TABLE OF CONTENTS

6-Python API- June 22 Summary

Class objectives:

- Students will create applications from scratch using nothing but their knowledge of Python and an API documentation
- Students will load JSON from API responses into a Pandas DataFrame
- Students will be able to use try and except blocks to handle errors

Exercise 01 - JSON Traversal Review

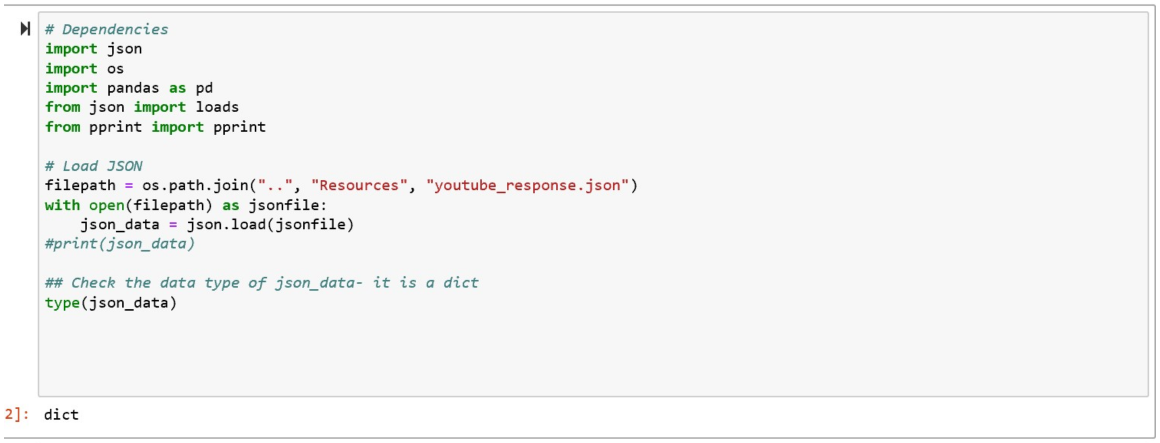
This activity we to practice load and parse JSON in Python.

Instructions

- Load the provided JSON
- Retrieve the video's title
- Retrieve the video's rating
- Retrieve the link to the video's thumbnail
- Retrieve the number of views this video has

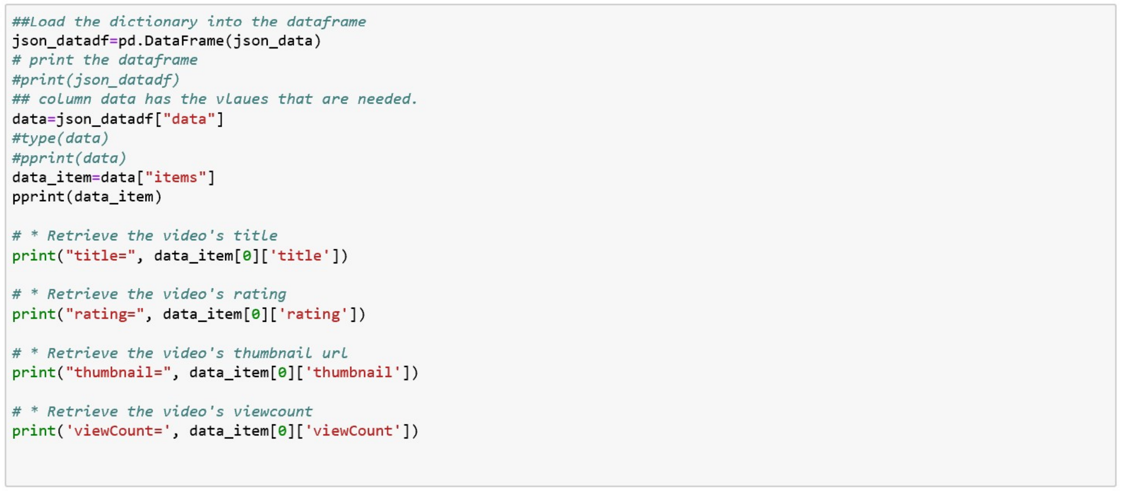
Divya's Solution Code:

Load the JSON file and check the data type.



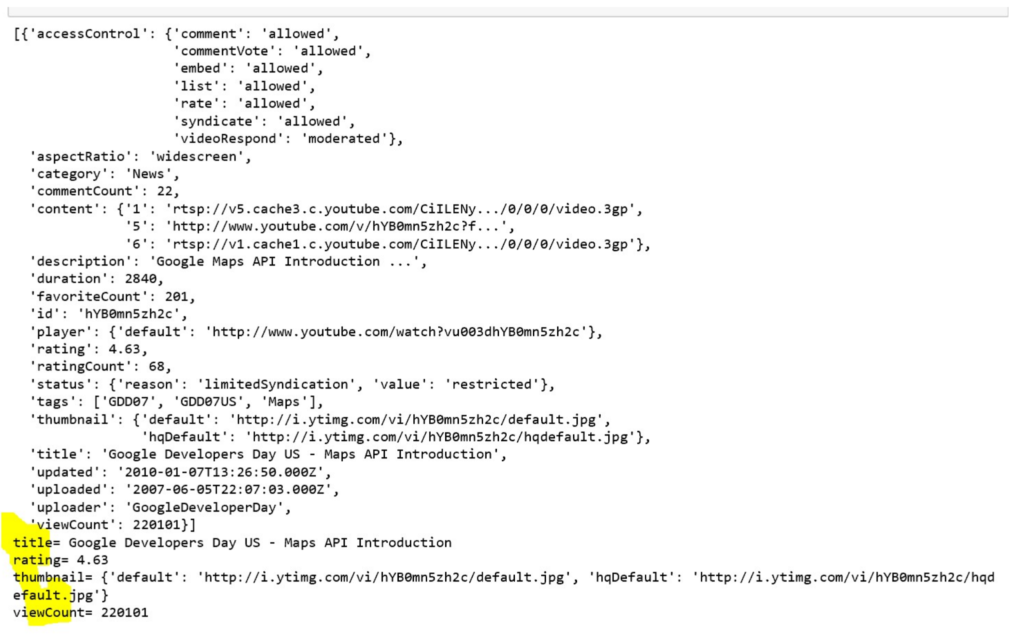
```
# Dependencies
import json
import os
import pandas as pd
from json import loads
from pprint import pprint

# Load JSON
filepath = os.path.join("..", "Resources", "youtube_response.json")
with open(filepath) as jsonfile:
    json_data = json.load(jsonfile)
#print(json_data)

## Check the data type of json_data- it is a dict
type(json_data)
```

```
2]: dict
```



```
##Load the dictionary into the dataframe
json_datadf=pd.DataFrame(json_data)
# print the dataframe
#print(json_datadf)
## column data has the vlaues that are needed.
data=json_datadf["data"]
#type(data)
#pprint(data)
data_item=data["items"]
pprint(data_item)

# * Retrieve the video's title
print("title=", data_item[0]['title'])

# * Retrieve the video's rating
print("rating=", data_item[0]['rating'])

# * Retrieve the video's thumbnail url
print("thumbnail=", data_item[0]['thumbnail'])

# * Retrieve the video's viewcount
print('viewCount=', data_item[0]['viewCount'])
```

