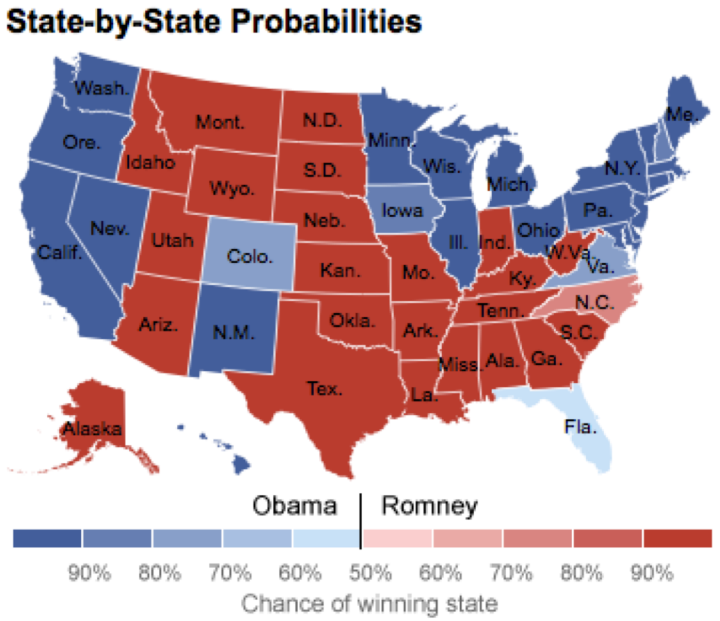
Preparation Homework:
https://www.youtube.com/watch?v=N4mEzFDjqtA

Until 32 minutes

What can be done with “data”?

Predicting US elections based on tweets analysis
http://www.aioptify.com/predictinguselection.php



More examples

https://github.com/d3/d3/wiki/Gallery
https://processing.org/exhibition/
https://greensock.com/examples-showcases
https://www.kaspersky.com/blog/cool-big-data-projects/8186/

Common ways to get the data

- Data dumps
- Scrapping
- API's

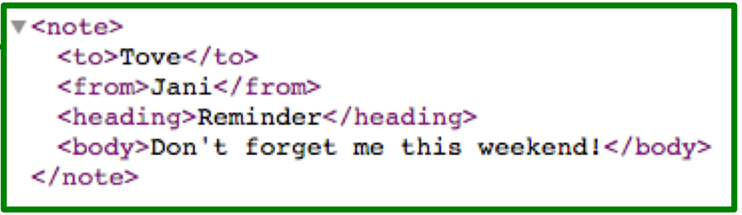
Data Dumps

Data stored in file(s) data

Can have various formats:

- CSV
- SQL,
- XML

```
ID,name,age
1,Mike,24
2,Elizabeth,56
```



```
▼<note>
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
 </note>
```

data EASY FOR COMPUTERS TO READ/PROCESS

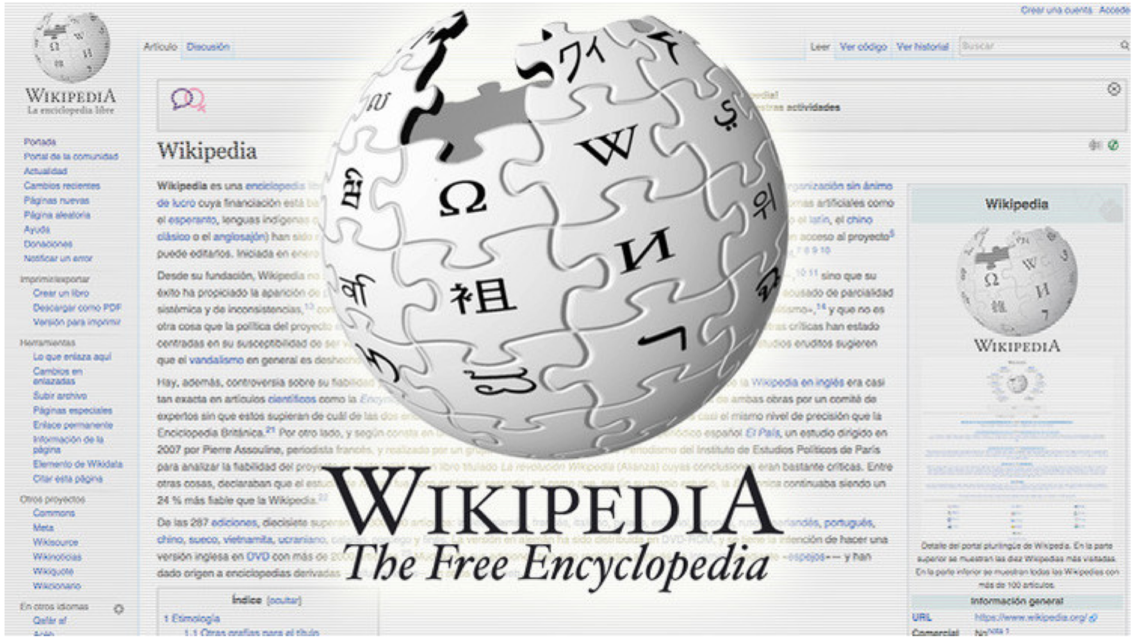
Might come with tools to pre-process and transform data

Examples: Data Dumps

English wikipedia data dumps

http://download.wikimedia.org/enwiki/

StackExchange academia Q&A forum
https://archive.org/download/stackexchange/academia.meta.stackexchange.com.7z



API

Application

Programming

Interface

an **I**nterface

used by **P**rograms to interact

with an **A**pplication

WEB Scraping
WEB Bot
WEB Spider
WEB Robot
WEB Harvesting
WEB Data Retrieval

Extracting
UNSTRUCTURED
EASY FOR HUMANS TO READ
data from websites

Collecting data from webpages in the way we see

API

Application

Programming

Interface

an **I**nterface

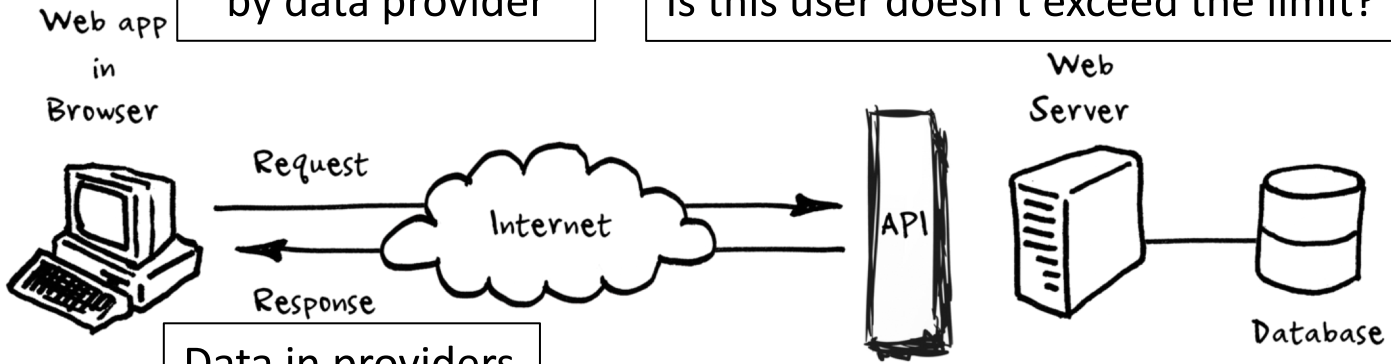
used by **P**rograms to interact

with an **A**pplication

Web API's for data gathering

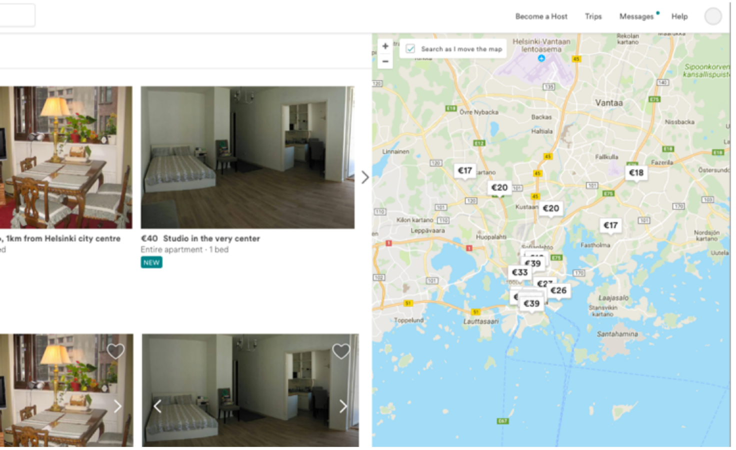
Approach designed by data provider

Is this user has rights to access the data?
Is this user doesn't exceed the limit?



