* [Quick Overview of HTTP Requests](https://www.digitalocean.com/community/tutorials/getting-started-with-python-requests-get-requests#quick-overview-of-http-requests)
* [Install Python Requests](https://www.digitalocean.com/community/tutorials/getting-started-with-python-requests-get-requests#install-python-requests)
* [Our First Request](https://www.digitalocean.com/community/tutorials/getting-started-with-python-requests-get-requests#our-first-request)
* [Status Codes](https://www.digitalocean.com/community/tutorials/getting-started-with-python-requests-get-requests#status-codes)
* [Headers](https://www.digitalocean.com/community/tutorials/getting-started-with-python-requests-get-requests#headers)
* [Response Text](https://www.digitalocean.com/community/tutorials/getting-started-with-python-requests-get-requests#response-text)
* [Using the Translate API](https://www.digitalocean.com/community/tutorials/getting-started-with-python-requests-get-requests#using-the-translate-api)
* [Translate API Error Cases](https://www.digitalocean.com/community/tutorials/getting-started-with-python-requests-get-requests#translate-api-error-cases)
* [Conclusion](https://www.digitalocean.com/community/tutorials/getting-started-with-python-requests-get-requests#conclusion)

**Related**

How To Setup uWSGI On Ubuntu 12.10

View

How To Create Nagios Plugins With Python On CentOS 6

View

**// Tutorial //**

**Getting Started With Python Requests - GET Requests**

Published on September 21, 2020 · Updated on September 14, 2020

* [Python](https://www.digitalocean.com/community/tags/python)



By Anthony Herbert

Developer and author at DigitalOcean.



While we believe that this content benefits our community, we have not yet thoroughly reviewed it. If you have any suggestions for improvements, please let us know by clicking the “report an issue“ button at the bottom of the tutorial.

**Introduction**

In many web apps, it’s normal to connect to various third-party services by using APIs. When you use these APIs you can get access to data like weather information, sports scores, movie listings, tweets, search engine results, and pictures. You can also use APIs to add functionality to your app. Examples of these are payments, scheduling, emails, translations, maps, and file transfers. If you were to create any of those on your own it would take a ton of time, but with APIs, it can take only minutes to connect to one and access its features and data.

In this article, we’ll learn about the Python Requests library, which allows you to send HTTP requests in Python.

And since using an API is sending HTTP requests and receiving responses, Requests allows you to use APIs in Python. We’ll demonstrate the use of a language translation API here so you can see an example of how it works.

**Quick Overview of HTTP Requests**

HTTP requests are how the web works. Every time you navigate to a web page, your browser makes multiple requests to the web page’s server. The server then responds with all the data necessary to render the page, and your browser then actually renders the page so you can see it.

The generic process is this: a client (like a browser or Python script using Requests) will send some data to a URL, and then the server located at the URL will read the data, decide what to do with it, and return a response to the client. Finally, the client can decide what to do with the data in the response.

Part of the data the client sends in a request is the request method. Some common request methods are GET, POST, and PUT. GET requests are normally for reading data only without making a change to something, while POST and PUT requests generally are for modifying data on the server. So for example, the Stripe API allows you to use POST requests to create a new charge so a user can purchase something from your app.

**Note:** This article will cover GET requests, because we won’t be modifying any data on a server.

When sending a request from a Python script or inside a web app, you, the developer, gets to decide what gets sent in each request and what to do with the response. So let’s explore that by first sending a request to [Scotch.io](http://Scotch.io) and then by using a language translation API.

**Install Python Requests**

Before we can do anything, we need to install the library. So let’s go ahead and install requests using pip. It’s a good idea to create a virtual environment first if you don’t already have one.

1. pip install requests

**Our First Request**

To start, let’s use Requests for requesting the [Scotch.io](http://Scotch.io) site. Create a file called script.py and add the following code to it. In this article, we won’t have much code to work with, so when something changes you can just update the existing code instead of adding new lines.

script.py

import requests

res = requests.get('https://scotch.io')

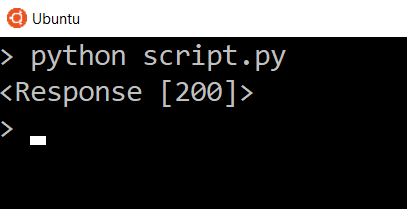
print(res)

So all this code is doing is sending a GET request to [Scotch.io](http://Scotch.io). This is the same type of request your browser sent to view this page, but the only difference is that Requests can’t actually render the HTML, so instead you will just get the raw HTML and the other response information.

