MAKE SCHOOL

APIs & JSON

WEB 1.1

Agenda

- Learning Outcomes
- Hook: Hip-Hop Artists API
- Postman & APIs
- **BREAK**
- Using the 'requests' library to make an API call
- Using Flask + requests!
- Wrap-Up

Learning Outcomes

By the end of today, you should be able to…

1. **Explain** how APIs can be useful in retrieving data.
2. **Use** the Postman desktop program to make an API call.
3. **Use** the `requests` library to make an API call within Python code.

Review: APIs

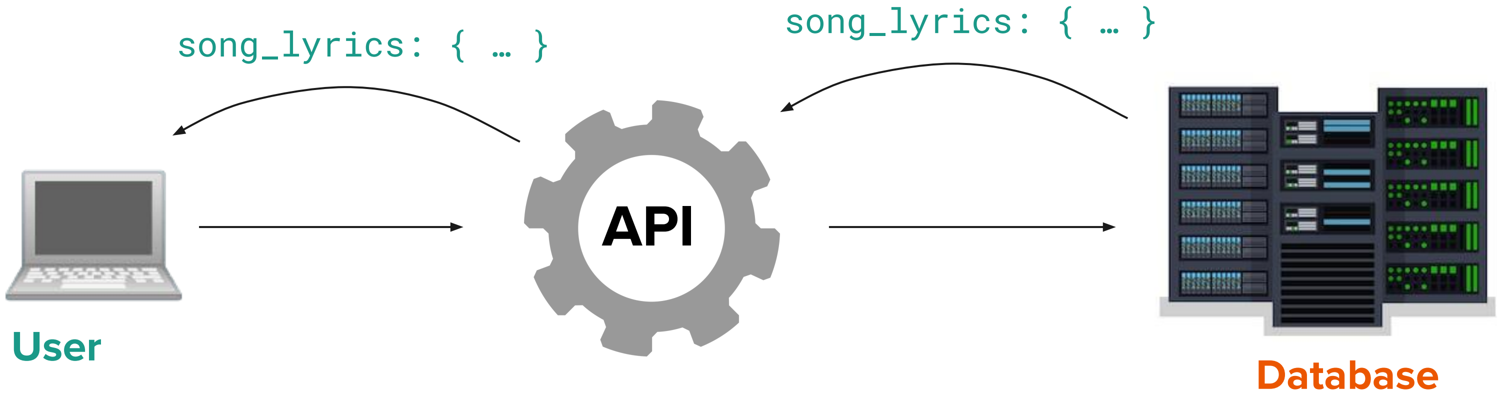
Warm-Up (8 minutes)

Explain to a partner **what happens when we make an API call**. Pretend you are explaining it to your 7-year-old cousin. Then, switch roles.

(Hint: Use an analogy!)

APIs

API stands for **Application Programming Interface**. It's the **interface** we use to access someone else's database.



Requests Library

MAKE SCHOOL

The `PrettyPrinter` class is used to print out the JSON data in a nicely formatted way. Let’s see the difference!

```python
import requests
from pprint import PrettyPrinter

pp = PrettyPrinter(indent=4) # make a PrettyPrinter object

params = { "limitTo": "nerdy" } # set the request’s query parameters
result = requests.get(
    "http://api.icndb.com/jokes/random",
    params=params) # make the request
joke_json = result.json() # get the JSON data of the response
pp.pprint(joke_json) # print it out, nicely formatted :)
```

PrettyPrinter is a tool that prints JSON (the data we get from an API) in a nicely formatted, more readable way.

Requests Library

Here’s a **simplified look** at how to make an API request. Replace “KEY” and “VALUE” with any request parameters (optional), and replace the URL with the API’s URL.

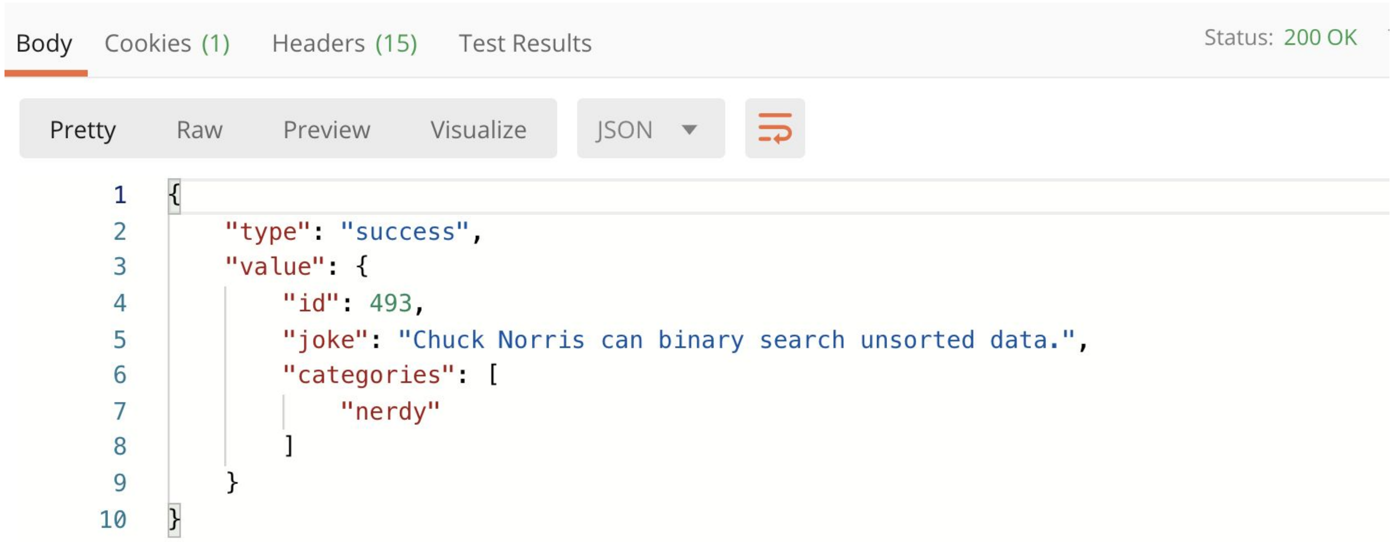
```
import requests

...

params = { "KEY": "VALUE" }
result = requests.get("http://my-api.co", params=params)
result_json = result.json()
// do something with result_json
```