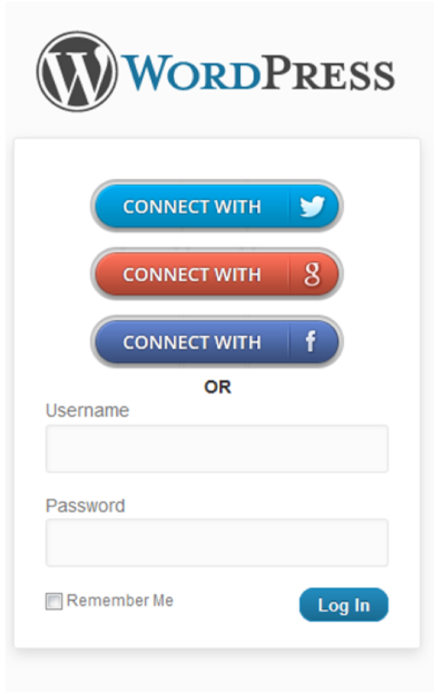
Data in providers decided format

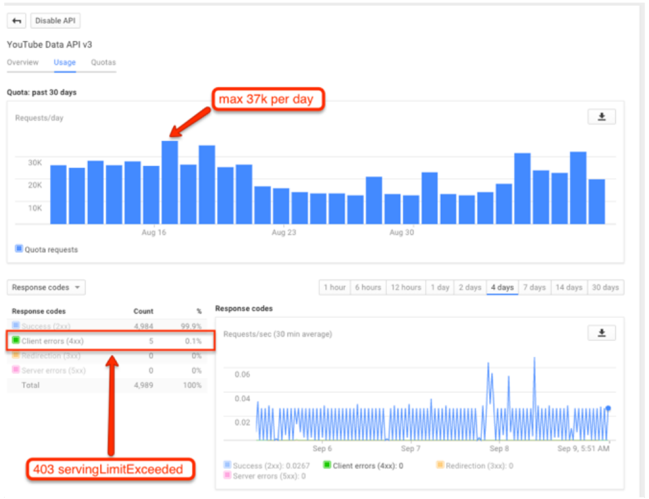
Give to the user the data he/she requests



, 1km from Helsinki city centre

€40 Studio in the very center
Entire apartment · 1 bed
NEW





YouTube Data API v3

Quota: past 30 days

Response codes	Count	%
Success (2xx)	4,984	99.9%
Client errors (4xx)	5	0.1%
Redirection (3xx)	0	0%
Server errors (5xx)	0	0%
Total	4,989	100%

Response codes

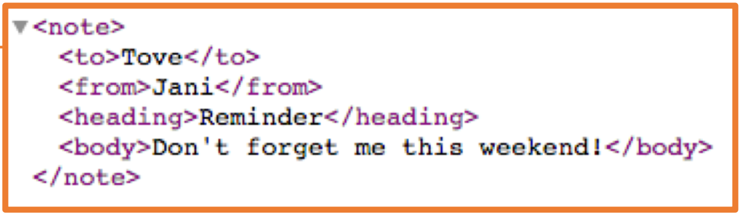
Success (2xx): 0.0267 Client errors (4xx): 0 Redirection (3xx): 0
Server errors (5xx): 0

Web API

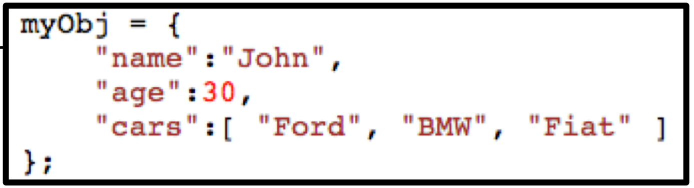
WEB Uri (Query):
https://www.googleapis.com/youtube/v3/search?part=snippet&q=processing&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE

Response

XML

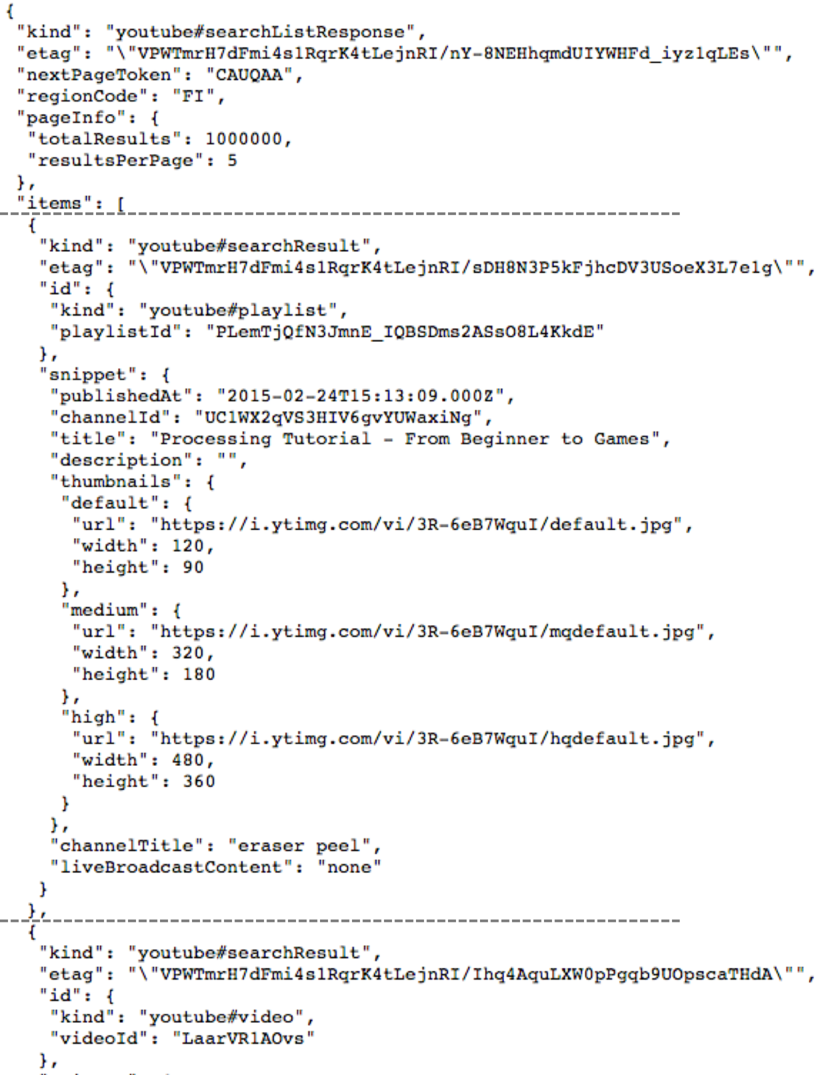


```
▼<note>
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
 </note>
```



```
myObj = {
    "name":"John",
    "age":30,
    "cars":[ "Ford", "BMW", "Fiat" ]
};
```

JSON



```
{
 "kind": "youtube#searchListResponse",
 "etag": "\"VPWTmrH7dFmi4s1RqrK4tLejnRI/nY-8NEHhqmdUIYWHFd_iyz1qLEs\"",
 "nextPageToken": "CAUQAA",
 "regionCode": "FI",
 "pageInfo": {
  "totalResults": 1000000,
  "resultsPerPage": 5
 },
 "items": [
  {
   "kind": "youtube#searchResult",
   "etag": "\"VPWTmrH7dFmi4s1RqrK4tLejnRI/sDH8N3P5kFjhcDV3USoeX3L7e1g\"",
   "id": {
    "kind": "youtube#playlist",
    "playlistId": "PLemTjQfN3JmnE_IQBSDms2ASsO8L4KkdE"
   },
   "snippet": {
    "publishedAt": "2015-02-24T15:13:09.000Z",
    "channelId": "UC1WX2qVS3HIV6gvYUWaxiNg",
    "title": "Processing Tutorial - From Beginner to Games",
    "description": "",
    "thumbnails": {
     "default": {
      "url": "https://i.ytimg.com/vi/3R-6eB7WquI/default.jpg",
      "width": 120,
      "height": 90
     },
     "medium": {
      "url": "https://i.ytimg.com/vi/3R-6eB7WquI/mqdefault.jpg",
      "width": 320,
      "height": 180
     },
     "high": {
      "url": "https://i.ytimg.com/vi/3R-6eB7WquI/hqdefault.jpg",
      "width": 480,
      "height": 360
     }
    },
    "channelTitle": "eraser peel",
    "liveBroadcastContent": "none"
   }
  },
  {
   "kind": "youtube#searchResult",
   "etag": "\"VPWTmrH7dFmi4s1RqrK4tLejnRI/Ihq4AquLXW0pPgqb9UOpscaTHdA\"",
   "id": {
    "kind": "youtube#video",
    "videoId": "LaarVR1AOvs"
   },
```

