Flask API

What is our GOAL for this MODULE?

The goal of this module is to curate all the data we have in a list of dictionaries and create a Flask API for it.

What did we ACHIEVE in the class TODAY?

- Curated all the data
- Created a Flask API
- Tested the API

Which CONCEPTS/CODING BLOCKS did we cover today?

- Python
- Flask
- MVC Architecture

How did we DO the activities?

1. We have calculated the gravity and the speed of the planet. We will add these fields into our data where we create the list of dictionaries.

```
try:
   if gravity < 100:
     features_list.append("gravity")
   planet_data.append(gravity)
 except: planet_data.append("Unknown")
```

```
try:
   try:
     distance = 2 * 3.14 * (float(planet_data[8].split(" ")[0]) *
1.496e+9)
   except:
     try:
       distance = 2 * 3.14 * (float(planet_data[8]) * 1.496e+9)
     except: pass
   try:
     time, unit = planet_data[9].split(" ")[0],
planet_data[9].split(" ")[1]
     if unit.lower() == "days":
       time = float(time)
     else:
       time = float(time) * 365
   except:
     time = planet_data[9]
   time = time * 86400
   speed = distance / time
   if speed < 200:
     features_list.append("speed")
   planet_data.append(speed)
 except: planet_data.append("Unknown")
```

2. We create a list of dictionaries containing specific data points for all the planets. There data points are:
    - Name
    - Distance from Earth
    - Planet Mass
    - Planet Radius
    - Planet Type
    - Distance from their sun (Orbital Radius)
    - Orbital Period
    - Gravity
    - Orbital Speed
    - Specifications / Features that we have in our **final_dict**

```
final_planet_list = []

for planet_data in planet_data_rows:
 temp_dict = {
                "name": planet_data[1],
                "distance_from_earth": planet_data[2],
                "planet_mass": planet_data[3],
                "planet_type": planet_data[6],
                "planet_radius": planet_data[7],
                "distance_from_their_sun": planet_data[8],
                "orbital_period": planet_data[9],
                "gravity": planet_data[20],
                "orbital_speed": planet_data[21]
            }
 temp_dict["specifications"] = final_dict[planet_data[1]]
 final_planet_list.append(temp_dict)

print(final_planet_list)
```

3. Now, we are ready to create our Flask API. In the terminal, create a new directory for Flask Project.
4. Create the virtual environment and source it.
5. **Pip install flask** inside the virtual environment.
6. Create a file **data.py**. Inside this file, we want to create a variable **data** and we will copy the list of dictionaries we printed above (**final_planet_list**) and paste it as the value of this variable.

7. Create a new file **main.py** and import flask in it. Create the app variable using Flask.

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

8. Create a view for index/home and return **"Index"** in it for now.

```
@app.route("/")
def index():
   return "Index"
```

9. Finally run the app.

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

10. Run the server from the terminal using **python main.py** command.
11. Browse to **localhost:5000** to see the output.
12. We want to display the list of all the planets in our home page of the mobile app. We want to make sure that this route returns all the data stored in **data.py** for the same.
13. We want to return this data as a JSON. For that, import **jsonify** as well as data.

```
from flask import Flask, jsonify
from data import data
```

14. Now change the view function for the index page to return the data.

```
@app.route("/")
def index():
   return jsonify({
       "data": data,
       "message": "success"
   }), 200
```

15. Re-run the server and check on the browser the result you get. It should now contain all the curated data of the planets that we saved in our **data.py.**
16. This was our first API. We want to create another API that will return data for only one planet. For this, we need to have a unique identifier with which we can tell what planet's data the user is requesting. We can use the name of the planet for the same.

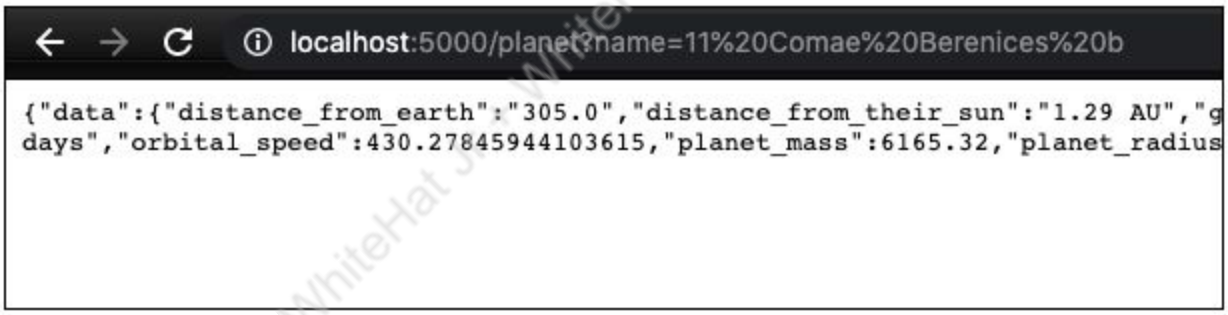
17. We provide the name of the planet as the URL parameter. To fetch that from the URL, we need to import **request** from flask.

```
from flask import Flask, jsonify, request
```

18. We create another route which will take the name of the planet from the URL argument and then find its data in the list of dictionaries and then return the data.

```
@app.route("/planet")
def planet():
    name = request.args.get("name")
    planet_data = next(item for item in data if item["name"] == name)
    return jsonify({
        "data": planet_data,
        "message": "success"
    }), 200
```

19. Re-run the server to check if this API works fine!



20. We are done!

What's NEXT?

In the next class, We will create a Flask API for our data so that we can integrate the API in a mobile application.

EXTEND YOUR KNOWLEDGE:

You can read the following blog on speed of our planet to understand more:
https://flask.palletsprojects.com/en/1.1.x/quickstart/#a-minimal-application