed through command-line. Anaconda, an open-source distribution, streamlines package management and environment creation, ideal for scientific computing, data analysis, and machine learning, all enhancing Python's capabilities.

What is Anaconda?

Anaconda is an open-source distribution designed for scientific computing, data analysis, and machine learning. It includes a variety of commonly used data science tools and libraries. The core components of the Anaconda distribution include the Python programming language, the Conda package manager, and a wide range of libraries and tools for data analysis and scientific computing.

- **Python Interpreter:** Anaconda comes with a version of the Python programming language, typically from the Python 3.x series.

- **Conda Package Manager:** Conda is the package management tool of Anaconda, used for installing, managing, and updating Python packages and other software. It facilitates the creation of virtual environments to isolate dependencies for different projects and works across platforms.

Install Anaconda on your laptop

For MAC Users:

- Download the package through https://www.anaconda.com/download
- Package Installation Guide: https://docs.anaconda.com/free/anaconda/install/mac-os/

For Windows Users:

- Download the package through https://www.anaconda.com/download#downloads
- Package Installation Guide: https://docs.anaconda.com/free/anaconda/install/windows/

Common Conda Commands

Here are some common Conda commands that you can use in the command-line interface (CLI) to manage packages, environments, and more:

- Create a new environment:
  conda create --name myenv python=3.9
- Activate an environment:
  conda activate myenv
- Deactivate the current environment:
  conda deactivate
- List all existed python environment
  conda env list

2. FastAPI Introduction and Configuration（20 mins）

Introduction of FastAPI

FastAPI is a modern Python web framework designed for building high-performance APIs and web applications. It offers the following features:

- **Fast**: True to its name, FastAPI is focused on delivering exceptional performance. Built on the Starlette framework, it leverages Python's asynchronous programming capabilities to achieve high concurrency and low latency request handling.

- **Automatic Documentation Generation**: FastAPI can automatically generate API documentation based on type annotations in your code. This simplifies the process of documenting APIs, and the documentation is always in sync with the actual code.

- **Data Validation**: FastAPI uses type annotations to define the data structures of input parameters and response results. It automatically validates request data and generates appropriate error responses, reducing the need for manual data validation.

- **Developer-Friendly**: FastAPI incorporates modern development practices such as automated documentation and interactive API testing, enhancing the development experience.

Install FastAPI

Run the following command to install FastAPI:

conda install fastapi

Also install uvicorn to work as the server:

conda install uvicorn uvicorn-standard

3. Introduction FastAPI, JSON and HTTP Requests（75 mins）

What is HTTP Request?

Here's a breakdown of the differences between the HTTP request methods POST, GET, PUT, and DELETE:

GET:

- Purpose: Used to retrieve data from the server, typically for reading resources.
- Data Passing: Data is appended to the URL's query string and is visible in the URL.
- Idempotence: GET requests are idempotent, meaning making the same GET request multiple times won't have different effects on the server.
- Safety: GET requests are considered safe as they don't modify server data.

POST:

- Purpose: Used to submit data to the server, often for creating new resources.
- Data Passing: Data is included in the request's body and is not visible in the URL.
- Idempotence: POST requests are not necessarily idempotent; making the same POST request multiple times may create different resources each time.
- Safety: POST requests are not safe, as they can potentially modify server data.

PUT:

- Purpose: Used to update or replace a resource on the server.
- Data Passing: Data is included in the request's body and is used to update the resource specified in the URL.
- Idempotence: PUT requests are idempotent; making the same PUT request multiple times will result in the same outcome.

- Safety: PUT requests are not safe, as they can modify server data.

DELETE:

- Purpose: Used to delete a resource from the server.
- Data Passing: Data is typically not passed since the target for - on is specified in the URL.
- Idempotence: DELETE requests are idempotent; making the same DELETE request multiple times will result in the same outcome.
- Safety: DELETE requests are not safe, as they can modify server data.

In web development, we use GET and POST as most common request methods to retrieve data from the database or pose data manipulation.

How to run FastAPI locally:

cd <your project folder>
uvicorn main:app --reload

The command uvicorn main:app refers to:

- main: the file main.py (the Python "module").
- app: the object created inside of main.py with the line app = FastAPI().
- --reload: make the server restart after code changes. Only used for development.