- YouTube data through API (JSON)

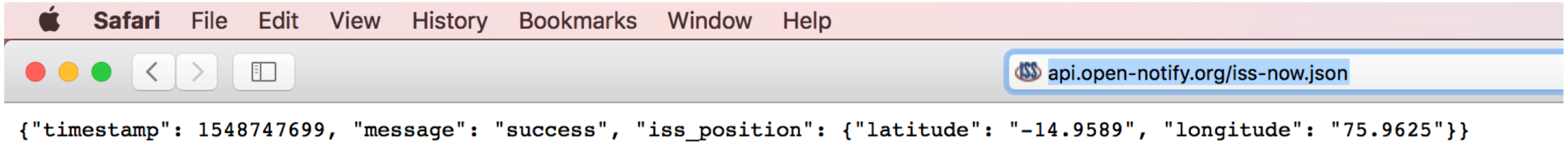
Response status

https://www.dataquest.io/blog/python-api-tutorial/

200 - everything went okay, and the result has been returned
301 - the server is redirecting you to a different endpoint. *This can happen when a company switches domain names, or an endpoint name is changed.*
401 - the server thinks you're not authenticated. *This happens when you don't send the right credentials to access an API*
400 - the server thinks you made a bad request. *This can happen when you don't send along the right data, among other things.*
403 - the resource you're trying to access is forbidden. *When you don't have the right permissions to see it.*
404 - the resource you tried to access wasn't found on the

Looks like an ordinary web link:

http://api.open-notify.org/iss-now.json

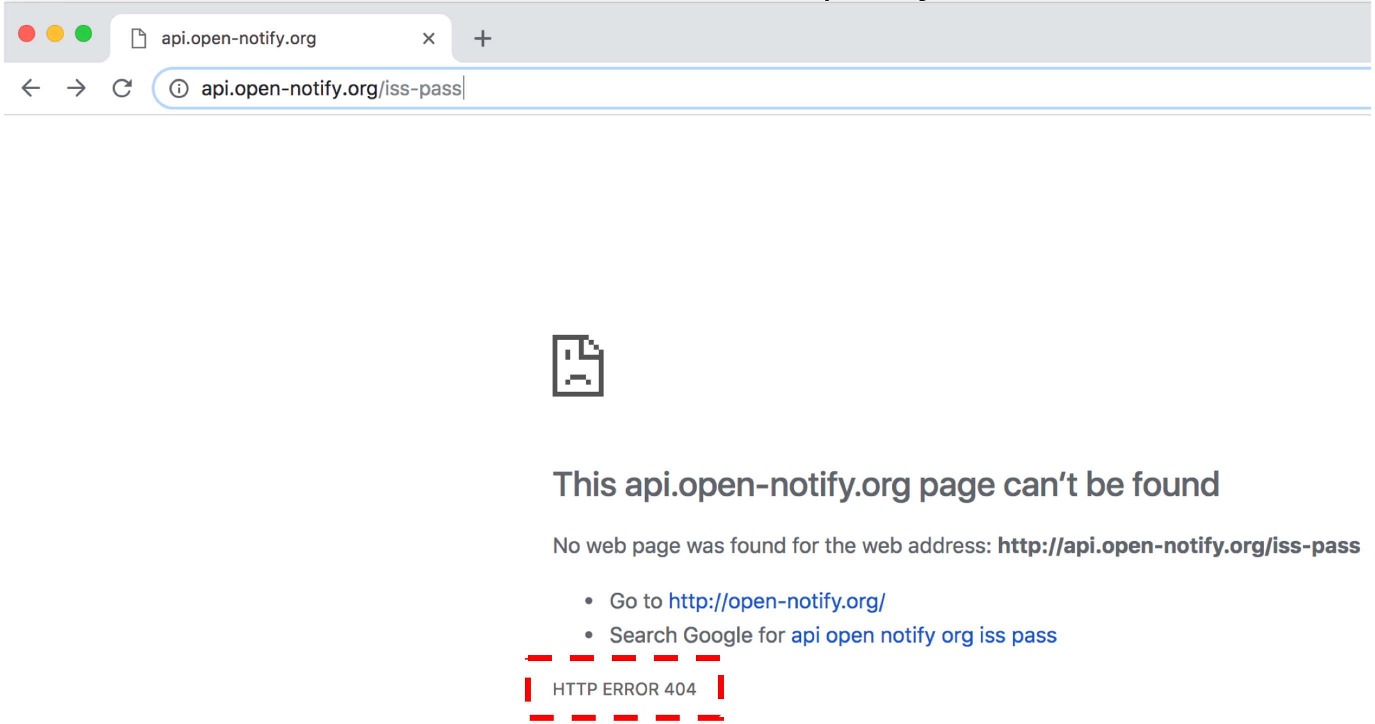


```
{"timestamp": 1548747699, "message": "success", "iss_position": {"latitude": "-14.9589", "longitude": "75.9625"}}
```

Returns position of international space station

There is no such resources as *…/iss-pass* thus it returns **404 error**

…/iss-pass.json would return some page


api.open-notify.org

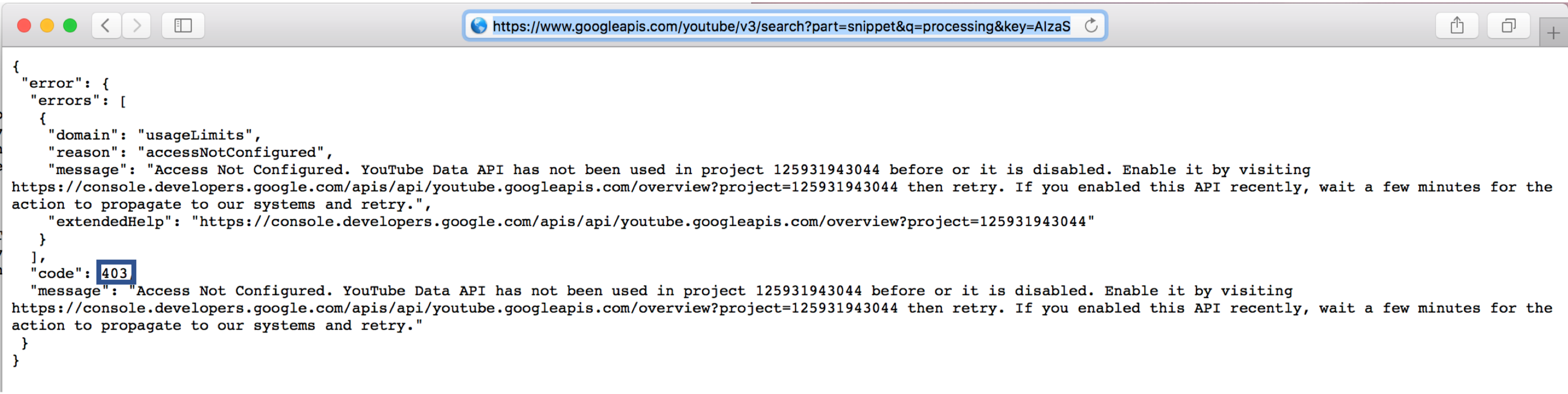
api.open-notify.org/iss-pass

This api.open-notify.org page can’t be found

No web page was found for the web address: **http://api.open-notify.org/iss-pass**