We’re using the .get() function here, but Requests allows you to use other functions like .post() and .put() to send those requests as well.

You can run it by executing the [script.py](http://script.py) file.

1. python script.py

And here’s what you get in return: 

**Status Codes**

The first thing we can do is check the status code. HTTP codes range from the 1XX to 5XX. Common status codes that you have probably seen are 200, 404, and 500.

Here’s a quick overview of what each status code means:

* 1XX - Information
* 2XX - Success
* 3XX - Redirect
* 4XX - Client Error (you made an error)
* 5XX - Server Error (they made an error)

Generally, what you’re looking for when you perform your own requests are status codes in the 200s.

Requests recognizes that 4XX and 5XX status codes are errors, so if those status codes get returned, the response object from the request evaluates to False.

You can test if a request responded successfully by checking the response for truth. For example:

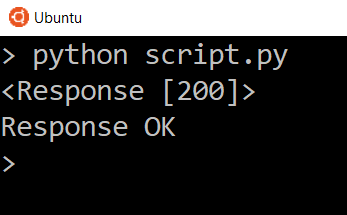
script.py

if res:

print('Response OK')

else:

print('Response Failed')



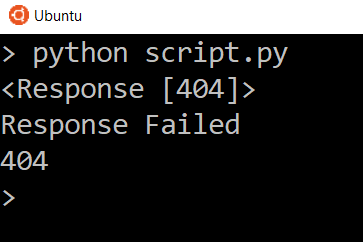
The message “Response Failed” will only appear if a 400 or 500 status code returns. Try changing the URL to some nonsense to see the response fail with a 404.

You can take a look at the status code directly by adding:

script.py

print(res.status\_code)

This will show you the status code directly so you can check the number yourself.



**Headers**

Another thing you can get from the response are the headers. You can take a look at them by using the headers dictionary on the response object.

script.py

print(res.headers)



Headers are sent along with the request and returned in the response. Headers are used so both the client and the server know how to interpret the data that is being sent and received in the response/response.

We see the various headers that are returned. A lot of times you won’t need to use the header information directly, but it’s there if you need it.