Output of the calculations:



```
[{'accessControl': {'comment': 'allowed',
                    'commentVote': 'allowed',
                    'embed': 'allowed',
                    'list': 'allowed',
                    'rate': 'allowed',
                    'syndicate': 'allowed',
                    'videoRespond': 'moderated'},
  'aspectRatio': 'widescreen',
  'category': 'News',
  'commentCount': 22,
  'content': {'1': 'rtsp://v5.cache3.c.youtube.com/CiILENy.../0/0/0/video.3gp',
              '5': 'http://www.youtube.com/v/hYB0mn5zh2c?f...',
              '6': 'rtsp://v1.cache1.c.youtube.com/CiILENy.../0/0/0/video.3gp'},
  'description': 'Google Maps API Introduction ...',
  'duration': 2840,
  'favoriteCount': 201,
  'id': 'hYB0mn5zh2c',
  'player': {'default': 'http://www.youtube.com/watch?vu003dhYB0mn5zh2c'},
  'rating': 4.63,
  'ratingCount': 68,
  'status': {'reason': 'limitedSyndication', 'value': 'restricted'},
  'tags': ['GDD07', 'GDD07US', 'Maps'],
  'thumbnail': {'default': 'http://i.ytimg.com/vi/hYB0mn5zh2c/default.jpg',
                'hqDefault': 'http://i.ytimg.com/vi/hYB0mn5zh2c/hqdefault.jpg'},
  'title': 'Google Developers Day US - Maps API Introduction',
  'updated': '2010-01-07T13:26:50.000Z',
  'uploaded': '2007-06-05T22:07:03.000Z',
  'uploader': 'GoogleDeveloperDay',
  'viewCount': 220101}]
title= Google Developers Day US - Maps API Introduction
rating= 4.63
thumbnail= {'default': 'http://i.ytimg.com/vi/hYB0mn5zh2c/default.jpg', 'hqDefault': 'http://i.ytimg.com/vi/hYB0mn5zh2c/hqd
efault.jpg'}
viewCount= 220101
```

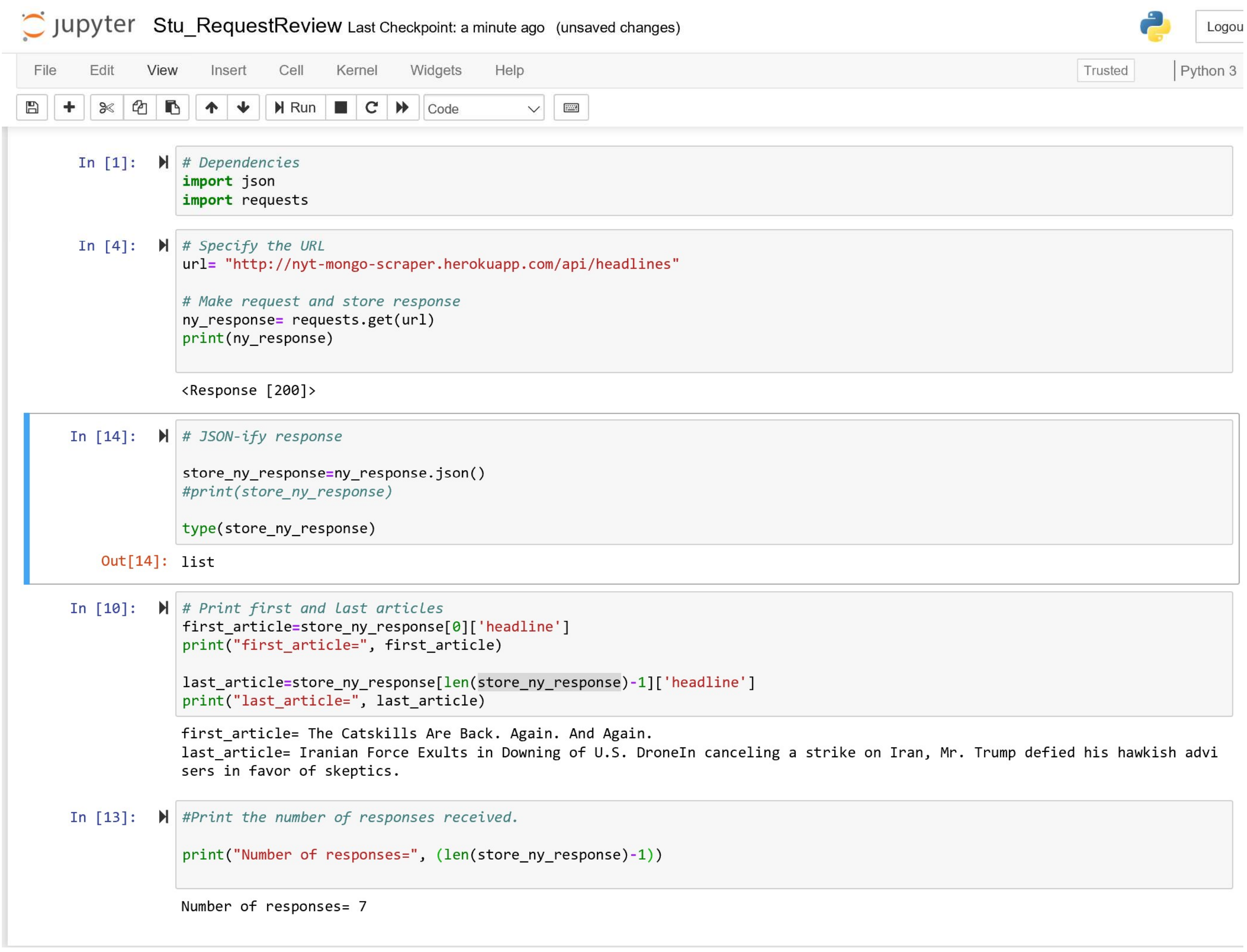
Exercise 02- Requests & Responses

This activity provides practice making API calls, converting the response to JSON, and then manipulating the result with Python.