- Go to http://open-notify.org/
- Search Google for api open notify org iss pass

HTTP ERROR 404

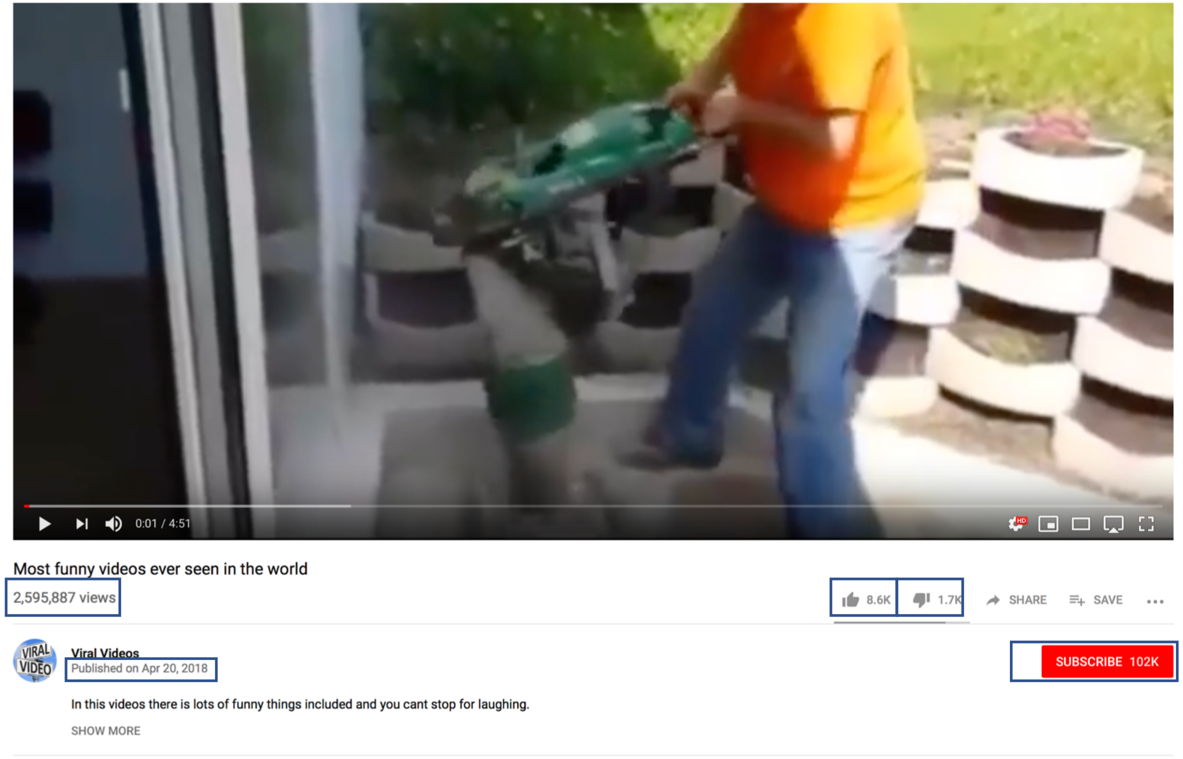


https://www.googleapis.com/youtube/v3/search?part=snippet&q=processing&key=AIzaS

```
{
 "error": {
  "errors": [
   {
    "domain": "usageLimits",
    "reason": "accessNotConfigured",
    "message": "Access Not Configured. YouTube Data API has not been used in project 125931943044 before or it is disabled. Enable it by visiting
https://console.developers.google.com/apis/api/youtube.googleapis.com/overview?project=125931943044 then retry. If you enabled this API recently, wait a few minutes for the
action to propagate to our systems and retry.",
    "extendedHelp": "https://console.developers.google.com/apis/api/youtube.googleapis.com/overview?project=125931943044"
   }
  ],
  "code": 403,
  "message": "Access Not Configured. YouTube Data API has not been used in project 125931943044 before or it is disabled. Enable it by visiting
https://console.developers.google.com/apis/api/youtube.googleapis.com/overview?project=125931943044 then retry. If you enabled this API recently, wait a few minutes for the
action to propagate to our systems and retry."
 }
}
```

https://www.youtube.com/watch?v=8ituvCpal3g

How it works?



Most funny videos ever seen in the world

2,595,887 views

Viral Videos

Published on Apr 20, 2018

In this videos there is lots of funny things included and you cant stop for laughing.

We can see the data in visual format

Main steps

1. Forming the query for YouTube server
2. Executing the Query
3. Parsing the the received document to JSON object
4. Extracting needed information from JSON object

1. Forming the query (1)

Query is given as a **String**. In simple terms, everything between quotes(" ") is a **String**.

- The query for YouTube API starts with:

  "https://www.googleapis.com/youtube/v3/"

- Then you need to append to the initial query with Strings for a particular resource you seek:
  - If you want to receive a list of YouTube <u>channels</u> add:
    "channels"
  - If you want to receive a list of YouTube <u>playlists</u> add:
    "playlists"
  - If you want to receive a list of YouTube <u>videos</u> add:
    "search"
- The example for video search should like this now:
  "*https://www.googleapis.com/youtube/v3/search"*

More on what type of resources you can search for https://developers.google.com/youtube/v3/getting-started#resources

1. Forming the query (2)

In order to search for a particular content - what a video/playlist/channel should be about, or when published, we need to add variables to the query.

Variables from the initial query separated with question mark (?):

“*https://www.googleapis.com/youtube/v3/search?part=snippet,statistics”*

We need to specify what type of data we are seeking:

- For this variable “part” is used. For videos search it can be equal to:

  “snippet”; “contentDetails”; “fileDetails”; “player”; “processingDetails”; “recordingDetails”; “statistics”; “status”; “suggestions”; “topicDetails”

- For getting snippets and statistics of queried videos, we should add to the query this part:

  “part=snippet,statistics”

- And exemplary query would look like this now:

“https://www.googleapis.com/youtube/v3/search?part=snippet,statistics”

More on ”part” https://developers.google.com/youtube/v3/getting-started#partial

1. Forming the query (3)

In order to search for a particular content - what a video/playlist/channel should be about, or when published, we need to add more variables to the query.

To append more Variables to the initial query we need to use a and sign(&):

“https://www.googleapis.com/youtube/v3/search?part=snippet,statistics&”

We need to add a variable and its value to specify the information we want to receive - to restrict to content, time of upload, etc…. Each of resources have different variables that can be used (also many are overlapping). Each variable must be equal to smth:

- For instance if searching for videos we can specify:

“q“; “channelId”; “channelType”; “fileDetails”; “eventType”; “location”; “maxResults”; “publishedAfter”; “pageToken”; “suggestions”; “topicDetails”; etc…

- … if for channels:

“categoryId”; “forUsername”; “maxResults”; “pageToken”; etc …

- Example to search for videos that its about processing and published after 2017 August would look like this:
  - *“https://www.googleapis.com/youtube/v3/search?part=snippet&q=processing&publishedAfter=2017-08-31T00:00:00Z”*

1. Forming the query (4)

Crucial step is defining API key variable to the query that equals to the personal API key.

Here are instructions to receive YouTube Data API V3 key:

- In Text:
  https://developers.google.com/youtube/android/player/register#Create_API_Keys
- In video:
  https://www.youtube.com/watch?v=GEqKjc6auSA
- After key is received it must be assigned to the query as other variables with "key=“
- Full query example should look like this:
  *"https://www.googleapis.com/youtube/v3/search?part=snippet&q=processing&publishedAfter=2017-08-31T00:00:00Z&key=*AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE*"*

YouTube API key

AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE

How to get YouTube API key

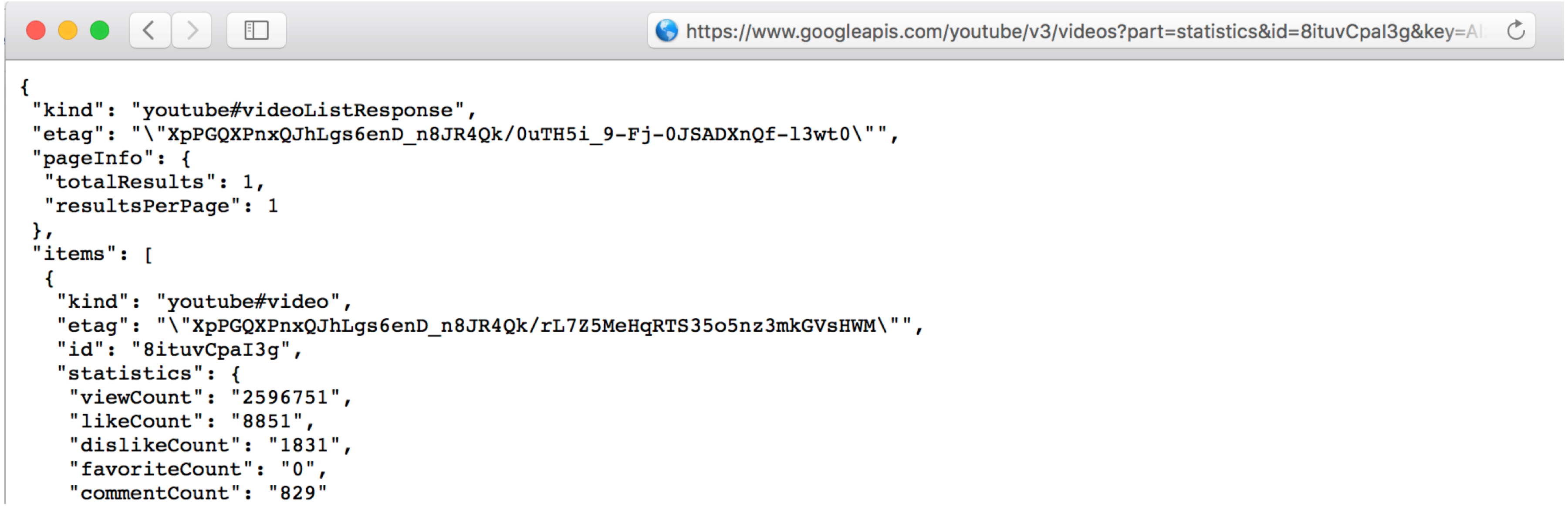
https://developers.google.com/youtube/v3/getting-started