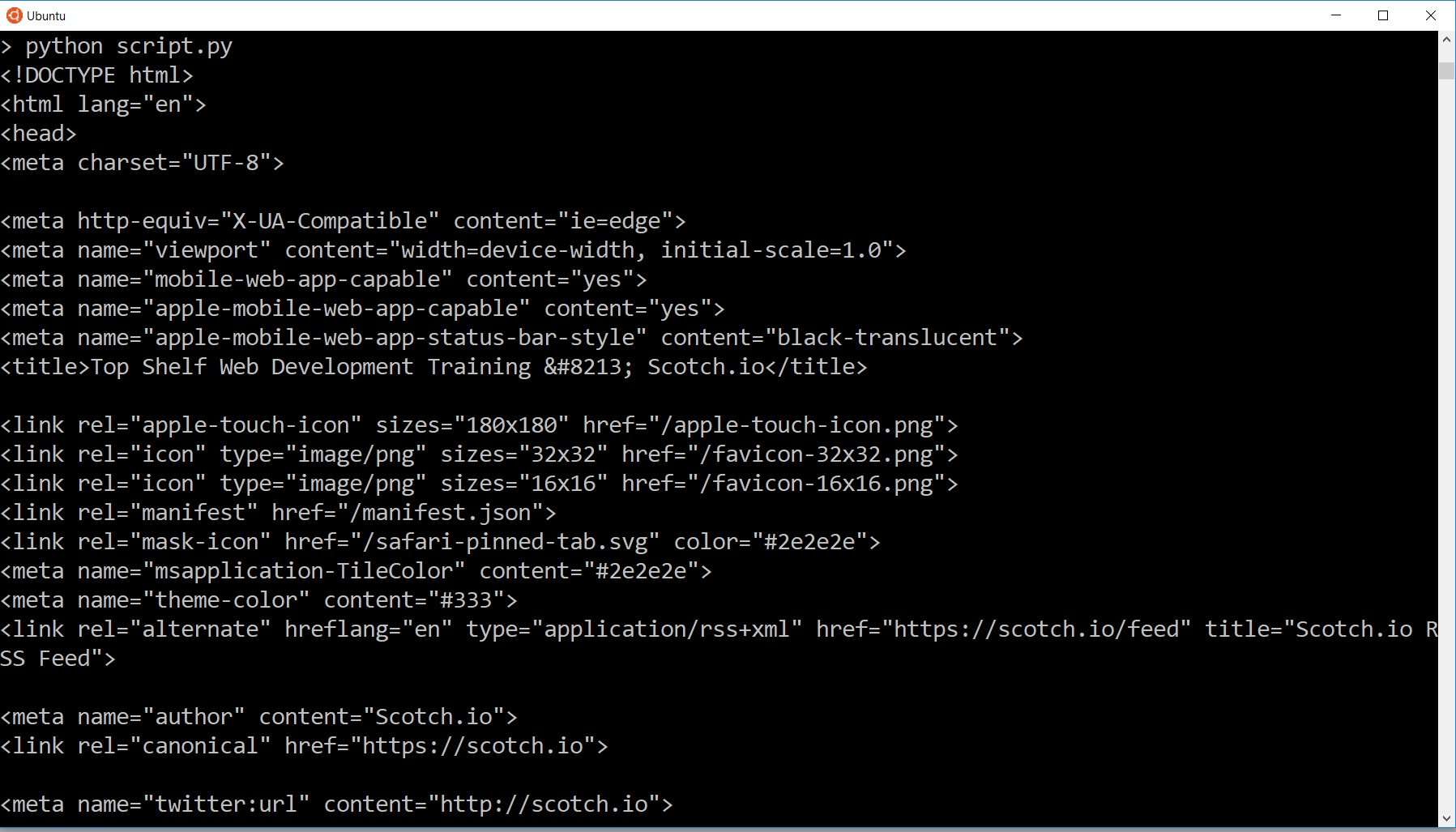
The content type is usually the one you may need because it reveals the format of the data, for example HTML, JSON, PDF, text, etc. But the content type is normally handled by Requests so you can access the data that gets returned.

**Response Text**

And finally, if we take a look at res.text (this works for textual data, like a HTML page like we are viewing) we can see all the HTML needed to build the home page of Scotch. It won’t be rendered, but we see that it looks like it belongs to Scotch. If you saved this to a file and opened it, you would see something that resembled the Scotch site. In a real situation, multiple requests are made for a single web page to load things like images, scripts, and stylesheets, so if you save only the HTML to a file, it won’t look anything like what the [Scotch.io](http://Scotch.io) page looks like in your browser because only a single request was performed to get the HTML data.

script.py

print(res.text)



**Using the Translate API**

So now let’s move on to something more interesting. We’ll use the Yandex Translate API to perform a request to translate some text to a different language.

To use the API, first you need to sign up. After you sign up, go to the Translate API and create an API key. Once you have the API key, add it to your file as a constant. Here’s the link where you can do all those things: <https://tech.yandex.com/translate/>

script.py

API\_KEY = 'your yandex api key'

The reason why we need an API key is so Yandex can authenticate us every time we want to use their API. The API key is a lightweight form of authentication, because it’s added on to the end of the request URL when being sent.

To know which URL we need to send to use the API, we can look at the [documentation for Yandex](https://tech.yandex.com/translate/doc/dg/reference/translate-docpage/).

If we look there, we’ll see all the information needed to use their Translate API to translate text.



When we see a URL with ampersands (&), question marks (?), and equals signs (=), you can be sure that the URL is for GET requests. Those symbols specify the parameters that go along with the URL.

Normally things in square brackets ([]) will be optional. In this case, format, options, and callback are optional, while the key, text, and lang are required for the request.

So let’s add some code to send to that URL. You can replace the first request we created with this:

script.py

url = 'https://translate.yandex.net/api/v1.5/tr.json/translate'

res = requests.get(url)

There are two ways we can add the parameters. We can either append it to the end of the URL directly, or we can have Requests do it for us. To do the latter, we can create a dictionary for our parameters. The three items we need are the key, the text, and the language. Let’s create the dictionary using the API key, 'Hello' for the text, and 'en-es' as the lang, which means we want to translate from English to Spanish.