Instructions

- Make a request to the following endpoint (<http://nyt-mongo-scraper.herokuapp.com/api/headlines>) and store the response.
- JSON-ify the response.
- Print the JSON representations of the first and last posts.
- Print number of posts received.

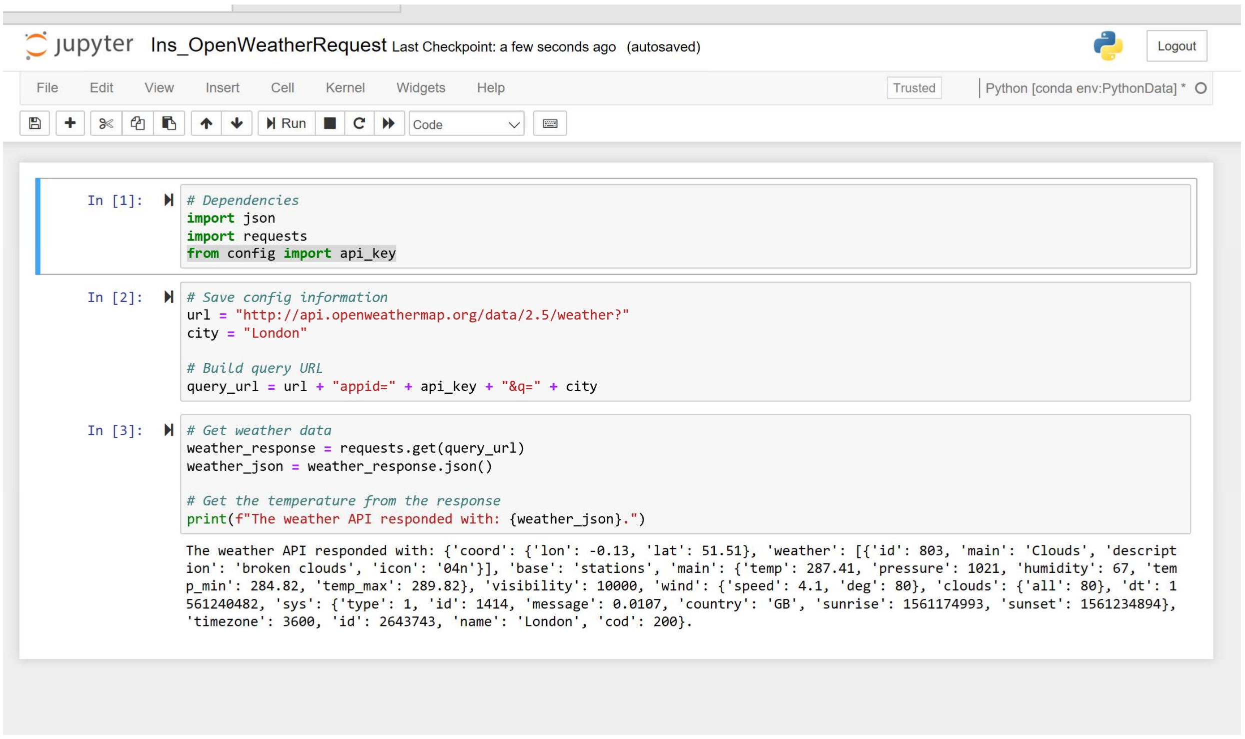
Divya’s Solution Code:


jupyter Stu_RequestReview Last Checkpoint: a minute ago (unsaved changes)
Logou
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [1]:  # Dependencies
         import json
         import requests

In [4]:  # Specify the URL
         url= "http://nyt-mongo-scraper.herokuapp.com/api/headlines"

         # Make request and store response
         ny_response= requests.get(url)
         print(ny_response)

         <Response [200]>

In [14]: # JSON-ify response

         store_ny_response=ny_response.json()
         #print(store_ny_response)

         type(store_ny_response)

Out[14]: list

In [10]: # Print first and last articles
         first_article=store_ny_response[0]['headline']
         print("first_article=", first_article)

         last_article=store_ny_response[len(store_ny_response)-1]['headline']
         print("last_article=", last_article)

         first_article= The Catskills Are Back. Again. And Again.
         last_article= Iranian Force Exults in Downing of U.S. DroneIn canceling a strike on Iran, Mr. Trump defied his hawkish advi
         sers in favor of skeptics.

In [13]: #Print the number of responses received.

         print("Number of responses=", (len(store_ny_response)-1))

         Number of responses= 7
```

Exercise 03- Use API key from a config file and access data

Use “from config import api_key” and keep a valid key in the config file. Replace the key if it expires. Used the api key mentioned in the slack channel and it worked!


jupyter Ins_OpenWeatherRequest Last Checkpoint: a few seconds ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python [conda env:PythonData] *
Run
Code

```
In [1]:  # Dependencies
         import json
         import requests
         from config import api_key

In [2]:  # Save config information
         url = "http://api.openweathermap.org/data/2.5/weather?"
         city = "London"

         # Build query URL
         query_url = url + "appid=" + api_key + "&q=" + city

In [3]:  # Get weather data
         weather_response = requests.get(query_url)
         weather_json = weather_response.json()