2. Executing the Query

Video ID	Views	Likes	Dislikes	Subscribers	Comments
8ituvCpal3g	2595887	8600	1700	102000	823
…	…	…	…	…	…

Access Programatically

https://www.googleapis.com/youtube/v3/videos?part=statistics&id=8ituvCpal3g&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE



```
{
 "kind": "youtube#videoListResponse",
 "etag": "\"XpPGQXPnxQJhLgs6enD_n8JR4Qk/0uTH5i_9-Fj-0JSADXnQf-l3wt0\"",
 "pageInfo": {
  "totalResults": 1,
  "resultsPerPage": 1
 },
 "items": [
  {
   "kind": "youtube#video",
   "etag": "\"XpPGQXPnxQJhLgs6enD_n8JR4Qk/rL7Z5MeHqRTS35o5nz3mkGVsHWM\"",
   "id": "8ituvCpaI3g",
   "statistics": {
    "viewCount": "2596751",
    "likeCount": "8851",
    "dislikeCount": "1831",
    "favoriteCount": "0",
    "commentCount": "829"
```

YouTube Example to search videos that has ”processing” key word

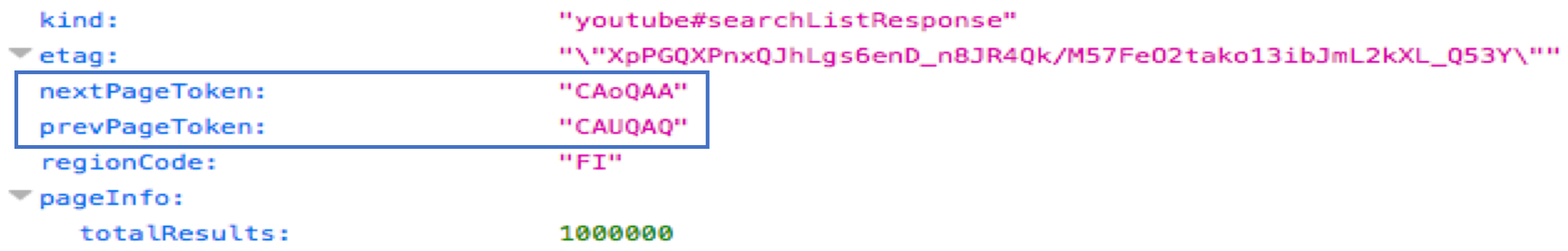
https://www.googleapis.com/youtube/v3/search?part=snippet&q=processing&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE

Video ID	Title	Date uploaded	Likes
8ituvCpal3g	Most funny videos ever seen in the world	…	…
8uwernsg	Processing data in style	…	…
9023sdfnsg	Learn js processing	…	…

https://www.googleapis.com/youtube/v3/search?pageToken=CAUQAA&part=snippet&q=processing&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE

pageToken

maxResults = [0, 50]



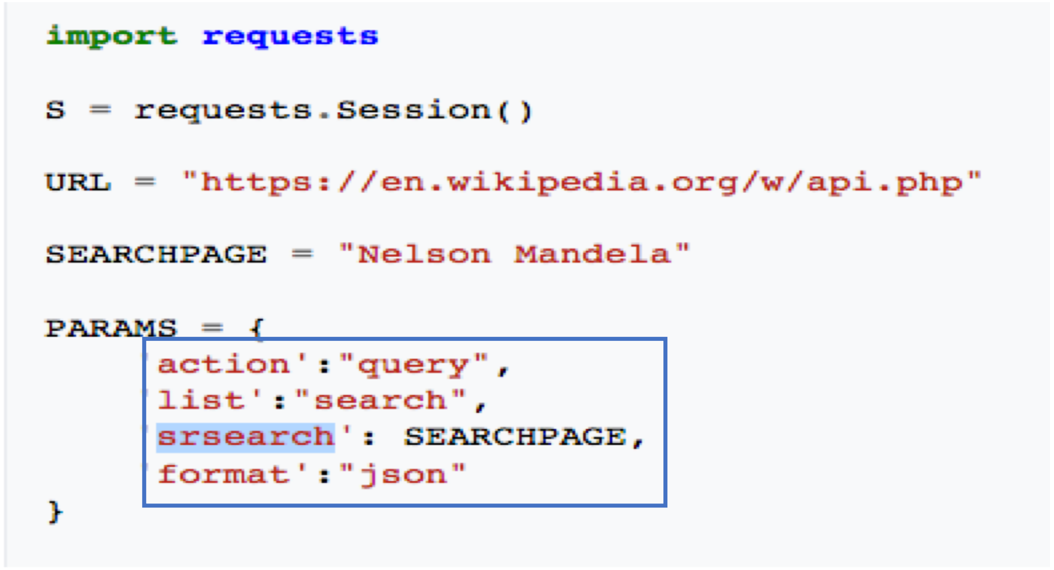
```
  kind:                  "youtube#searchListResponse"
▼ etag:                  "\"XpPGQXPnxQJhLgs6enD_n8JR4Qk/M57FeO2tako13ibJmL2kXL_Q53Y\""
  nextPageToken:         "CAoQAA"
  prevPageToken:         "CAUQAQ"
  regionCode:            "FI"
▼ pageInfo:
    totalResults:        1000000
```

Wikipedia API

https://en.wikipedia.org/w/api.php?action=query&list=search&format=json&srsearch=process

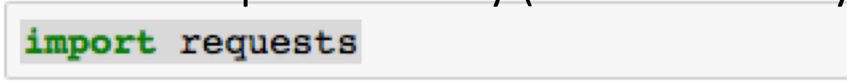
https://en.wikipedia.org/w/api.php	Base of the URL
?	Identification that following will be variables
&	Every next variable
Xxx=yyy	Variable assigned to value

https://en.wikipedia.org/w/api.php?action=query&list=search&format=json&srsearch=process



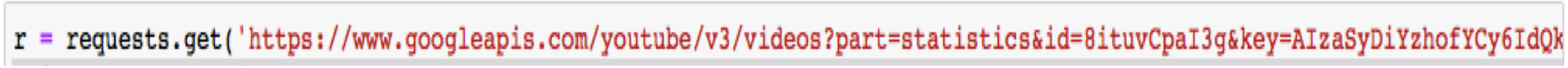
```python
import requests

S = requests.Session()

URL = "https://en.wikipedia.org/w/api.php"

SEARCHPAGE = "Nelson Mandela"

PARAMS = {
    'action':"query",
    'list':"search",
    'srsearch': SEARCHPAGE,
    'format':"json"
}
```

Define “requests” library (if its not found you need to install it first)



```
import requests
```

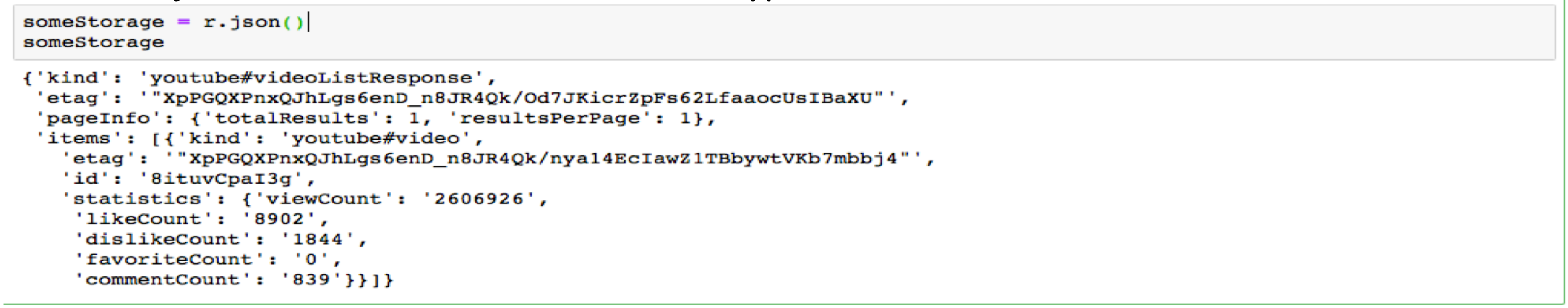
Now after we imported lib., we can use “requests” function



```
r = requests.get('https://www.googleapis.com/youtube/v3/videos?part=statistics&id=8ituvCpaI3g&key=AIzaSyDiYzhofYCy6IdQk
```