If you need to know any other language codes, you can look [here](https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes). You are looking for the 639-1 column.

We create a params dictionary by using the dict() function and passing in the keys and values we want in our dictionary.

script.py

params = dict(key=API\_KEY, text='Hello', lang='en-es')

Now we take the parameters dictionary and pass it to the .get() function.

script.py

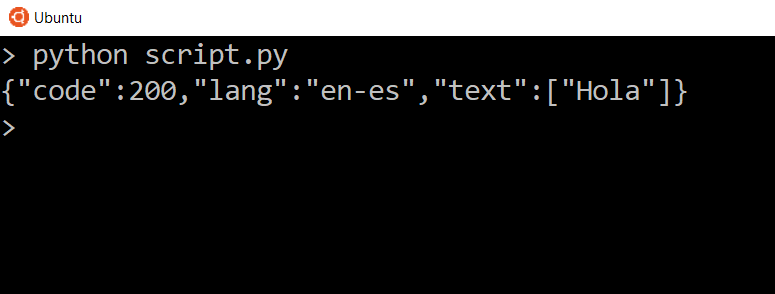
res = requests.get(url, params=params)

When we pass the parameters this way, Requests will go ahead and add the parameters to the URL for us.

Now let’s add a print statement for the response text and view what gets returned in the response.

script.py

print(res.text)



We see three things. We see the status code, which is exactly the same status code of the response itself, we see the language that we specified, and we see the translated text inside of the list. So you should see 'Hola' for the translated text.

Try again with en-fr as the language code, and you should see 'Bonjour' in the response now.

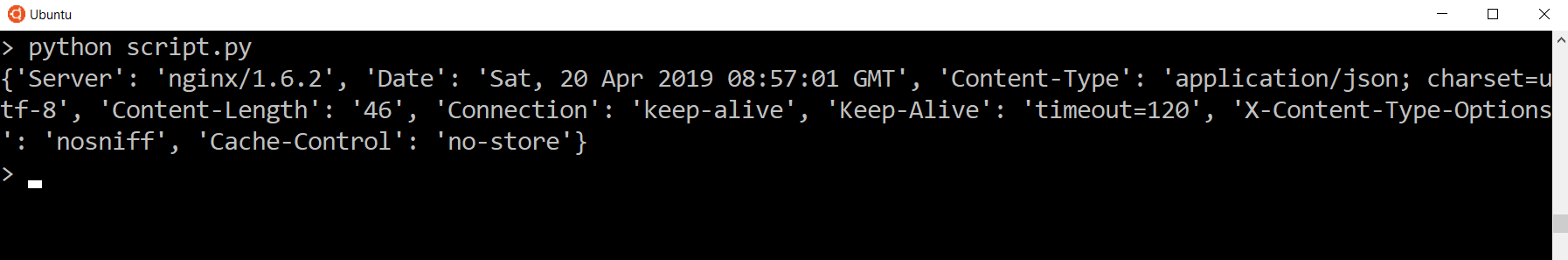
script.py

params = dict(key=API\_KEY, text='Hello', lang='en-fr')

Let’s take a look at the headers for this particular response.

script.py

print(res.headers)



Obviously the headers should be different because we’re communicating with a different server, but in this case the content type is application/json instead of text/html. What this means that the data can be interpreted as JSON.

When application/json is the content type of the response, we are able to have Requests convert the response to a dictionary and list so we can access the data easier.

To have the data parsed as JSON, we use the .json() method on the response object.

If you print it, you’ll see that the data looks the same, but the format is slightly different.

script.py

json = res.json()

print(json)

Graphical user interface, text

Description automatically generatedThe reason why it’s different is because it’s no longer plain text that you get from res.text. This time it’s a printed version of a dictionary.

Let’s say we want to access the text. Since this is now a dictionary, we can use the text key.

script.py

print(json['text'])

Text

Description automatically generatedAnd now we only see the data for that one key. In this case we are looking at a list of one item, so if we wanted to get that text in the list directly, we can access it by the index.

script.py

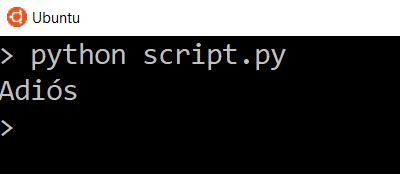
print(json['text'][0])

And now the only thing we see is the translated word.