         # Get the temperature from the response
         print(f"The weather API responded with: {weather_json}.")

         The weather API responded with: {'coord': {'lon': -0.13, 'lat': 51.51}, 'weather': [{'id': 803, 'main': 'Clouds', 'descript
         ion': 'broken clouds', 'icon': '04n'}], 'base': 'stations', 'main': {'temp': 287.41, 'pressure': 1021, 'humidity': 67, 'tem
         p_min': 284.82, 'temp_max': 289.82}, 'visibility': 10000, 'wind': {'speed': 4.1, 'deg': 80}, 'clouds': {'all': 80}, 'dt': 1
         561240482, 'sys': {'type': 1, 'id': 1414, 'message': 0.0107, 'country': 'GB', 'sunrise': 1561174993, 'sunset': 1561234894},
         'timezone': 3600, 'id': 2643743, 'name': 'London', 'cod': 200}.
```

APIs have different versions.

Exercise 03 – Set up API keys and look at API documentation

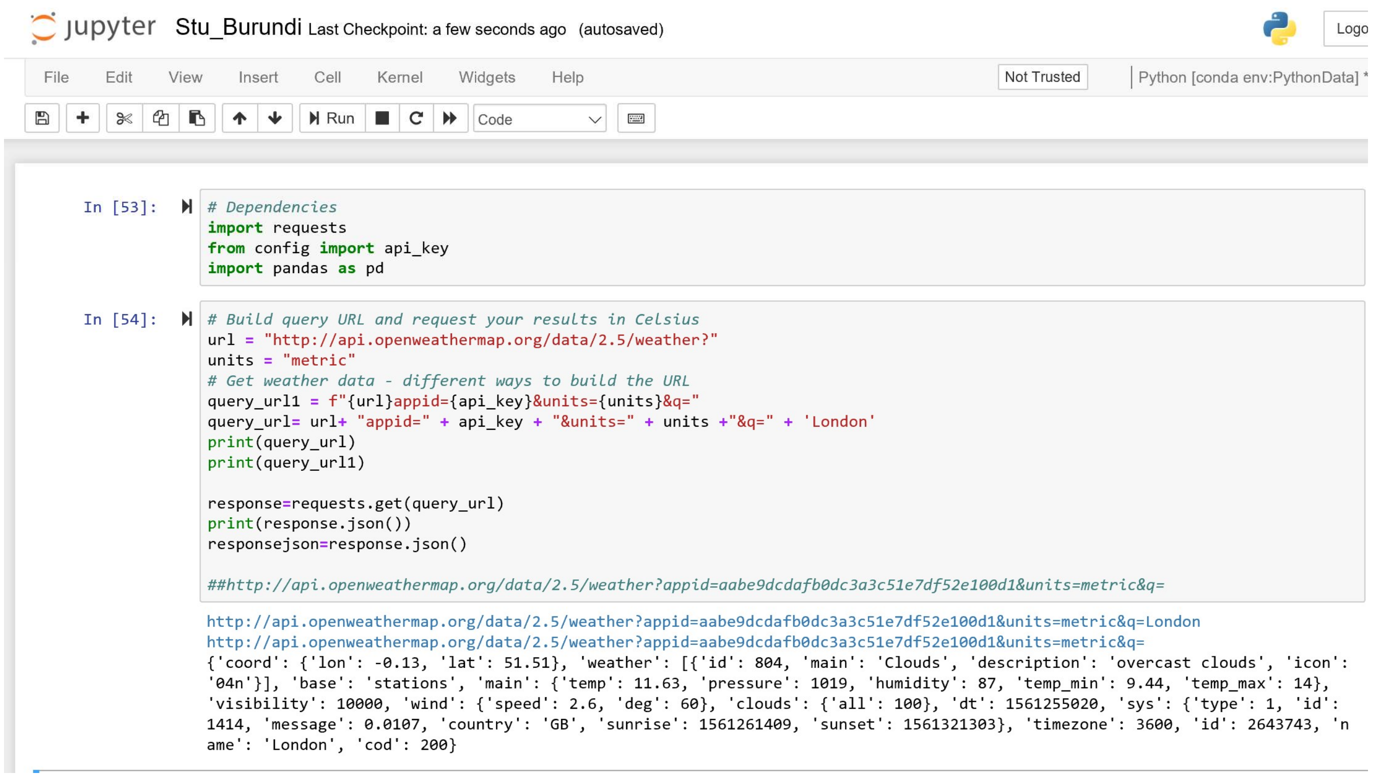
Sign up at Open Weather Map and get my API KEY - https://home.openweathermap.org/api_keys

Looking at API Documentation - https://openweathermap.org/api
Understanding what APIS are available to work on and pick the appropriate one.

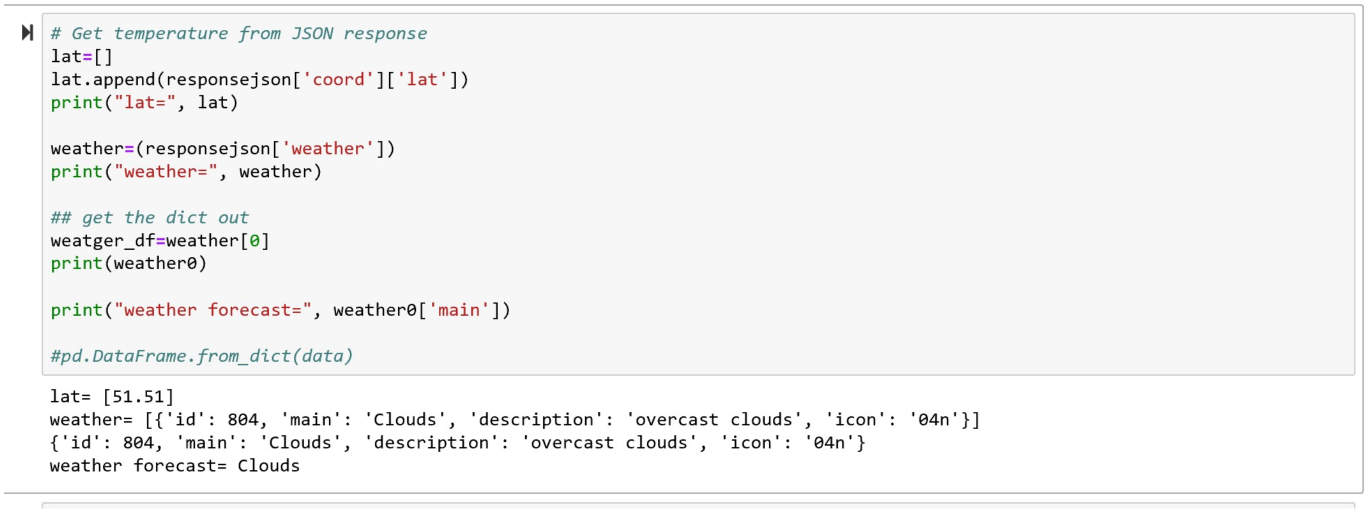
Exercise 04- Weather in Bujumbura

This activity gives students practice with making API calls and handling responses.
Instructions

- Save all of your "config" information—i.e., your API key; the base URL; etc.—before moving on.
- Build your query URL. Check the documentation to figure out how to request temperatures in Celsius.
- Make your request, and save the API response.
- Retrieve the current temperature in Bujumbura from the JSON response.
- Print the temperature to the console.


jupyter Stu_Burundi Last Checkpoint: a few seconds ago (autosaved)
Logo
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python [conda env:PythonData] *
Run
Code

```
In [53]:  # Dependencies
          import requests
          from config import api_key
          import pandas as pd