Basic Python Request

We need to define what type of output we will receive (usually from APIs comes JSON or XML). In this case .json. And we can view it if we will type variables name

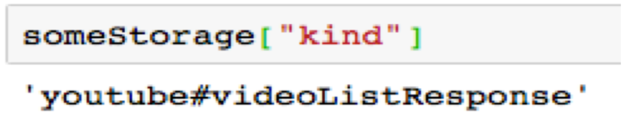


```
someStorage = r.json()
someStorage
```

```
{'kind': 'youtube#videoListResponse',
 'etag': '"XpPGQXPnxQJhLgs6enD_n8JR4Qk/Od7JKicrZpFs62LfaaocUsIBaXU"',
 'pageInfo': {'totalResults': 1, 'resultsPerPage': 1},
 'items': [{'kind': 'youtube#video',
   'etag': '"XpPGQXPnxQJhLgs6enD_n8JR4Qk/nya14EcIawZ1TBbywtVKb7mbbj4"',
   'id': '8ituvCpaI3g',
   'statistics': {'viewCount': '2606926',
    'likeCount': '8902',
    'dislikeCount': '1844',
    'favoriteCount': '0',
    'commentCount': '839'}}]}
```

JSON is a so called dictionary. It’s has very flexible structure for data storing. It has keys, and each key has value(s). For example, here were keys with names ‘kind’, ‘etag’, ‘pageInfo’, ‘totalResults’, etc., and values ‘youtube#videoListResponse’, ‘I’, etc.
Dictionary is defined by curly brackets ‘{‘ ‘}’

To access values of dictionary we must first call variable name and inside square brackets state the key (in this case ‘kind’) with quotes

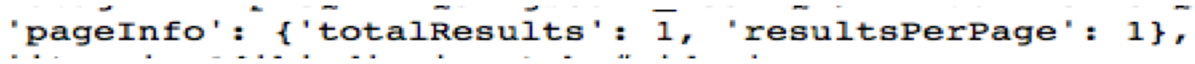


```
someStorage["kind"]
```

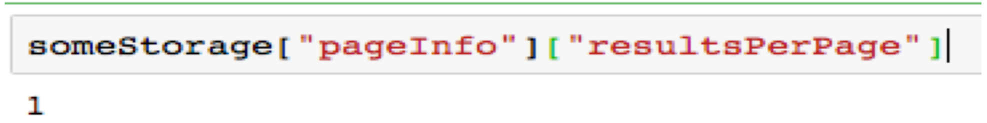
```
'youtube#videoListResponse'
```

Extracting JSON

If a key stores another dictionary value, we can access it by stating the parent key name (that stores dictionary), and the child (the next dictionary), in this case ‘pageInfo’ and ‘resultsPerPage’



```
'pageInfo': {'totalResults': 1, 'resultsPerPage': 1},
```



```
someStorage["pageInfo"]["resultsPerPage"]
```

```
1
```

If we find array (the object identified by square brackets '[' ']') in the dictionary, that meant that there are one or more dictionary that this key (in our case 'items') holds

```
'items': [{'kind': 'youtube#video',
  'etag': '"XpPGQXPnxQJhLgs6enD_n8JR4Qk/KzztUnwTJ-9nzNXBb6bfJe5ny2w"',
  'id': '8ituvCpaI3g',
  'statistics': {'viewCount': '2613729',
   'likeCount': '8928',
   'dislikeCount': '1846',
   'favoriteCount': '0',
   'commentCount': '840'}}]}
```

Therefore we must first state which element number we want to access. The numbers start from 0, not from 1.

```
someStorage["items"][0]["id"]
```

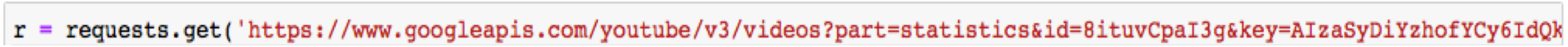
```
'8ituvCpaI3g'
```

To access 'commentCount' value the sequences will be like this:

```
someStorage["items"][0]["statistics"]["commentCount"]
```

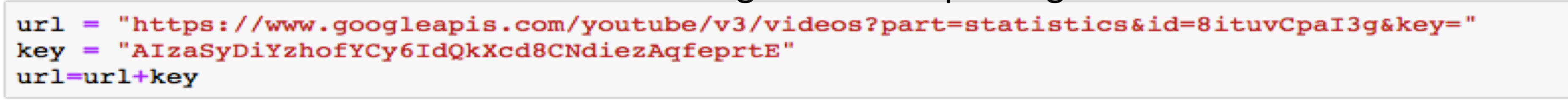
```
'840'
```

Extracting JSON



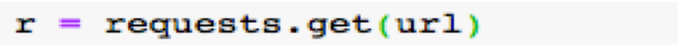
```
r = requests.get('https://www.googleapis.com/youtube/v3/videos?part=statistics&id=8ituvCpaI3g&key=AIzaSyDiYzhofYCy6IdQk
```

What is between the quotes is treated as simple text. We could divide the text by assigning parts to different variables and after combining them with plus sign



```
url = "https://www.googleapis.com/youtube/v3/videos?part=statistics&id=8ituvCpaI3g&key="
key = "AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE"
url=url+key
```

Then we could manipulate the key and feed the variable name instead of full text

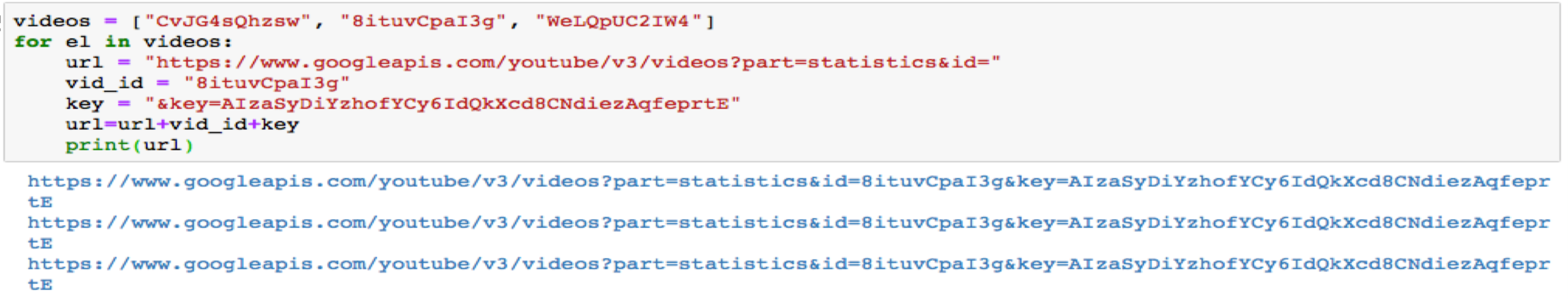


```
r = requests.get(url)
```

Manipulating Input

If we would like to make automatic queries for multiple times, we could adjust our url by dynamically adding search term.