So of course if we change things in our parameters, we’ll get different results. Let’s change the text to be translated from Hello to Goodbye, change the target language back to Spanish, and send the request again.

script.py

params = dict(key=API\_KEY, text='Goodbye', lang='en-es')

Try translating longer text in different languages and see what responses the API gives you.

**Translate API Error Cases**

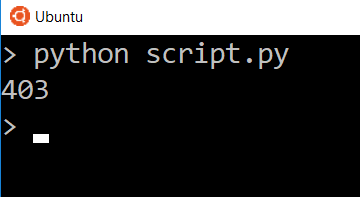
Finally, we’ll take a look at an error case. Everything doesn’t always work, so we need to know when that happens.

Try changing your API key by removing one character. When you do this your API key will no longer be valid. Then try sending a request.

If you take a look at the status code, this is what you get:

script.py

print(res.status\_code)

So when you are using the API, you’ll want to check if things are successful or not so you can handle the error cases according to the needs of your app.

**Conclusion**

Here’s what we learned:

* How HTTP requests work
* The various status codes possible in a response
* How to send requests and receive responses using the Python Requests library
* How to use a language translation API to translate text
* How to convert application/JSON content responses to dictionaries
* [Start Here](javascript:void(0);) 
  + [Machine Learning](https://courses.analyticsvidhya.com/courses/Machine-Learning-Certification-Course-for-Beginners?utm_source=blog_navbar&utm_medium=start_here_button)
  + [Deep Learning](https://courses.analyticsvidhya.com/courses/getting-started-with-neural-networks?utm_source=blog_navbar&utm_medium=start_here_button)
  + [NLP](https://courses.analyticsvidhya.com/courses/Intro-to-NLP?utm_source=blog_navbar&utm_medium=start_here_button)
* [Articles](javascript:void(0);)
  + [Guides](https://www.analyticsvidhya.com/blog/category/guide/?utm_source=blog_navbar&utm_medium=machine_learning_button)
  + [Machine Learning](https://www.analyticsvidhya.com/blog/category/machine-learning/?utm_source=blog_navbar&utm_medium=machine_learning_button)
  + [Deep Learning](https://www.analyticsvidhya.com/blog/category/deep-learning/?utm_source=blog_navbar&utm_medium=deep_learning_button)
  + [NLP](https://www.analyticsvidhya.com/blog/category/nlp/?utm_source=blog_navbar&utm_medium=_button)
  + [Computer Vision](https://www.analyticsvidhya.com/blog/category/computer-vision/?utm_source=blog_navbar&utm_medium=article_button)
  + [Data Visualization](https://www.analyticsvidhya.com/blog/category/data-visualization/?utm_source=blog_navbar&utm_medium=_button)
  + [Interview Questions](https://www.analyticsvidhya.com/blog/category/interview-questsions/?utm_source=blog_navbar&utm_medium=career_button)
  + [More](javascript:void(0);)

[Courses](javascript:void(0);)

* [Certified AI & ML BlackBelt Plus](https://blackbelt.analyticsvidhya.com/plus?utm_source=blog_navbar&utm_medium=blackbelt_button)
* [Data Science Immersive Bootcamp](https://bootcamp.analyticsvidhya.com/?utm_source=blog_navbar&utm_medium=bootcamp_button)
* [Blogathon](https://datahack.analyticsvidhya.com/blogathon/?utm_source=blog&utm_medium=nav_bar)
* [DataHour](https://datahack.analyticsvidhya.com/datahour/?utm_source=datahack&utm_medium=Navbar)



 50+ Exciting Industry Projects to become a Full-Stack Data Scientist

1. [Home](https://www.analyticsvidhya.com/blog/)

# An Introduction to APIs (Application Programming Interfaces) & 5 APIs a Data Scientist must know!

[Sauravkaushik8 Kaushik](https://www.analyticsvidhya.com/blog/author/sauravkaushik8/) — Published On November 18, 2016 and Last Modified On July 5th, 2020

