Open in app Get started

Published in Quick Code

You have **2** free member-only stories left this month. Sign up for Medium and get an extra one

Prashanth Xavier Follow
Feb 8, 2019 · 10 min read ★ · Listen

Save

Absolute beginners guide to slaying APIs using Python

Have you ever wondered how online travel search engines like skyscanner or expedia cater the best deals for you, How they manage to keep their data updated even if it means gathering data from hundreds of carriers, How this data is relaying back and forth between a number of websites?

The answer is API, acronym for Application Programming Interface.

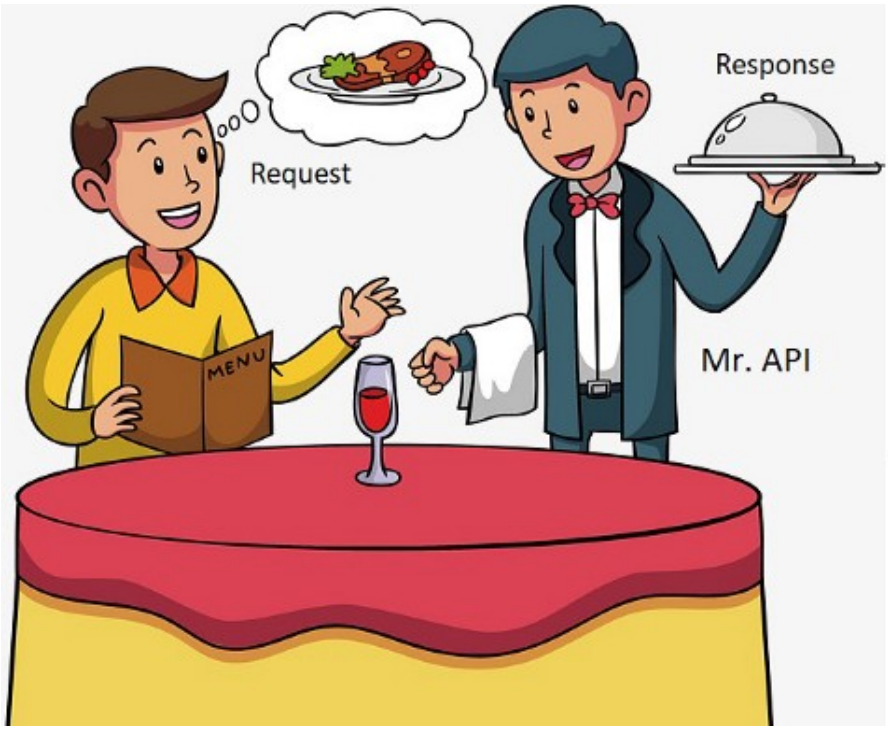
words and finally we conclude with the implementation of a more complex API which involves authentication, all this and more in under 10 minutes.

What is an API ?

The technical definition of an **API** goes like this, a set of subroutine definitions, communication protocols ……zzzzzzzzzzzzzzzzzzzzzzzzzzzzzz

Fancy! Lets try to flatten this a bit, imagine you are at a restaurant and you are in the mood for some cheese(apparently its your cheat day), and the waiter gives you the menu and you customize a pizza the way you like it and place your order. The waiter then sends your order straight to the kitchen, where it is prepared for you by Gordon Ramsay himself. The waiter then delivers the pizza to you.

Lets consider this analogy, you are the **requester,** the kitchen is a **system** or a **service** or a **website**, the waiter is the **API** and Gordon is Gordon. Yes, the API is the waiter who is merely a messenger who facilitates communication between two entities. The order you placed is the **request** and the pizza delivered to you is the **response**.



> *Application **P**rogramming **I**nterface is a standard that facilitates intercommunication between two or more computer programs.*

reason we want to use API is that it is easy to access, and data keeps changing, companies like twitter, facebook, google have tons of data that they serve in the form of API, now for sure you are never going to have as much data as they do.

That's enough about whats under the hood, lets jump in.

Modules in Python

In this part of the tutorial, we dive in with an assumption that we know a little bit of python and how modules are installed. If you are a complete beginner to python, you can still follow along, but do check out some reference links below for great python tutorials.

To interact with an API, specifically a web API in python we can make use of the standard requests module to make the request, because most web service APIs return a response in a format known as JSON it will be useful to know little about it.

> *JSON is a way to store data in an organized, logical manner.*

We wont go into the specifics of JSON, in our analogy it is comparable to the container in which the food is delivered to us. The json module makes it easier to work with the data and it is a built in module in python as well. Python beginners should learn Flask since it provides flexibility and ease of use.

APIs in Python

To work with web API, firstly we need to choose a web service that serves its data over an API. For our example we ll work with Open Notify API. A simple API serving NASA's awesome data.

Lets begin by installing the requests module. If you do not know how to install modules in python please follow the tutorial.

```
pip install requests
```

Once the module is installed, lets import it in a new python file.


If you remember in our analogy that we needed to send a request in order to get a response, in order to retrieve data we can use the **GET** request. Get request takes the URL, in our case the url to Open Notify. Lets make a request and print what is returned. When we just make a request to a url without the right endpoint, we get the html content as response.