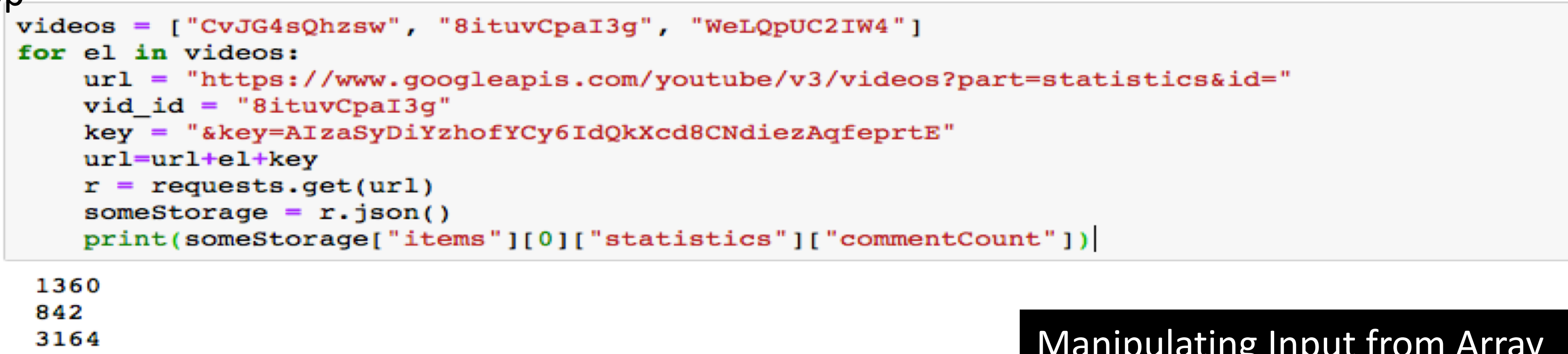
First we would need to create an array (in our case videos), then go through each of its ele



```
videos = ["CvJG4sQhzsw", "8ituvCpaI3g", "WeLQpUC2IW4"]
for el in videos:
    url = "https://www.googleapis.com/youtube/v3/videos?part=statistics&id="
    vid_id = "8ituvCpaI3g"
    key = "&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE"
    url=url+vid_id+key
    print(url)
```

```
https://www.googleapis.com/youtube/v3/videos?part=statistics&id=8ituvCpaI3g&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfepr
tE
https://www.googleapis.com/youtube/v3/videos?part=statistics&id=8ituvCpaI3g&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfepr
tE
https://www.googleapis.com/youtube/v3/videos?part=statistics&id=8ituvCpaI3g&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfepr
tE
```

Then we could run the same operation (call API and get comments) if we place it within ‘for’ loop

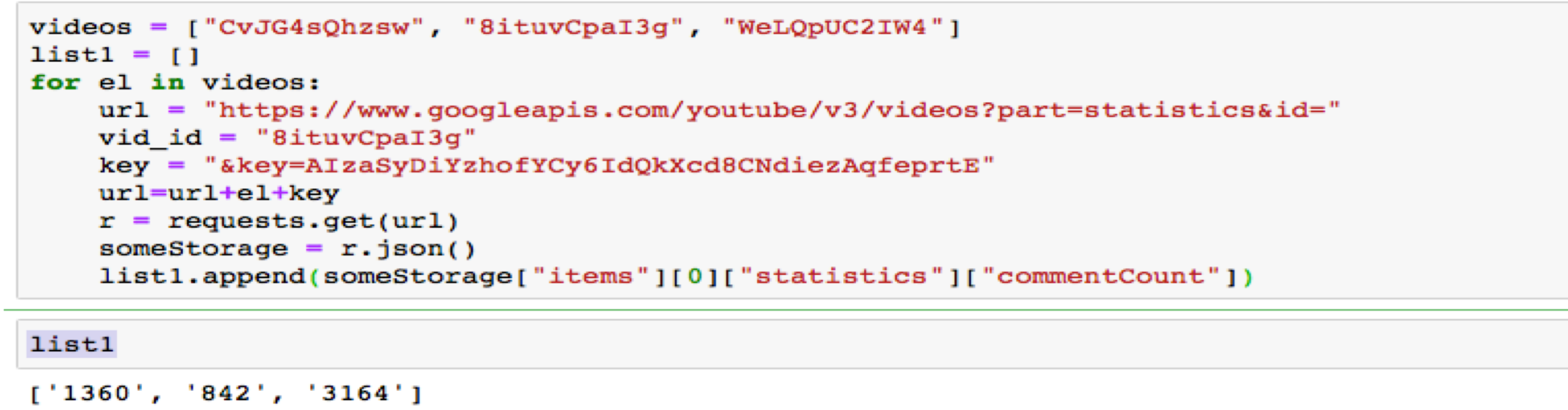


```
videos = ["CvJG4sQhzsw", "8ituvCpaI3g", "WeLQpUC2IW4"]
for el in videos:
    url = "https://www.googleapis.com/youtube/v3/videos?part=statistics&id="
    vid_id = "8ituvCpaI3g"
    key = "&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE"
    url=url+el+key
    r = requests.get(url)
    someStorage = r.json()
    print(someStorage["items"][0]["statistics"]["commentCount"])
```

```
1360
842
3164
```

Manipulating Input from Array

We could also store it in a list/array by appending to it every time the API is called

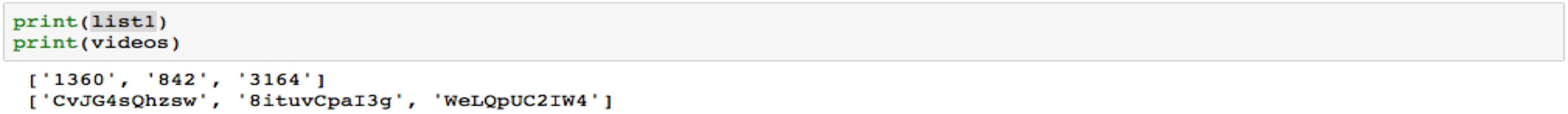


```
videos = ["CvJG4sQhzsw", "8ituvCpaI3g", "WeLQpUC2IW4"]
list1 = []
for el in videos:
    url = "https://www.googleapis.com/youtube/v3/videos?part=statistics&id="
    vid_id = "8ituvCpaI3g"
    key = "&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE"
    url=url+el+key
    r = requests.get(url)
    someStorage = r.json()
    list1.append(someStorage["items"][0]["statistics"]["commentCount"])
```

```
list1
```

```
['1360', '842', '3164']
```

Now we have two list which we can combine into a table

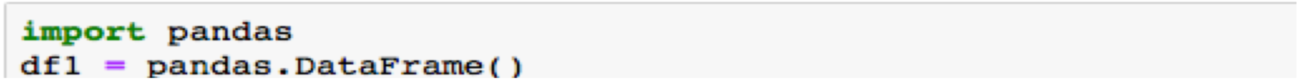


```
print(list1)
print(videos)
```

```
['1360', '842', '3164']
['CvJG4sQhzsw', '8ituvCpaI3g', 'WeLQpUC2IW4']
```

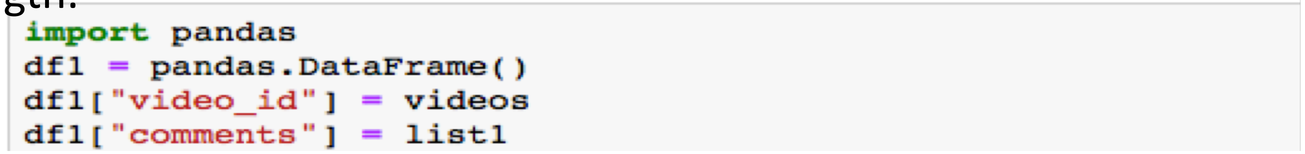
Manipulating Input from Array

For this we would store both lists into the DataFrame:
First we include library Pandas – DataFrames library, and create empty DataFrame



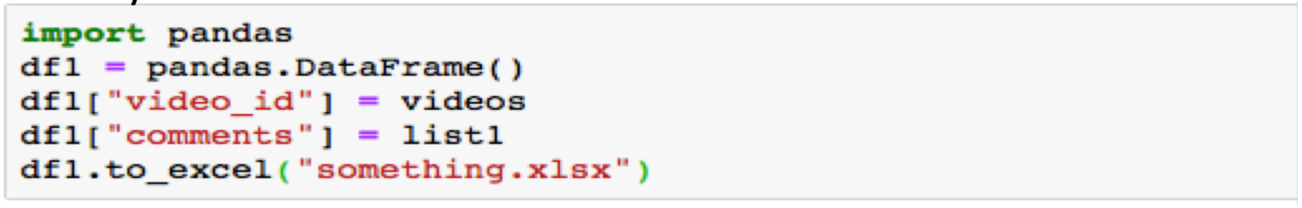
```
import pandas
df1 = pandas.DataFrame()
```

Later on we could create empty columns and assign to them our lists. Lists must be same length.