this is what’s actually making the API request

JSON

What is JSON?

JSON stands for **JavaScript Object Notation**. Any time we make an API call, the **response data** is returned in JSON format.



```json
{
    "type": "success",
    "value": {
        "id": 493,
        "joke": "Chuck Norris can binary search unsorted data.",
        "categories": [
            "nerdy"
        ]
    }
}
```

What is JSON?

In Python, we refer to **JSON data** as a **dictionary**. (In reality, they're slightly different, but their syntax/usage is the same.)

A **dictionary** is just a collection of **key-value pairs**. The keys must be strings, but the values can be any data type. Notice how a value can contain a list, or even another dictionary!

```
my_info = {
    'name': 'Fluffy',
    'likes': ['tennis balls', 'walks'],
    'address': {
        'street': '555 Post St',
        'city': 'San Francisco'
    }
}
```

What is JSON?

What if we want to get Fluffy’s street address? Type in the missing code!

```
my_info = {
    'name': 'Fluffy',
    'likes': ['tennis balls', 'walks'],
    'address': {
        'street': '555 Post St',
        'city': 'San Francisco'
    }
}

street_address = 
```

JSON Practice Activity (15 minutes)

With a partner, complete the TODOs in the Dictionaries Practice Repl.it.

MAKE SCHOOL

The Weather API

Weather API

For Homework 4, we'll be using the OpenWeatherMap API to build a project.

This API gives us data on the current temperature, humidity, sunrise/sunset times, description, etc. for a given city.

Follow the steps on this page to get your own API key!

Weather API - Activity (15 minutes)

With a partner, complete the TODOs in the Weather API Practice Repl.it. **Make sure you use your own API key!**

Break - 10 min

Midterm Self-Assessment

Self-Assessment

Take 15-20 minutes to complete the Midterm Self-Assessment on Gradescope.

Your instructor will take this into account when compiling midterm grades!

MAKE SCHOOL

Virtual Environments

What is a virtual environment?

A **virtual environment** allows us to manage Python packages for different projects.

Think of it like a **sandbox or container** where we can install packages and they won't affect the rest of your file system or projects.

For example, for **Project A** you may need **Version 1** of a package, but for **Project B** you may need **Version 2**. Since those versions are incompatible, you'll need a separate virtual environment for each project.

How do I use one?

MAKE SCHOOL

Navigate to your project directory, then:

- `python3 -m venv env` - create a folder called `env` that will hold all installed packages
- `source env/bin/activate` - activate your virtual environment; do this before you run your code or install packages
- `deactivate` - deactivate your virtual environment; do this when switching to work on a different project

What is requirements.txt?

In Python projects, we use a file called `requirements.txt` to list all of the project's required packages.

- `pip3 install -r requirements.txt` - install all packages listed in the requirements
- `pip3 freeze > requirements.txt` - populate the requirements.txt file with all packages that are currently installed in the environment

Activity (10 minutes)

Navigate to your Homework 2 directory. Use the steps on the previous slides to create a virtual environment, activate it, install Flask, & "freeze" to a requirements file.

Environment Variables

What are Environment Variables?

Sometimes, there are pieces of information (e.g. API keys) that you want to keep secret from the public. (Why?)

We call these pieces of information **secrets**.

It is very important to hide your secrets!!!

We can do this by saving each "secret" as an **environment variable**.

What are Environment Variables?

An **environment variable** is a key-value pair that is saved to your operating system's environment.

We can read in environment variables using the **dotenv** Python package.

How do we use it?

1. Install the dotenv package: `pip3 install python-dotenv`

2. Create a file called `.env` and enter your key-value pairs (do not use quotes):

```
# .env
SECRET_KEY=ilikebananas
```

3. In your Python code, enter the following:

```python
import os
from dotenv import load_dotenv
load_dotenv()

secret_key = os.getenv('SECRET_KEY')
print(secret_key) # 'ilikebananas'
```

Activity (10 minutes)

Create a new folder called `dotenv-practice`. In it, create `.env` and `app.py`.

Follow the steps on the previous slide to create a key-value pair and read it in using `app.py` and print it to the console.

Lab Time: Work on Homework 3 or 4

Homework

If you are taking the quiz, leave the zoom room so I don't disturb you!

Due on Thursday: Homework 3