[Data Mining](https://www.analyticsvidhya.com/blog/category/data-mining/) [Intermediate](https://www.analyticsvidhya.com/blog/category/intermediate/) [Listicle](https://www.analyticsvidhya.com/blog/category/listicle/) [Machine Learning](https://www.analyticsvidhya.com/blog/category/machine-learning/) [Research & Technology](https://www.analyticsvidhya.com/blog/category/research-technology/) [Resource](https://www.analyticsvidhya.com/blog/category/resource/)

## Introduction

If you are in tech domain, you will invariably bump in references to something called an “API”. You just can’t skip it – if you do, you are bound to hear it again. APIs are being used almost everywhere. But, if you have ever wondered what exactly is an API? or Why are they important? or How do they help? this article will help you out.

In this article, I will explain what is an API in simple terms. I will tell various categories / types of API. Then I will be introducing you to the different APIs which you commonly encounter in your day to day life. To add more value, I have listed down 5 useful projects you can work on using an API. I bet you will be tempted to try your hands on at least one of them.

Let’s get started!

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/11/api.jpg)

## Table of Contents

1. Overview
2. Categories of API
3. Difference between an API and a Library
4. Walk through an example
5. 5 APIs every Data Scientists should know
6. List of 5 cool data science projects using APIs
7. Welcome to the new playground

Graphical user interface, application

Description automatically generated

##### Kickstart your Data Science journey | Get guidance from industry experts

## 1. Overview

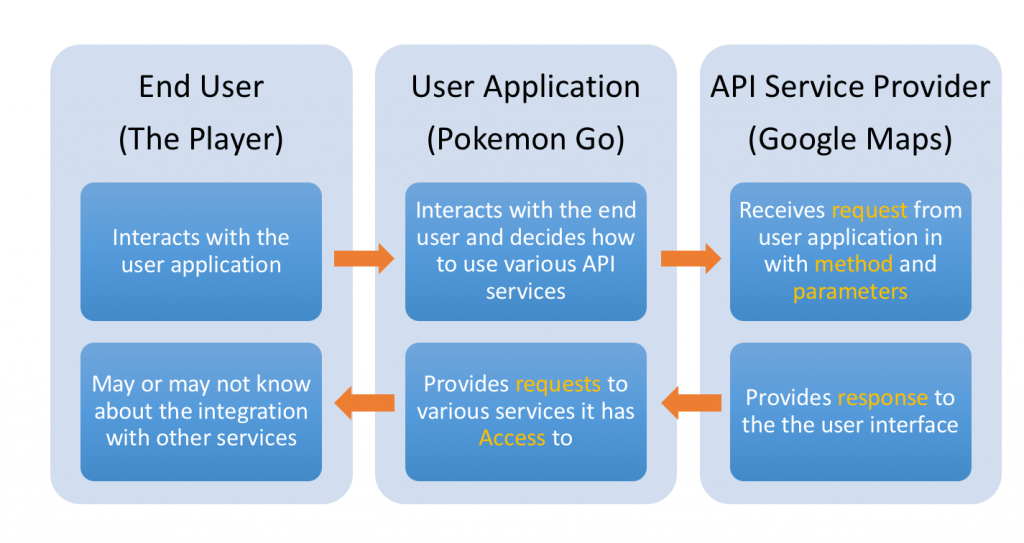
In simple words, an API is a (hypothetical) contract between 2 softwares saying if the user software provides input in a pre-defined format, the later with extend its functionality and provide the outcome to the user software. Think of it like this, Graphical user interface (GUI) or command line interface (CLI) allows humans to Interact with code, where as an Application programmable interface (API) allows one piece of code to interact with other code.

One of the most common use case for APIs is on the web. If you have spent a few hours on internet, you have certainly used APIs. Sharing things on social media, making payments over the web, displaying list of tweets through a social handle – all of these services use API at the back.

APIs are widely used by developers for implementing various features in their software. They simply use a simple API call within their software to implement complex features instead of having to code it by themselves.

Let’s try and understand it better with the help of an example:

Pokemon Go has been one of the most popular smartphone games. But in order to build such a game taking in account the large ecosystem, one requires complete information of routes and roads across the globe. I’m sure the developers of the Pokemon Go must have faced a dilemma if they should code the maps of the entire world or use the  existing Google maps to build their application on top of it. They choose the latter, simply because it’s practically  not possible to create something similar to Google maps in a short span of time.