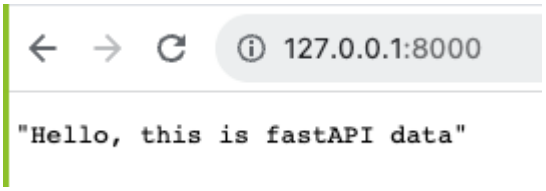
main.py :

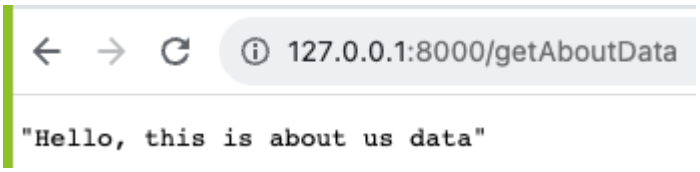
```
@app.get("/")
async def funcTest1():
      return "Hello, this is fastAPI data"


@app.get("/getAboutData")
async def funcTest2():
      return "Hello, this is about us data"
```

```
@app.get("/getHomeData")
async def funcTest3():
      return "Hello, this is home data"
```

Code can be found at: https://codesandbox.io/p/sandbox/fastapi-demo-kkd6yv?file=/main.py:19,1

Here are results with the corresponding routers:

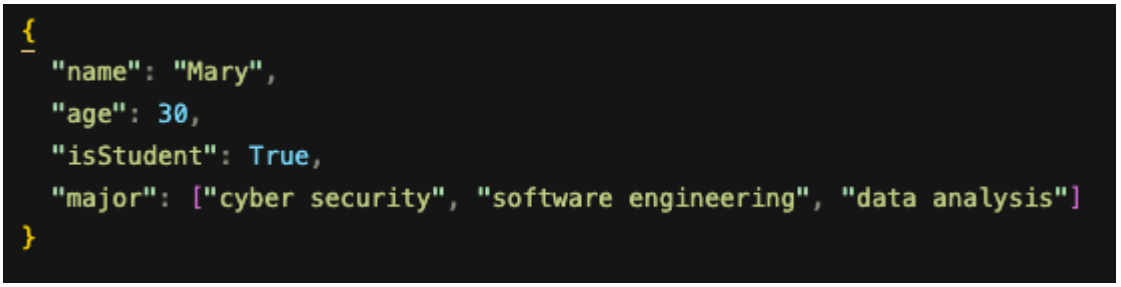




What is JSON Object?

In API documentation, **"JSON" stands for JavaScript Object Notation**. JSON is a lightweight data interchange format commonly used for passing data between different applications. It is a text-based format that is easy to read and write, as well as parse and generate.

In API documentation, JSON is often used to represent the structure of example requests and responses. It organizes data in key-value pairs and uses curly braces to represent objects and square brackets to represent arrays. Each key-value pair consists of a key (string) and a value (which can be a string, number, boolean, object, or array). Here is a sample JSON object.



Try to define a router inside main.py and return a JSON object
Sample code can be found here:
https://codesandbox.io/p/sandbox/fastapi-demo-kkd6yv?file=/main.py:21,16

View the auto-generated API document on:

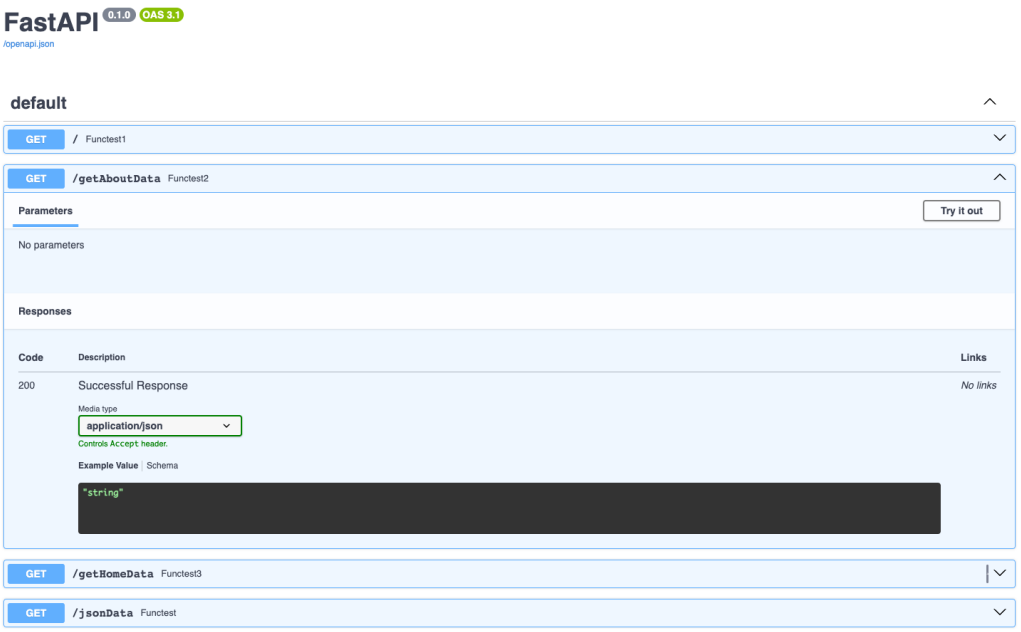
http://127.0.0.1:8000/docs

The documentation in FastAPI serves several purposes:

- *Automated Documentation Generation*: FastAPI automatically generates detailed API documentation based on type annotations and other metadata in the code. Developers only need to add appropriate comments and type annotations in the code, and the documentation is generated automatically, eliminating the need for manual documentation writing.