In [54]:  # Build query URL and request your results in Celsius
          url = "http://api.openweathermap.org/data/2.5/weather?"
          units = "metric"
          # Get weather data - different ways to build the URL
          query_url1 = f"{url}appid={api_key}&units={units}&q="
          query_url= url+ "appid=" + api_key + "&units=" + units +"&q=" + 'London'
          print(query_url)
          print(query_url1)

          response=requests.get(query_url)
          print(response.json())
          responsejson=response.json()

          ##http://api.openweathermap.org/data/2.5/weather?appid=aabe9dcdafb0dc3a3c51e7df52e100d1&units=metric&q=

          http://api.openweathermap.org/data/2.5/weather?appid=aabe9dcdafb0dc3a3c51e7df52e100d1&units=metric&q=London
          http://api.openweathermap.org/data/2.5/weather?appid=aabe9dcdafb0dc3a3c51e7df52e100d1&units=metric&q=
          {'coord': {'lon': -0.13, 'lat': 51.51}, 'weather': [{'id': 804, 'main': 'Clouds', 'description': 'overcast clouds', 'icon':
          '04n'}], 'base': 'stations', 'main': {'temp': 11.63, 'pressure': 1019, 'humidity': 87, 'temp_min': 9.44, 'temp_max': 14},
          'visibility': 10000, 'wind': {'speed': 2.6, 'deg': 60}, 'clouds': {'all': 100}, 'dt': 1561255020, 'sys': {'type': 1, 'id':
          1414, 'message': 0.0107, 'country': 'GB', 'sunrise': 1561261409, 'sunset': 1561321303}, 'timezone': 3600, 'id': 2643743, 'n
          ame': 'London', 'cod': 200}
```



```
# Get temperature from JSON response
lat=[]
lat.append(responsejson['coord']['lat'])
print("lat=", lat)

weather=(responsejson['weather'])
print("weather=", weather)

## get the dict out
weatger_df=weather[0]
print(weather0)

print("weather forecast=", weather0['main'])

#pd.DataFrame.from_dict(data)

lat= [51.51]
weather= [{'id': 804, 'main': 'Clouds', 'description': 'overcast clouds', 'icon': '04n'}]
{'id': 804, 'main': 'Clouds', 'description': 'overcast clouds', 'icon': '04n'}
weather forecast= Clouds
```

Exercise 05- Ins_OpenWeatherDataFrame

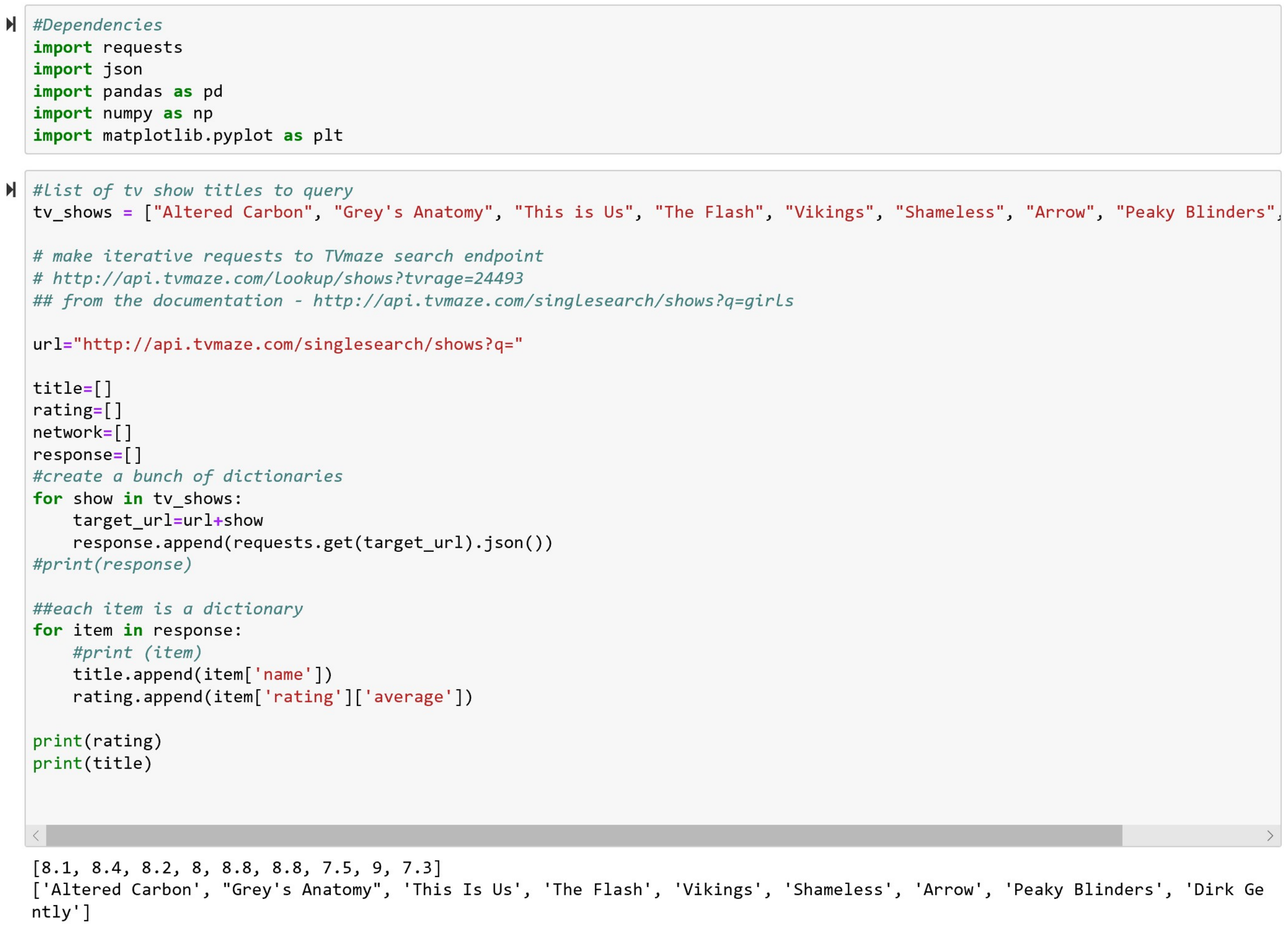
- Build a query with the API key
- Make a call and get the response and initiate a JSON object.
- Start calculations of the response.
- Plot the data with matplotlib.

Test the length of the lists to make sure you don’t exceed the response limit for the day set by the server.

Exercise 06- TV Ratings

- You may use the list of TV shows provided in the starter file or create your own.
- Request information on each TV show from the [TVmaze API's Show Search endpoint](https://www.tvmaze.com/api#show-search)
- Store the name and rating information into lists.
- Store this data in a dictionary and use it to create a Pandas DataFrame.
- Use matplotlib to create a bar chart comparing the ratings of each show.

Divya’s code:

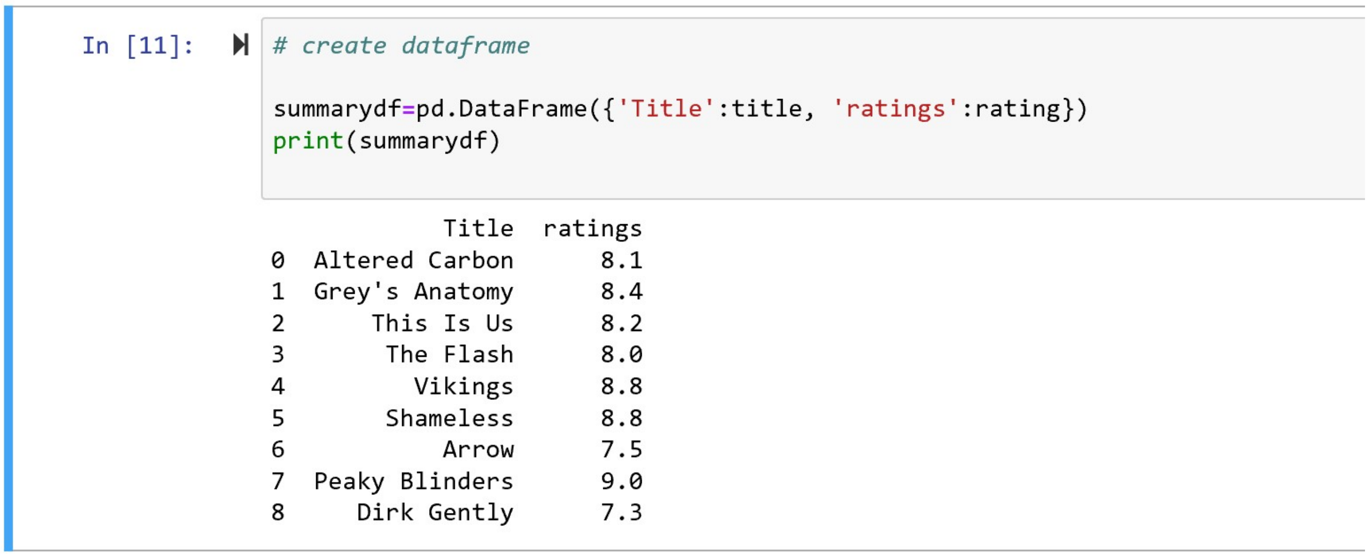


