Introduction to APIs

Lesson Objectives

After this lesson, you will be able to…

- Describe what an application programming interface (API) is and why we might use one.
- Identify common APIs on the web.
- Call an API.

Discussion: Web Magic

Have you seen…

- A website with Google Maps on the page (like Yelp)?
- A program that had live stock market info?
- A website that isn’t Twitter but shows a live Twitter feed?
- Any app that pulls info from somewhere else?

How did they do this?

APIs (Application Program Interfaces)

An API is a service that provides raw data for public use.

APIs give us data, maps, anything!

What's the API?	Sample URL — put this in a new tab!
The Star Wars API: Request R2-D2 info	http://swapi.co/api/people/3
Markit Digital's API: Request current Apple stock info	http://dev.markitondemand.com/Api/Quote/xml?symbol=AAPL
OpenWeatherMap: The current weather in London	https://samples.openweathermap.org/data/2.5/weather?q=London,uk&appid=b6907d289e10d714a6e88b30761fae22

Do you think you've been on websites that call an API?

Does the JSON look unreadable in the browser? If you're using Chrome, install the JSONView plugin.

How Do We Use an API?

We'll use the `requests` module.

```
import requests


# Call the API by opening the URL and reading the data.
# We use the `get()` function in `requests`.
response = requests.get("<API URL HERE>")


print(response)
# Prints out the requested information!
```

This works, but there's one very helpful line missing!

Before we see this in action, let's look at what the API might return.

JSON vs. XML

Imagine: You write code for a list.

```
my_list = [1, 4, 2]

my_list.append(len(my_list))

my_list[1] = "new element!"


for item in my_list:

  print item
```

But then, `my_list` is unexpectedly a dictionary, or an int, or even a class! The code we wrote won't work.

APIs can give data back in two ways: JSON or XML. Depending on what the API does, we need to write our program a different way.

How Do APIs Give Us Info? Option 1: JSON

Here's a potential return from an API:

```
{
  "users": [
    {"name": "Wonder Woman", "id": 0},
    {"name": "Black Panther", "id": 1},
    {"name": "Batgirl", "id": 2}
  ]
}
```

Looks like a dictionary with a list of dictionaries inside it, right?

But it's not a dictionary! It's **JSON** (JavaScript Object Notation).

The `requests` module has a built-in JSON decoder to turn JSON into a Python dictionary.

We can **decode** JSON with `decoded_data = response_from_request.json()`.

How Do APIs Give Us Info? Option 2: XML

Instead of JSON, we might get XML:

```
<users>
  <user id="0">
    <name>Wonder Woman</name>
  </user>
  <user id="1">
    <name>Black Panther</name>
  </user>
  <user id="2">
    <name>Batgirl</name>
  </user>
</users>
```

JSON is certainly easier to read!

- We'll stick with JSON whenever we can.

Pro tip: *Most of you don't need to know about XML, but if you're working with legacy code or an older API, you may have to use it. In that case, look up Element Tree XML.*

Let’s Choose an API

To recap: APIs give us data we can use in either XML or JSON.