> *End points are the location of the resources, by hitting the right end point we can retrieve the data we need.*

```
request = requests.get('http://api.open-notify.org')
print(request.text)
```

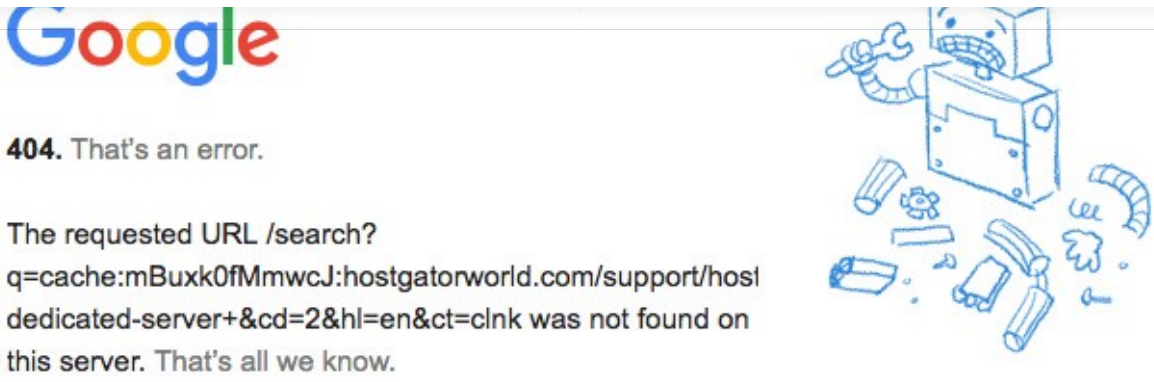
Along with the data we also receive certain statuses, that tells us a bit about the response. For example if a request returns a status code 200 then everything is OK, if it returns 404 then the page or resource was not found. Lets print the status code of the above get request.

```
print(request.status_code)
200
```

200 means everything is **OK**. Lets try to hit a fake end point that does not exist.

```
request2 = requests.get('http://api.open-notify.org/fake-endpoint')
print(request2.status_code)
404
```

As expected we receive 404 error. This image may look familiar.

https://www.seroundtable.com/

This is exactly what happens when we enter the wrong url in the web, internally we are trying to hit an end point that does not exist or the resource was not found at this end point.

Data from International Space Station

Lets take a look at a practical example. Open Notify API serves a couple of end points to access the NASA data which is very cool! The endpoint `/iss-now.json` tells you the exact location of the space station right at this moment. Another `/iss-pass.json` endpoint returns the time at which the space station would do an overhead pass. Another interesting endpoint `/astros.json` returns the number of astronauts in the space. Lets

When we run the above code we print something like this.

> {“people”: [{“name”: “Oleg Kononenko”, “craft”: “ISS”}, {“name”: “David Saint-Jacques”, “craft”: “ISS”}, {“name”: “Anne McClain”, “craft”: “ISS”}], “number”: 3, “message”: “success”}

It may not look very readable but, if we look closely you can see that there are 3 people in space and we can also see their names along with a message “success”.

Lets try to make it more readable, earlier we introduced json, most APIs return a json, luckily **requests** has a built in json decode method, that can turn our data into a native python datatype and make it easier to read. Moreover, Best Python tutorials are a great resource for Python beginners.

```
people_json  = people.json()
print(people_json)
```

Above code molds our response into a python dictionary with key value pairs.

> {‘people’: [{‘name’: ‘Oleg Kononenko’, ‘craft’: ‘ISS’}, {‘name’: ‘David Saint-Jacques’, ‘craft’: ‘ISS’}, {‘name’: ‘Anne McClain’, ‘craft’: ‘ISS’}], ‘number’: 3, ‘message’: ‘success’}

It looks pretty much the same as before, but now we can make use of almighty python to garnish this data.

```
#To print the number of people in space
```

```
    print(p['name'])
```

The result looks something like this! Much better! Your results may differ, because the data is constantly being updated in real time.

> Number of people in space: 3
> Oleg Kononenko
> David Saint-Jacques
> Anne McClain

Train your Vocabulary

In this part we will work with another open API called datamuse. It is a word finding query engine, you can look for words using the api by specifying constraints. You can do things like finding words that starts with a certain letter, that rhymes with a certain word and so on.

At this stage we introduce an important concept known as passing parameters to the url string, also know as **query parameters**. The request we place, is the query and we can

like this.

```
https://api.datamuse.com/words?rel_rhy=jingle
```

Here **words,** is the end point we are hitting and by placing **?** symbol we can apply the constraints, in this case we are asking the API to get the words that rhyme with the word '**jingle**'. According to the documentation in datamuse, **rel_rhy** is keyword indicating to the API to get perfect rhymes for the word specified.Go ahead and copy paste it in your browser and you ll probably see bunch of text with words that rhyme with jingle.