```python
#Dependencies
import requests
import json
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```python
#list of tv show titles to query
tv_shows = ["Altered Carbon", "Grey's Anatomy", "This is Us", "The Flash", "Vikings", "Shameless", "Arrow", "Peaky Blinders",

# make iterative requests to TVmaze search endpoint
# http://api.tvmaze.com/lookup/shows?tvrage=24493
## from the documentation - http://api.tvmaze.com/singlesearch/shows?q=girls

url="http://api.tvmaze.com/singlesearch/shows?q="

title=[]
rating=[]
network=[]
response=[]
#create a bunch of dictionaries
for show in tv_shows:
    target_url=url+show
    response.append(requests.get(target_url).json())
#print(response)

##each item is a dictionary
for item in response:
    #print (item)
    title.append(item['name'])
    rating.append(item['rating']['average'])

print(rating)
print(title)
```

```
[8.1, 8.4, 8.2, 8, 8.8, 8.8, 7.5, 9, 7.3]
['Altered Carbon', "Grey's Anatomy", 'This Is Us', 'The Flash', 'Vikings', 'Shameless', 'Arrow', 'Peaky Blinders', 'Dirk Ge
ntly']
```

```
ntly']
```

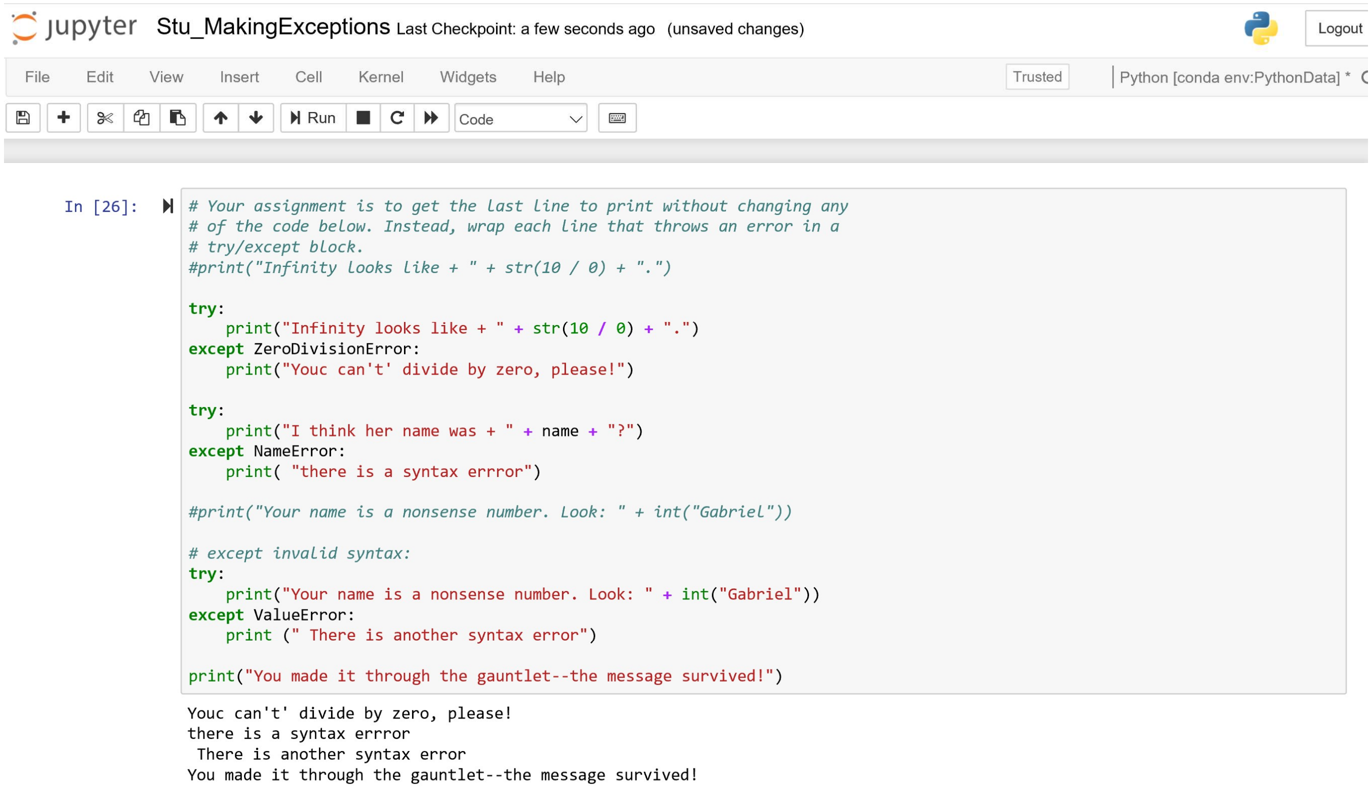


```
In [11]:  # create dataframe

          summarydf=pd.DataFrame({'Title':title, 'ratings':rating})
          print(summarydf)
```

```
             Title  ratings
0   Altered Carbon      8.1
1   Grey's Anatomy      8.4
2       This Is Us      8.2
3        The Flash      8.0
4          Vikings      8.8
5        Shameless      8.8
6            Arrow      7.5
7   Peaky Blinders      9.0
8      Dirk Gently      7.3
```

Exercise 07 & 08- Exception handling

- Try and catch blocks of code to handle exceptions.

