IBM Developer
SKILLS NETWORK

Application Programming Interface

Estimated time needed: **15** minutes

Objectives

After completing this lab you will be able to:

- Create and Use APIs in Python

Introduction

An API lets two pieces of software talk to each other. Just like a function, you don’t have to know how the API works only its inputs and outputs. An essential type of API is a REST API that allows you to access resources via the internet. In this lab, we will review the Pandas Library in the context of an API, we will also review a basic REST API

Table of Contents

In [1]:

```
!pip install nba_api
```

```
Collecting nba_api
  Downloading https://files.pythonhosted.org/packages/f0/07/d32f5106c95fbe
e8e54b22d2795f94c2d2213ed6d2e5caac390b56667d37/nba_api-1.1.9-py3-none-any.
whl (242kB)
     |████████████████████████████████| 245kB 7.3MB/s eta 0:00:01
Requirement already satisfied: requests in /home/jupyterlab/conda/envs/pyt
hon/lib/python3.6/site-packages (from nba_api) (2.25.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /home/jupyterlab/cond
a/envs/python/lib/python3.6/site-packages (from requests->nba_api) (3.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/jupyterlab/c
onda/envs/python/lib/python3.6/site-packages (from requests->nba_api) (1.2
5.11)
Requirement already satisfied: certifi>=2017.4.17 in /home/jupyterlab/cond
a/envs/python/lib/python3.6/site-packages (from requests->nba_api) (2020.1
2.5)
Requirement already satisfied: idna<3,>=2.5 in /home/jupyterlab/conda/env
s/python/lib/python3.6/site-packages (from requests->nba_api) (2.10)
Installing collected packages: nba-api
Successfully installed nba-api-1.1.9
```

Pandas is an API

You will use this function in the lab:

In [2]:

```
def one_dict(list_dict):
    keys=list_dict[0].keys()
    out_dict={key:[] for key in keys}
    for dict_ in list_dict:
        for key, value in dict_.items():
            out_dict[key].append(value)
    return out_dict
```

Pandas is an API

Pandas is actually set of software components , much of which is not even written in Python.

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
```

You create a dictionary, this is just data.

In [5]:

```
dict_={'a':[11,21,31],'b':[12,22,32]}
```

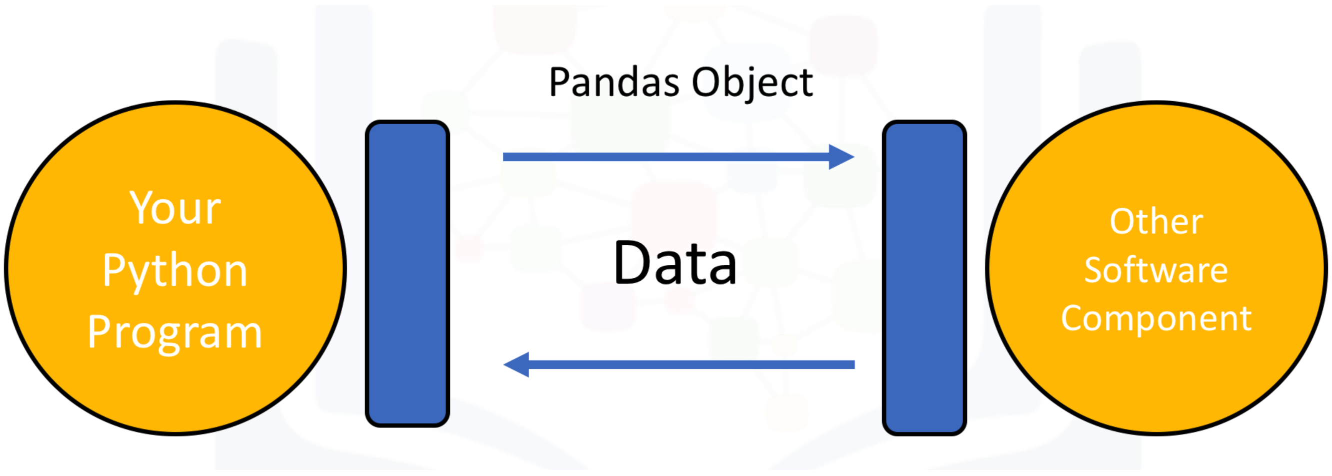
When you create a Pandas object with the Dataframe constructor in API lingo, this is an "instance". The data in the dictionary is passed along to the pandas API. You then use the dataframe to communicate with the API.

In [6]:

```
df=pd.DataFrame(dict_)
type(df)
```

Out[6]:

```
pandas.core.frame.DataFrame
```



When you call the method head the dataframe communicates with the API displaying the first few rows of the dataframe.

In [7]:

```
df.head()
```

Out[7]:

	a	b
0	11	12
1	21	22
2	31	32

When you call the method mean,the API will calculate the mean and return the value.

In [8]:

```
df.mean()
```

Out[8]:

```
a    21.0
b    22.0
dtype: float64
```

REST APIs

Rest API’s function by sending a **request**, the request is communicated via HTTP message. The HTTP message usually contains a JSON file. This contains instructions for what operation we would like the service or **resource** to perform. In a similar manner, API returns a **response**, via an HTTP message, this response is usually contained within a JSON.

In this lab, we will use the NBA API (https://pypi.org/project/nba-api/) to determine how well the Golden State Warriors performed against the Toronto Raptors. We will use the API do the determined number of points the Golden State Warriors won or lost by for each game. So if the value is three, the Golden State Warriors won by three points. Similarly it the Golden State Warriors lost by two points the result will be negative two. The API is relatively will handle a lot of the details such a Endpoints and Authentication

In the nba api to make a request for a specific team, it's quite simple, we don't require a JSON all we require is an id. This information is stored locally in the API we import the module teams

In [9]:

```
from nba_api.stats.static import teams
import matplotlib.pyplot as plt
```

In [ ]:

```
#https://pypi.org/project/nba-api/
```

The method `get_teams()` returns a list of dictionaries the dictionary key id has a unique identifier for each team as a value

In [10]:

```
nba_teams = teams.get_teams()
```

The dictionary key id has a unique identifier for each team as a value, let's look at the first three elements of the list:

In [11]:

```
nba_teams[0:3]
```

Out[11]:

```
[{'id': 1610612737,
  'full_name': 'Atlanta Hawks',
  'abbreviation': 'ATL',
  'nickname': 'Hawks',
  'city': 'Atlanta',
  'state': 'Atlanta',
  'year_founded': 1949},
 {'id': 1610612738,
  'full_name': 'Boston Celtics',
  'abbreviation': 'BOS',
  'nickname': 'Celtics',
  'city': 'Boston',
  'state': 'Massachusetts',
  'year_founded': 1946},
 {'id': 1610612739,
  'full_name': 'Cleveland Cavaliers',
  'abbreviation': 'CLE',
  'nickname': 'Cavaliers',
  'city': 'Cleveland',
  'state': 'Ohio',
  'year_founded': 1970}]
```

To make things easier, we can convert the dictionary to a table. First, we use the function `one dict` , to create a dictionary. We use the common keys for each team as the keys, the value is a list; each element of the list corresponds to the values for each team. We then convert the dictionary to a dataframe, each row contains the information for a different team.

In [12]:

```
dict_nba_team=one_dict(nba_teams)
df_teams=pd.DataFrame(dict_nba_team)
df_teams.head()
```

Out[12]:

	id	full_name	abbreviation	nickname	city	state	year_founded
0	1610612737	Atlanta Hawks	ATL	Hawks	Atlanta	Atlanta	1949
1	1610612738	Boston Celtics	BOS	Celtics	Boston	Massachusetts	1946
2	1610612739	Cleveland Cavaliers	CLE	Cavaliers	Cleveland	Ohio	1970
3	1610612740	New Orleans Pelicans	NOP	Pelicans	New Orleans	Louisiana	2002
4	1610612741	Chicago Bulls	CHI	Bulls	Chicago	Illinois	1966

Will use the team's nickname to find the unique id, we can see the row that contains the warriors by using the column nickname as follows:

In [13]:

```
df_warriors=df_teams[df_teams['nickname']=='Warriors']
df_warriors
```

Out[13]:

	id	full_name	abbreviation	nickname	city	state	year_founded
7	1610612744	Golden State Warriors	GSW	Warriors	Golden State	California	1946

we can use the following line of code to access the first column of the dataframe:

In [14]:

```
id_warriors=df_warriors[['id']].values[0][0]
#we now have an integer that can be used   to request the Warriors information
id_warriors
```

Out[14]:

```
1610612744
```

The function "League Game Finder " will make an API call, its in the module `stats.endpoints`

In [15]:

```
from nba_api.stats.endpoints import leaguegamefinder
```

The parameter `team_id_nullable` is the unique ID for the warriors. Under the hood, the NBA API is making a HTTP request.
The information requested is provided and is transmitted via an HTTP response this is assigned to the object `gamefinder`.

In [ ]:

```
# Since https://stats.nba.com does not allow api calls from Cloud IPs and Skills Networ
k Labs uses a Cloud IP.
# The following code is comment out, you can run it on jupyter labs on your own compute
r.
# gamefinder = leaguegamefinder.LeagueGameFinder(team_id_nullable=id_warriors)
```

we can see the json file by running the following line of code.

In [ ]:

```
# Since https://stats.nba.com does not allow api calls from Cloud IPs and Skills Networ
k Labs uses a Cloud IP.
# The following code is comment out, you can run it on jupyter labs on your own compute
r.
# gamefinder.get_json()
```

The game finder object has a method `get_data_frames()` , that returns a dataframe. If we view the dataframe, we can see it contains information about all the games the Warriors played. The `PLUS_MINUS` column contains information on the score, if the value is negative the Warriors lost by that many points, if the value is positive, the warriors one by that amount of points. The column `MATCHUP` had the team the Warriors were playing, GSW stands for Golden State Warriors and TOR means Toronto Raptors; `vs` signifies it was a home game and the <code>@ </code>symbol means an away game.

In [ ]:

```
# Since https://stats.nba.com does not allow api calls from Cloud IPs and Skills Networ
k Labs uses a Cloud IP.
# The following code is comment out, you can run it on jupyter labs on your own compute
r.
# games = gamefinder.get_data_frames()[0]
# games.head()
```

you can download the dataframe from the API call for Golden State and run the rest like a video.

In [16]:

```
! wget https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClas
s/PY0101EN/Chapter%205/Labs/Golden_State.pkl
```

```
--2021-01-16 13:01:23--  https://s3-api.us-geo.objectstorage.softlayer.ne
t/cf-courses-data/CognitiveClass/PY0101EN/Chapter%205/Labs/Golden_State.pk
l
Resolving s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objects
torage.softlayer.net)... 67.228.254.196
Connecting to s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.obj
ectstorage.softlayer.net)|67.228.254.196|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 811065 (792K) [application/octet-stream]
Saving to: ‘Golden_State.pkl’

Golden_State.pkl    100%[===================>] 792.06K  3.30MB/s    in 0.2
s

2021-01-16 13:01:23 (3.30 MB/s) - ‘Golden_State.pkl’ saved [811065/811065]
```

In [18]:

```
file_name = "Golden_State.pkl"
games = pd.read_pickle(file_name)
games.head()
```

Out[18]:

	SEASON_ID	TEAM_ID	TEAM_ABBREVIATION	TEAM_NAME	GAME_ID	GAME_DATE
0	22019	1610612744	GSW	Golden State Warriors	1521900066	2019-07-12
1	22019	1610612744	GSW	Golden State Warriors	1521900058	2019-07-10
2	22019	1610612744	GSW	Golden State Warriors	1521900039	2019-07-08
3	22019	1610612744	GSW	Golden State Warriors	1521900020	2019-07-07
4	22019	1610612744	GSW	Golden State Warriors	1521900007	2019-07-05

5 rows × 28 columns

We can create two dataframes, one for the games that the Warriors faced the raptors at home and the second for away games.

In [19]:

```
games_home=games [games ['MATCHUP']=='GSW vs. TOR']
games_away=games [games ['MATCHUP']=='GSW @ TOR']
```

We can calculate the mean for the column `PLUS_MINUS` for the dataframes `games_home` and `games_away` :

In [20]:

```
games_home.mean()['PLUS_MINUS']
```

Out[20]:

```
3.730769230769231
```

In [21]:

```
games_away.mean()['PLUS_MINUS']
```

Out[21]:

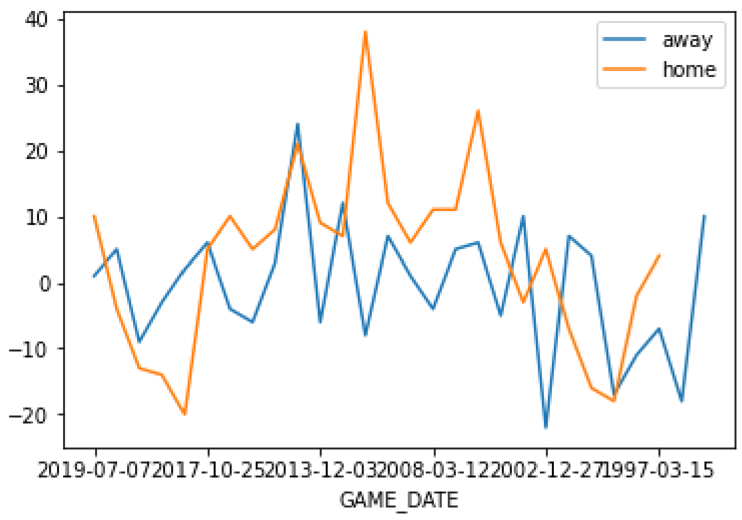
```
-0.6071428571428571
```

We can plot out the `PLUS MINUS` column for for the dataframes `games_home` and `games_away` . We see the warriors played better at home.

In [22]:

```
fig, ax = plt.subplots()

games_away.plot(x='GAME_DATE',y='PLUS_MINUS', ax=ax)
games_home.plot(x='GAME_DATE',y='PLUS_MINUS', ax=ax)
ax.legend(["away", "home"])
plt.show()
```



Quiz

Calculate the mean for the column `PTS` for the dataframes `games_home` and `games_away` :

In [23]:

```
# Write your code below and press Shift+Enter to execute
games_home.mean()['PTS']

games_away.mean()['PLUS_MINUS']
```

Out[23]:

```
-0.6071428571428571
```

▶ Click here for the solution

Learn

Get started or get better with built-in learning.

Create

Use the best of open source tooling with IBM innovation.

Collaborate

Work smarter using community, work faster with your team.

Sign Up For a Free Trial

(https://cloud.ibm.com/catalog/services/watson-studio)

Authors:

Joseph Santarcangelo (https://www.linkedin.com/in/joseph-s-50398b136?cm_mmc=Email_Newsletter-_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBMDeveloperSkillsNetwork-PY0101EN-SkillsNetwork-19487395&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=email.Newslette
_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBMDeveloperSkillsNetwork-PY0101EN-SkillsNetwork-19487395&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=email.Newslette

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-09-09	2.1	Malika Singla	Spell Check
2020-08-26	2.0	Lavanya	Moved lab to course repo in GitLab