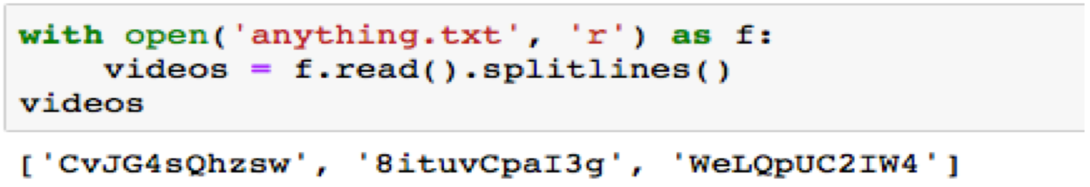
```
import pandas
df1 = pandas.DataFrame()
df1["video_id"] = videos
df1["comments"] = list1
```

And finally we can save our results to the excel file



```
import pandas
df1 = pandas.DataFrame()
df1["video_id"] = videos
df1["comments"] = list1
df1.to_excel("something.xlsx")
```

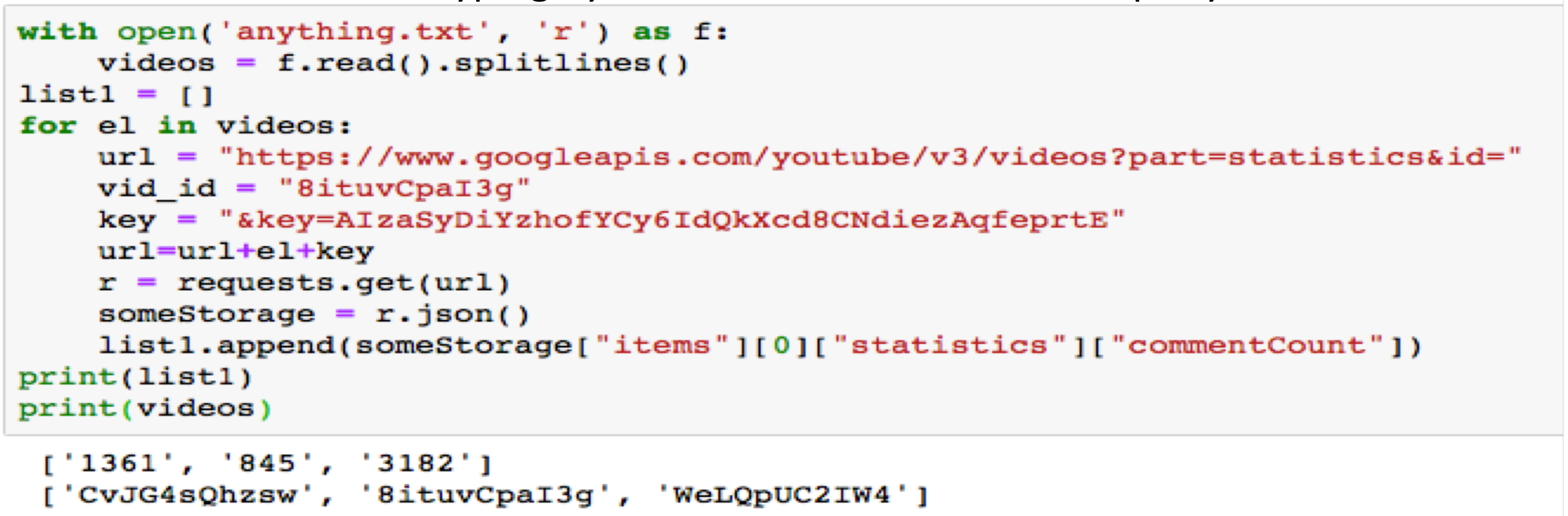
We can also read a list from file. If have a txt file, we need to open it (with open(‘filename.extension’, ‘r’)) and read every line to a list (in our case videos)



```
with open('anything.txt', 'r') as f:
     videos = f.read().splitlines()
videos
```

```
['CvJG4sQhzsw', '8ituvCpaI3g', 'WeLQpUC2IW4']
```

And that allows us to avoid typing by hand needed values for our query



```
with open('anything.txt', 'r') as f:
     videos = f.read().splitlines()
list1 = []
for el in videos:
     url = "https://www.googleapis.com/youtube/v3/videos?part=statistics&id="
     vid_id = "8ituvCpaI3g"
     key = "&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE"
     url=url+el+key
     r = requests.get(url)
     someStorage = r.json()
     list1.append(someStorage["items"][0]["statistics"]["commentCount"])
print(list1)
print(videos)
```

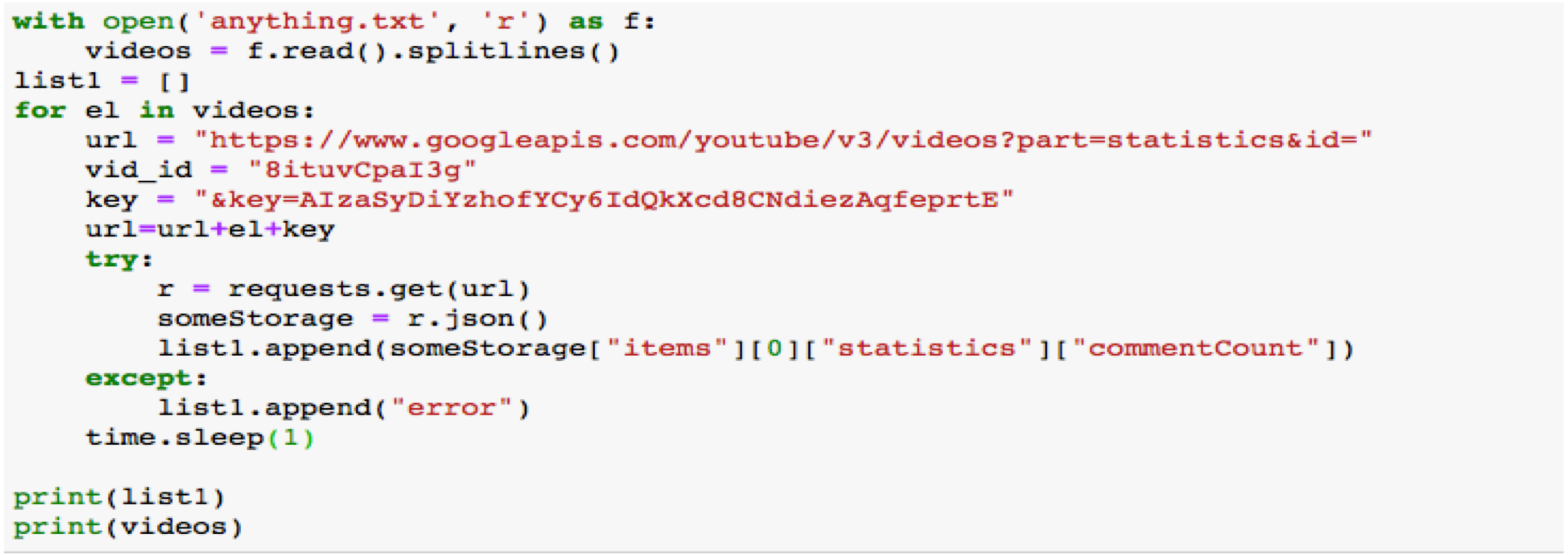
```
['1361', '845', '3182']
['CvJG4sQhzsw', '8ituvCpaI3g', 'WeLQpUC2IW4']
```

Try with Wikipedia API

Search for different titles: “api”, “python”, “cancer”

https://en.wikipedia.org/w/api.php?action=query&list=search&format=json&srsearch=**process**

To avoid some errors which could happen due to unresponded server, or missing element, we can include try/except statement. What is inside “try:” will happen unless there is an error, then the program won’t stop completely but continue in “except:” section



```
with open('anything.txt', 'r') as f:
    videos = f.read().splitlines()
list1 = []
for el in videos:
    url = "https://www.googleapis.com/youtube/v3/videos?part=statistics&id="
    vid_id = "8ituvCpaI3g"
    key = "&key=AIzaSyDiYzhofYCy6IdQkXcd8CNdiezAqfeprtE"
    url=url+el+key
    try:
        r = requests.get(url)
        someStorage = r.json()
        list1.append(someStorage["items"][0]["statistics"]["commentCount"])
    except:
        list1.append("error")
    time.sleep(1)

print(list1)
print(videos)
```

Additionally we could “import time” and set some waiting time, as server might not respond immediately and we could miss results

- Open Data sets
  - *https://www.quora.com/Where-can-I-find-large-datasets-open-to-the-public*
  - *https://github.com/caesar0301/awesome-public-datasets#social-networks*
- Lists of API's
  - *https://console.developers.google.com/apis/library*
  - *https://www.programmableweb.com/apis*

Tasks for Friday:

- Install selenium:

  In terminal ”pip install selenium”

- Download geckodriver and extract the package

  https://github.com/mozilla/geckodriver/releases