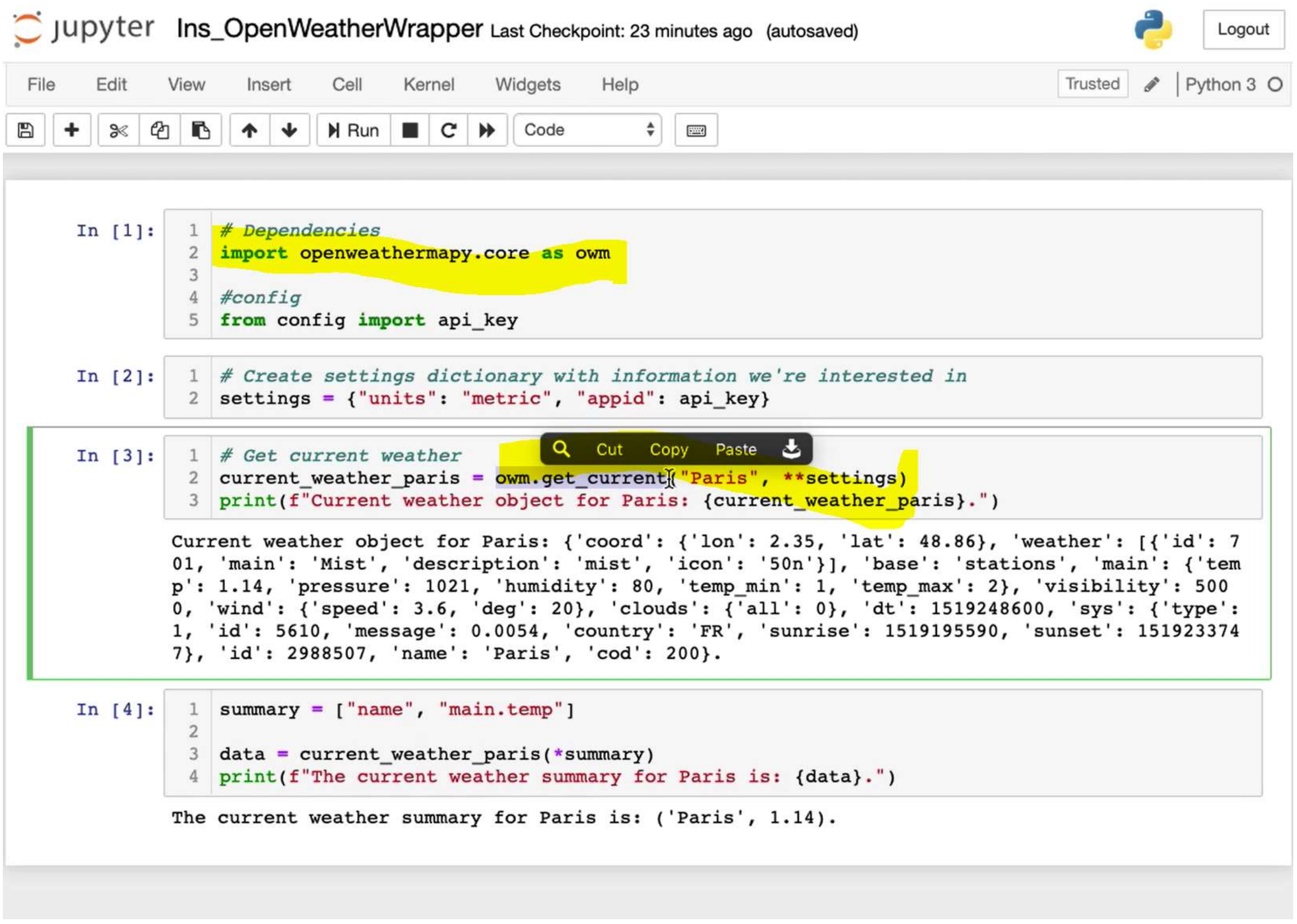

jupyter Stu_MakingExceptions Last Checkpoint: a few seconds ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python [conda env:PythonData] *
Run
Code

```
In [26]:  # Your assignment is to get the last line to print without changing any
          # of the code below. Instead, wrap each line that throws an error in a
          # try/except block.
          #print("Infinity looks like + " + str(10 / 0) + ".")

          try:
              print("Infinity looks like + " + str(10 / 0) + ".")
          except ZeroDivisionError:
              print("Youc can't' divide by zero, please!")

          try:
              print("I think her name was + " + name + "?")
          except NameError:
              print( "there is a syntax errror")

          #print("Your name is a nonsense number. Look: " + int("Gabriel"))

          # except invalid syntax:
          try:
              print("Your name is a nonsense number. Look: " + int("Gabriel"))
          except ValueError:
              print (" There is another syntax error")

          print("You made it through the gauntlet--the message survived!")
```

```
Youc can't' divide by zero, please!
there is a syntax errror
 There is another syntax error
You made it through the gauntlet--the message survived!
```

Exercise 09 – importing API specific modules and using them to build URLs and other things.

pip install openweathermapy


jupyter Ins_OpenWeatherWrapper Last Checkpoint: 23 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
Cut Copy Paste

In [1]:
```
# Dependencies
import openweathermapy.core as owm