Lets do the same in a more pythonic way. We can define a variable called parameter and pass this along with the request.

```
parameter = {"rel_rhy":"jingle"}
request = requests.get('https://api.datamuse.com/words',parameter)
```

By passing the parameter this way we are doing exactly the same thing we did before, using the url. Now, lets go ahead and print the first 3 words that rhyme with jingle.

```
rhyme_json = request.json()
for i in rhyme_json[0:3]:
 print(i['word'])
```

First we decode the json, and slice this list to print the first 3 words that rhyme with jingle, and my results are:

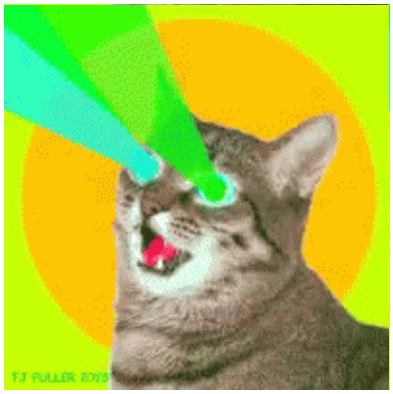
single
mingle
shingle

Do checkout this awesome api, coupled with the concept of query parameters and the


Sending SMS from Python

Lets shift a couple of gears up and try out a practical use case, wherein we can make use of the exceptional twilio api to send an sms from your python program.

If you thought that was cool, hold on, lets take it up another notch, lets try to get the number of people in space using Open Notify and then send it to a friend using twilio. BONKERS!!



tenor.gif

Firstly we gotta take care of some pre-requisites, twilio is a developer platform that allows applications to send voice, video, messages. In order to make this work, we need to follow the steps below:

1. Sign up on twilio and create a free trial account.
2. When you sign up you will be asked to verify your personal number.
3. Once you verify the number you will be asked to create a new project.
4. After you create your project you can access your dashboard and here you can find **AccountSID** that looks something like ACxxxxxxxxxx and an **aunthentication token**. Make a note of these two.
5. You will need to create a twilio free/paid number that can be used to send sms.
6. To do this, in the menu chose Phone Numbers,with trial account you get free credits, use it to buy a phone number.


Lets create the code that will do the trick for us. Firstly we need to install the twilio module.

```
pip install twilio
```

Create a new python file and open it in the editor. Copy the code below into it.

```
from twilio.rest import Client
import requests

account_sid = 'ACxxxxxxxxxxxxxxxxxxxx'
auth_token = 'xxxxxxxxxxxxxxxxxx'

client = Client(account_sid, auth_token)

r = requests.get('http://api.open-notify.org/astros.json')
people = r.json()

number_iss = people['number']

Message = 'Hi Fun fact,Number of people in space right now is
'+str(number_iss)

#formulate the message that will be sent
message = client.messages.create(
    to="+xxxxxxxx",
    from_="+xxxxxxx",
    body=Message)

print(message.sid)
```

There's a lot going in this bite sized snippet, lets go one by one. We start by importing the required modules.

```
account_sid = 'ACxxxxxxxxxxxxxxxxxxxx'
auth_token = 'xxxxxxxxxxxxxxxxxx'
```


```
client = Client(account_sid, auth_token)
```

This step ensures that you are authenticated correctly.

```
r = requests.get('http://api.open-notify.org/astros.json')
people = r.json()

number_iss = people['number']

Message = 'Hi Fun fact,Number of people in space right now is
'+str(number_iss)
```

This is just a reiteration of what we did earlier with Open Notify, we make a request to `/astros.json` endpoint, and decode the json. As we know that the number of people can be retrieved by accessing the [‘number’] key. We store this number and formulate a message for our friend.

```
message = client.messages.create(
    to="+xxxxxxxx",
    from_="+xxxxxxx",
    body=Message)

print(message.sid)
```

Now comes the center piece of our assignment, formulating the message and sending it. In here **to** is the number to which you would like to send a message to, and **from_** is the number that you created inside of twilio(*By default you can only send messages to a verified number, so you can send a message to your number without any problem, if you would like to send it to your friend verify the number in twilio*). Replace the values correctly, and in this step we also indicate what the body of the message should contain, in our case the text from our variable Message:Number of people in space now.

Pheww! Now its time to Hit Save and Execute the program.

Exit

In a nutshell, the process of working with APIs is simple

1. Choose the API to work with
2. Read the API documentation
3. Start with small code, and complement it with more features.

I know we started off as an absolute beginners guide but gradually we built a complex program brick by brick. Trust me, the possibilities around working with APIs are endless. Hopefully you can draw some inspiration from this blog and go slay them APIs.

If the tutorial shifted gears a little too fast, I have great links in the reference section, check it out and you should be abl

1.2K 10

References

1. https://www.programiz.com/python-programming
2. https://www.dataquest.io/blog/python-api-tutorial/
3. https://www.pythonforbeginners.com/api/

Sign up for Developer Updates

By Quick Code

Receive weekly updates about new posts on programming, development, data science, web development and more Take a look.

Your email

Get this newsletter

By signing up, you will create a Medium account if you don't already have one. Review our Privacy Policy for more information about our privacy practices.

Open in app Get started

Get the Medium app