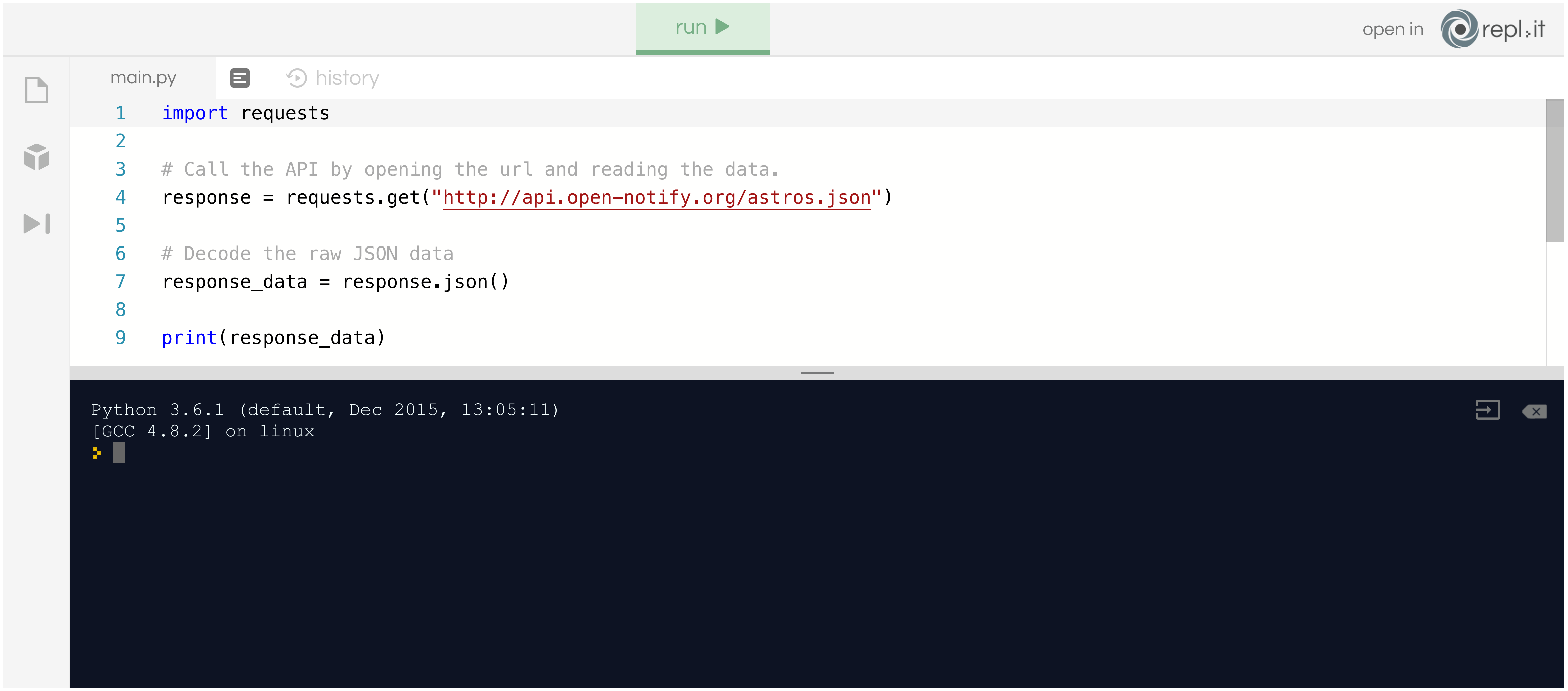
Let’s call one!

Check out http://api.open-notify.org/astros.json, which tells us the people currently aboard the International Space Station (ISS).

```
{
    "number": 5,
    "people": [
        {"craft": "ISS", "name": "Oleg Novitskiy"},
        {"craft": "ISS", "name": "Thomas Pesquet"},
        {"craft": "ISS", "name": "Peggy Whitson"},
        {"craft": "ISS", "name": "Fyodor Yurchikhin"},
        {"craft": "ISS", "name": "Jack Fischer"}
        ],
        "message": "success"
}
```

Calling an API

- Import the `request` module.
- Call the API (`requests.get()`).
- Parse the response with `response.json()`.



```python
import requests

# Call the API by opening the url and reading the data.
response = requests.get("http://api.open-notify.org/astros.json")

# Decode the raw JSON data
response_data = response.json()

print(response_data)
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
```

You Do: Calling an API

Open a new file, `my_api.py`. Type and run the code:

```
import requests


# Call the API by opening the URL and reading the data.

response = requests.get("http://api.open-notify.org/astros.json")


# Decode the raw JSON data.

response_data = response.json()


print(response_data)
```

We Do: A New API

Awesome! Go back to your file. Let’s instead call this URL:

`http://dev.markitondemand.com/Api/Quote/xml?symbol=AAPL`

Why does it break? We can’t parse XML like JSON.

```xml
<QuoteApiModel>
  <Data>
    <Status>SUCCESS</Status>
    <Name>Apple Inc</Name>
    <Symbol>AAPL</Symbol>
    <LastPrice>185.5</LastPrice>
    <Change>1.34</Change>
    <ChangePercent>0.7276281494</ChangePercent>
    <Timestamp>Thu Jun 28 00:00:00 UTC-04:00 2018</Timestamp>
    <MarketCap>911758099000</MarketCap>
    <Volume>17365235</Volume>
```

Quick Review

We've called an API! Great job. We did this with the `get()` function in the `requests` module. APIs are made available by other websites or applications. They give us data we can use in either XML or JSON.

```
import requests
response = requests.get("http://api.open-notify.org/astros.json")
response_data = response.json()
print(response_data)
```

JSON:

```
{
  "users": [
    {"name": "Wonder Woman", "id": 0},
    {"name": "Black Panther", "id": 1}
  ]
}
```

XML:

You Do: Back to JSON

Back in your file, change the API call back to `http://api.open-notify.org/astros.json`.