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/11/Screen-Shot-2016-11-18-at-10.20.04-AM.png)

This is just one example. There are a lot of developers using various APIs to implement complex features into their applications instead of coding it themselves. Therefore, API provides a very convenient way of making code reusable.

**Basic elements of an API:**

An API has three primary elements:

* **Access**: is the user or who is allowed to ask for data or services?
* **Request**: is the actual data or service being asked for (e.g., if I give you current location from my game (Pokemon Go), tell me the map around that place).  A Request has two main parts:
  + **Methods**: i.e. the questions you can ask, assuming you have access (it also defines the type of responses available).
  + **Parameters**: additional details you can include in the question or response.
* **Response**: the data or service as a result of your request.

## 2. Categories of API

### Web-based system

A web API is an interface to either a web server or a web browser. These APIs are used extensively for the development of web applications. These APIs work at either the server end or the client end. Companies like Google, Amazon, eBay all provide web-based API.

Some popular examples of web based API are Twitter REST API, Facebook Graph API, Amazon S3 REST API, etc.

### Operating system

There are multiple OS based API that offers the functionality of various OS features that can be incorporated in creating windows or mac applications.

Some of the examples of OS based API are Cocoa, Carbon, WinAPI, etc.

### Database system

Interaction with most of the database is done using the API calls to the database. These APIs are defined in a manner to pass out the requested data in a predefined format that is understandable by the requesting client.

This makes the process of interaction with databases generalised and thereby enhancing the compatibility of applications with the  various database. They are very robust and provide a structured interface to database.

Some popular examples are Drupal 7 Database API, Drupal 8 Database API, Django API.

### Hardware System

These APIs allows access to the various hardware components of a system. They are extremely crucial for establishing communication to the hardware. Due to which it makes possible for a range of functions from the collection of sensor data to even display on your screens.

For example, the Google PowerMeter API will allow device manufacturers to build home energy monitoring devices that work with Google PowerMeter.

Some other examples of Hardware APIs are: QUANT Electronic, WareNet CheckWare,OpenVX Hardware Acceleration, CubeSensore, etc.

## 3. Difference between an API and a Library

At this point, I believe you might be scratching your head and confusing APIs with libraries. Let me simplify it for you, an application programming interface (API) is an interface that defines the way by which an application program may request service from the  libraries.

An API is a set of rules with which the interaction between various entities is defined. We are specifically talking about interaction between two software.

Even a library also has an API which denotes the area of the library which is actually accessible to the user from outside.

## 4. Walk through an example

IBM Watson has made certain data science APIs public for people like us to build amazing projects with only a few lines of code. Here, we’ll be looking at one such amazing API offered by IBM called Personality Insights.

This API takes as input in JSON, HTML or simple text format. The input contains text related to the person whose personality interests you. It can be anything like tweets, daily experiences, applications, opinion, etc of that person.

The output generated by the API is in the standard format of JSON or CSV file that contains the information on various social traits of that person. And the developer only needs to display this generated file to the user instead of coding the whole functionality yourself.

There is also a demo on the IBM website that can be accessed [here](https://personality-insights-livedemo.mybluemix.net/). You can choose either the tweets or replies of few famous personalities to analyze their personality traits. The text can also be customized based on what input you want to provide and analyze the personality traits of that person.

You can integrate this API in your code as well and build an application on top of this API.

## 5. 5 APIs every Data Scientists should know

### Facebook API

Facebook API provides an interface to a large amount of data generated everyday. The innumerable post, comments and shares in various groups & pages produces massive data. And this massive public data provides a large number of opportunities for analyzing the crowd.

It is also incredibly convenient to use Facebook Graph API with both R and python to extract data. To read more about the Facebook API, [click here](https://developers.facebook.com/docs/graph-api).

### Google Map API

Google Map API is one of the commonly used API. Its applications vary from integration in a cab service application to the popular Pokemon Go.

You can retrieve all the information like location coordinates, distances between locations, routes etc. The fun part is that you can also use this API for creating the distance feature in your datasets as well. [Read here](https://www.analyticsvidhya.com/blog/2015/03/hacking-google-maps-create-distance-features-model-applications/) to find out its complete implementation.