#config
from config import api_key
```

In [2]:
```
# Create settings dictionary with information we're interested in
settings = {"units": "metric", "appid": api_key}
```

In [3]:
```
# Get current weather
current_weather_paris = owm.get_current("Paris", **settings)
print(f"Current weather object for Paris: {current_weather_paris}.")
```

```
Current weather object for Paris: {'coord': {'lon': 2.35, 'lat': 48.86}, 'weather': [{'id': 7
01, 'main': 'Mist', 'description': 'mist', 'icon': '50n'}], 'base': 'stations', 'main': {'tem
p': 1.14, 'pressure': 1021, 'humidity': 80, 'temp_min': 1, 'temp_max': 2}, 'visibility': 500
0, 'wind': {'speed': 3.6, 'deg': 20}, 'clouds': {'all': 0}, 'dt': 1519248600, 'sys': {'type':
1, 'id': 5610, 'message': 0.0054, 'country': 'FR', 'sunrise': 1519195590, 'sunset': 151923374
7}, 'id': 2988507, 'name': 'Paris', 'cod': 200}.
```

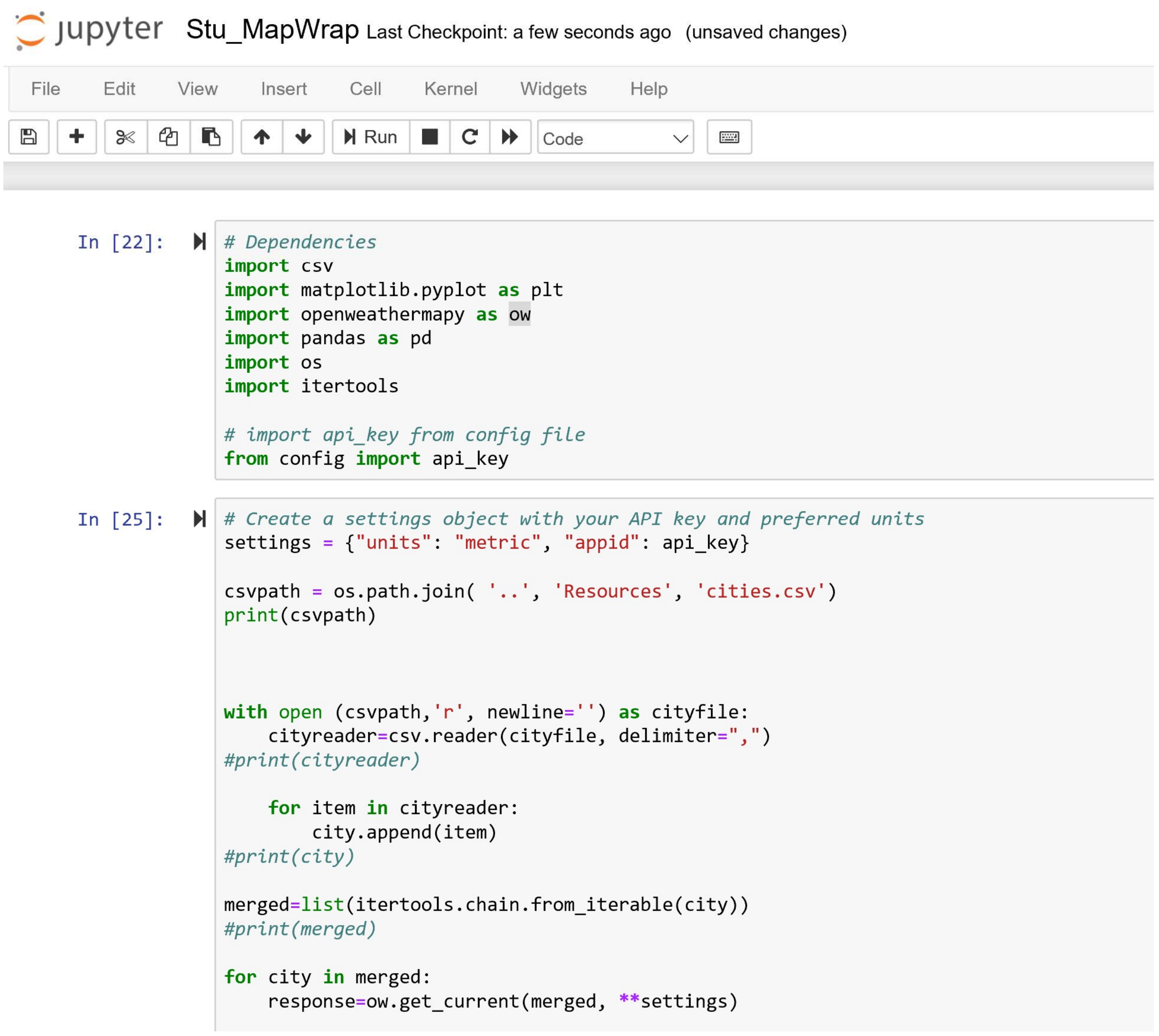
In [4]:
```
summary = ["name", "main.temp"]

data = current_weather_paris(*summary)
print(f"The current weather summary for Paris is: {data}.")
```

```
The current weather summary for Paris is: ('Paris', 1.14).
```

Exercise 010 – importing API specific modules and using them to build URLs and other things.

Read from the CSV file and iterate through the cities. Calculate the values for each city by reading the JSON response from the API calls.

Code to read from the CSV file and change list of a list to flat list. Got invalid API key while trying to access web data, will talk to the instructors on Tuesday.


jupyter
Stu_MapWrap Last Checkpoint: a few seconds ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

In [22]:
```
# Dependencies
import csv
import matplotlib.pyplot as plt
import openweathermapy as ow
import pandas as pd
import os
import itertools

# import api_key from config file
from config import api_key
```

In [25]:
```
# Create a settings object with your API key and preferred units
settings = {"units": "metric", "appid": api_key}

csvpath = os.path.join( '..', 'Resources', 'cities.csv')
print(csvpath)



with open (csvpath,'r', newline='') as cityfile:
    cityreader=csv.reader(cityfile, delimiter=",")
#print(cityreader)

    for item in cityreader:
        city.append(item)
#print(city)

merged=list(itertools.chain.from_iterable(city))
#print(merged)

for city in merged:
    response=ow.get_current(merged, **settings)
```

Exercise 011 - Ins_WorldBankAPI

The response object is an XML. Check if you can get JSON from the same API instead.

Exercise 013 –Citypy- Will be discussed next class.