Once it's decoded, it's a dictionary!

Replace your `print` statement:

```
for key, ratings in response_data_decoded.items():
    print("Key:", key, "Value:", ratings, "\n")
```

Can we go further? Try to only print the `name`s of the astronauts.

Name Printing: Solution

Working backward, we have a:

- Dictionary (key: `name`).
- Which is inside a list (the value of `people`).
- Which is inside a dictionary (key: `people`).

```
For message the value is success.


For people the value is [{'craft': 'ISS', 'name': 'Oleg Artemyev'}, {'craft': '


For number the value is 6.
```

```python
for item in response_data_decoded["people"]:
    print(item["name"])
```

You Do: Shakespeare

In your file, call the Shakespeare API `http://ShakeItSpeare.com/api/poem`.

Print only the poem.

Shakespeare: Solution

Print only the poem.

```
import requests


# Call the API by opening the URL and reading the data.
response = requests.get("http://ShakeItSpeare.com/api/poem")


# Decode the raw JSON data.
response_data = response.json()


print(response_data["poem"])
```

Quick Review

When we convert JSON, it keeps the same format, only in a Python structure.

When parsing an API's return, look through the JSON to find the exact structure you need. Is it the string value from the `poem` key? Or the value from each `name` key in a list of dictionaries, which is the value of the `people` key?

Think it through before writing your code.

```
# From the ISS API:
{   # The outer dictionary
    "number": 5,   # Key: value
    "people": [   # Key and value, again. Here, the value is a list of dictic
        {"craft": "ISS", "name": "Oleg Novitskiy"},
        {"craft": "ISS", "name": "Thomas Pesquet"},
        {"craft": "ISS", "name": "Peggy Whitson"},
        {"craft": "ISS", "name": "Fyodor Yurchikhin"},
        {"craft": "ISS", "name": "Jack Fischer"}
        ],
        "message": "success"   # Key and value.
```

I Do: API Authentication

Many APIs are free but require a **key**. This identifies the developer requesting access.

If we call the Giphy API:

- With no key, `http://api.giphy.com/v1/gifs/search?q=funny+cat`, we get `Error - Unauthorized`!

- With a key, `http://api.giphy.com/v1/gifs/search?q=funny+cat&api_key=dc6zaTOxFJmzC`, it works!

Syntax Notes:

- The main API URL is `http://api.giphy.com/v1/gifs/search`.
- `?` always delineates a URL and its parameters.
  - (The `?` is a standard for every URL! Searching Google for “banana,” with `q` short for “query:” `https://www.google.com/search?q=banana`).
  - (Here’s another one! Searching Amazon for “banana:” `https://www.amazon.com/s?field-keywords=banana`.)

Most importantly, never publish your key for a backend service, including on GitHub! (This is an example.) There are other ways to provide your key to a server in order to keep that key safe. However, if your code is using

You Do: OpenWeather API

Read about the API here.

Use this key: `&appid=052f26926ae9784c2d677ca7bc5dec98`.

Call this URL: `http://api.openweathermap.org/data/2.5/weather?zip=<ZIP_CODE_HERE>,us&appid=052f26926ae9784c2d677ca7bc5dec98`.

Note the parameters:

- `zip=<ZIP_CODE_HERE>`
- `appid=<KEY HERE>`

Enter any zip code you choose (e.g., `60614`).

- Display the current temperature, high and low temperature, current weather description, and the name of the city that came back from the API.

Bonus: Print the temperature in both Fahrenheit and Celsius.

Summary

APIs:

- Handy URLs from which we can get information.
- Sometimes require keys.
- Usually free.
- Call with the `requests()` module.

XML and JSON:

- Two formats in which APIs might return information to us.
- XML is legacy.
- JSON looks like a dictionary.

Additional Resources

- Here's an example of a stolen key horror story.
- The Programmable Web API Directory
- Element Tree XML