- *Interactive API Testing*: FastAPI documentation is not just static documentation—it allows you to directly test API endpoints within the documentation interface. You can input parameters, send requests, and view responses, enabling you to quickly verify the functionality of the API without leaving the documentation page.

- *Request and Response Examples*: The documentation not only provides detailed explanations of the API but also displays examples of requests and responses. This helps developers understand the expected behavior of the API and provides example code for constructing requests and handling responses.



4. Workshop Tasks

Write GET Methods

Example 1: GET data with the URL

```
@app.get("/jsonData")
      async def funcTest():
            jsonResult = {
                  "name": "Your name",
                  "Uni-year": 2,
                  "isStudent": True,
                  "hobbies": ["reading", "swimming"]
            }
            return jsonResult
```

Example 2: Passing data to GET URL

```
@app.get("/student/{student_id}")
async def getStudentId(student_id: int):
      return {"student_id": student_id}
```

Write POST Method:

Example 3: Passing JSON data to POST URL

Firstly, you will need to define a model to verify the incoming data for POST methods.

Import the BaseModel in the top:

```
from pydantic import BaseModel
```

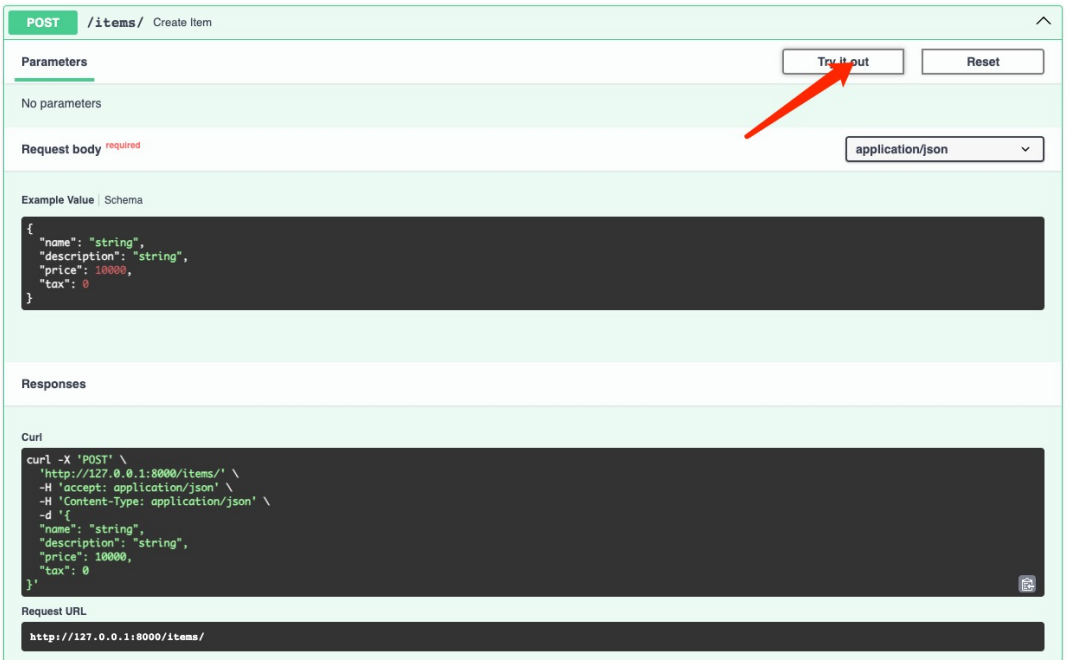
Define the data class

```
class Item(BaseModel):
    name: str
    description: str = None
    price: float
    tax: float = None
```

Define the POST method

```
@app.post("/items/", response_model=Item)
def create_item(item: Item):
    return item
```

Test Function In the FastAPI Documentation:



Sample code can be found here:

https://codesandbox.io/p/sandbox/fastapi-demo-kkd6yv?file=/main.py:43,1-46,1

5. Reflection (5 mins)

Give students the opportunity to share what they learned, found interesting, or had difficulty understanding. Offer additional resources for them to learn more about FastAPI Python Framework

Online Code Sample:
https://codesandbox.io/p/sandbox/fastapi-demo-kkd6yv?file=/main.py:43,1-46,1

What is HTTP request method ?
https://www.c-sharpcorner.com/article/http-methods-in-fastapi/
https://fastapi.tiangolo.com/advanced/using-request-directly/

FastAPI Cookbook
https://lyz-code.github.io/blue-book/fastapi/

What is JSON
https://www.w3schools.com/whatis/whatis_json.asp
https://www.youtube.com/watch?v=iiADhChRriM&ab_channel=WebDevSimplified

Common commands in Anaconda
https://www.python-engineer.com/posts/anaconda-basics/