### Twitter API

Just like Facebook Graph API, Twitter data can be accessed using the Twitter API as well. You can access all the data like tweets made by any user, the tweets containing a particular term or even a combination of terms, tweets done on the topic in a particular date range, etc.

Twitter data is a great resource for performing the tasks like opinion mining, sentiment analysis. For detailed usage of twitter API, [read here](https://www.analyticsvidhya.com/blog/2014/11/text-data-cleaning-steps-python/).

### IBM Watson API

IBM Watson offers a set of APIs for performing a host of complex tasks such as Tone analyzer, document conversion, personality insights, visual recognition, text to speech, speech to text, etc by using just few lines of code.

This set of APIs differ from the other APIs discussed so far, as they provide service for manipulating and deriving insights from the data. To know indepth details about this API, [read here](http://www.ibm.com/watson/developercloud/services-catalog.html).

### Quandl API

Quandl lets you invoke the time series information of a large number of stocks for the specified date range. The setting up of Quandl API is very easy and provides a great resource for projects like Stock price prediction, stock profiling, etc. [Click here](https://www.quandl.com/), to read more details about Quandl API.

## 6. List of 5 cool data science projects using API

I am sure that you are fascinated after reading about the above APIs, but wondering if you could a create project using these APIs which will be great value add to your CV?  Well, here’s the list of ideas you can start with. You can either use these APIs to retrieve data & manipulate it to extract insights from it or pass the data to these APIs & perform complex functions.

Here are the list of the projects for you. I’ll leave the execution of these ideas to you.

* **Social Media Sentiment Analysis** : By using data from Twitter and Facebook API.
* **Opinion Mining** : By using data from Twitter and Facebook API.
* **Stock Prediction** : By using data from Yahoo Stock API and Quandl API.
* **Most Popular languages on Github** : By using data from Github API.
* **Microsoft Face Sentiment Recognition** : By using Microsoft face API.

Further reads:

* [Using the meetup API to find right meetup groups](https://www.analyticsvidhya.com/blog/2015/08/data-scientist-meetup-hack/)
* [Mining YouTube for social media analysis](https://www.analyticsvidhya.com/blog/2014/09/mining-youtube-python-social-media-analysis/)
* [Subject extraction using Google API](https://www.analyticsvidhya.com/blog/2015/03/text-mining-subject-extraction-google-api/)
* [Use Google Maps API to create distance features](https://www.analyticsvidhya.com/blog/2015/03/hacking-google-maps-create-distance-features-model-applications/)

## 7. Welcome to the new Playground

Take a step back –  you have just got a glimpse of an entirely new world. Think of all the possibilities it enables – need face recognition on your mobile application – no worries! just invoke Google Face recognition API. Need to translate documents in Japanese to English – why not try Google Translate! The possibilities are limitless!

A few things to keep in mind while you think of building and using APIs:

* **APIs are a robust way to integrate applications**. You can theoretically do integrations without APIs in some cases, but that often lands you in difficulty. For example, you want to pull 5 latest tweets from a user. You can do this easily through Twitter API and irrespective of changes happening at the front end of Twitter your results would be correct. If you would have done this using web scrapping, you would need to make changes with every change in web site layout. You would need to worry in case Twitter throws an advertisement in between the last 5 tweets and filter it out!
* **APIs might come with a limit on usage**. This could be in form of daily limits, number of requests in a time frame etc. This is to ensure that the servers of the service providers are under predictable loads. If you need higher access, check if there is a paid access or talk to the service provider.
* **While designing your application, think of what is the best way to use an API**. For example, you can send a single request to YouTube and get the top videos with some statistics. On the other hand, you can send one request for each video and get more detailed statistics. What do you need for your application – if you can do what is needed in one call – do it with one call!

## End Notes

After going through this article, I believe you would have acquired a better understanding of APIs and how helpful they can be. I have only mentioned some of the popular APIs but the list is endless. If you would like to add to the list of APIs, please share them in the comments below.

I’ll encourage you to pick up any of the suggested projects and work on it. I bet you’ll be shocked by the power and ease with which you’ll be able to perform a complex task that would otherwise have been difficult to implement yourself.