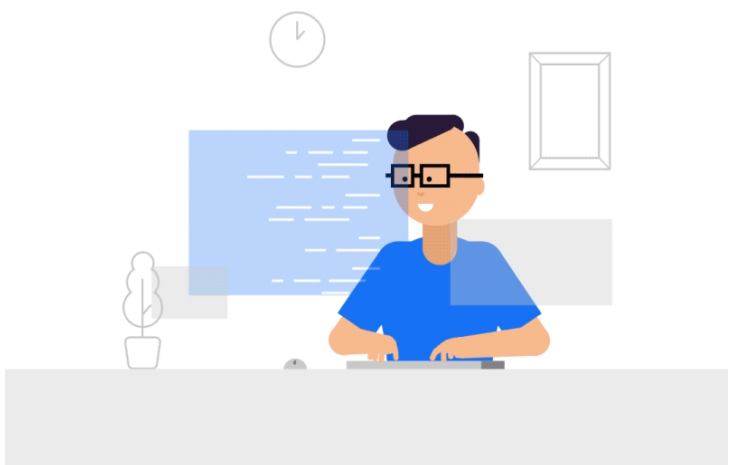
04 API

Scripting

Elke Boonen, Dries Geysens & Tristan Vandevelde

Learning objectives (ECTS)



- Learn the **syntax of Python**
- **Explore possibilities** regarding working with **data** (file, web API, web scraping...)
- Create **scripts**
- **Automate** processes

Learning materials

- **Canvas** LMS https://thomasmore.instructure.com/
  - Presentations & Assignments (CodeGrade)
  - E-Book: Automate the boring stuff with Python (https://automatetheboringstuff.com)
- **Online** https://www.python.org/ and https://pypi.org/ (search packages)
- **Github** Scripting (code from slides) & Scripting-Students (code from class)
- **Software**
  - https://www.python.org/downloads/ (IDLE Python)
  - https://www.anaconda.com/
  - Love to work with Visual Studio Code? Install Python extensions!

Schedule

Before Easter break	**After Easter break**
01 Python basics	08 Scheduling
02 Python basics	09 Images
03 Python basics	10 Databases
04 Strings & regex	11 Packaging
05 Files (IO)	12 Conclusion/exam prep
06 API	*13 Extra recap*
07 Webscraping	

Schedule is always subject to unexpected circumstances

Evaluation

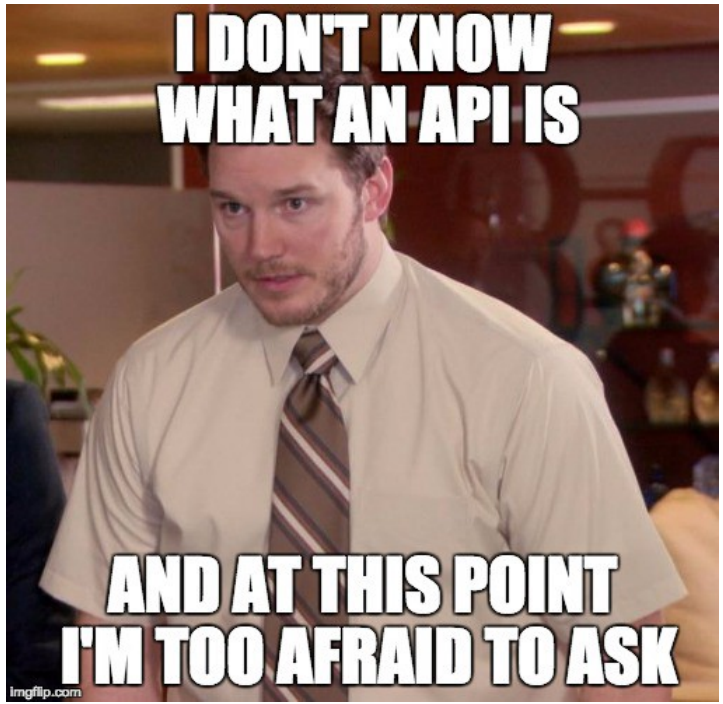
- **1st term**
  - Permanent Evaluation (30 %):
    - CodeGrade exercises (each week)
  - Take-home exam (70 %)
- **2nd term**
  - Computer Exam (100 %)

04 API

- What is a (web)API?
- Working with JSON & XML
- Doing the fun stuff

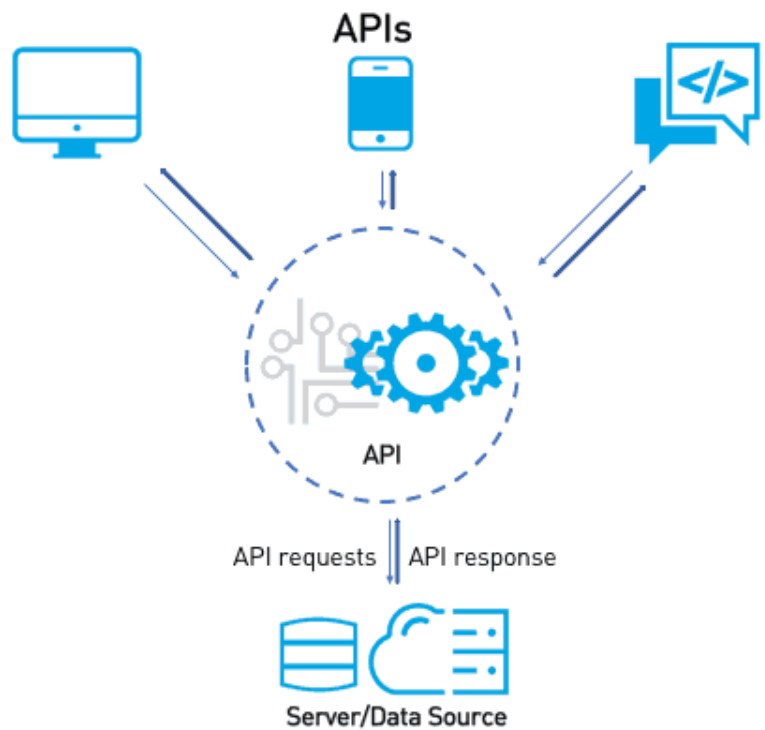
What is an API?

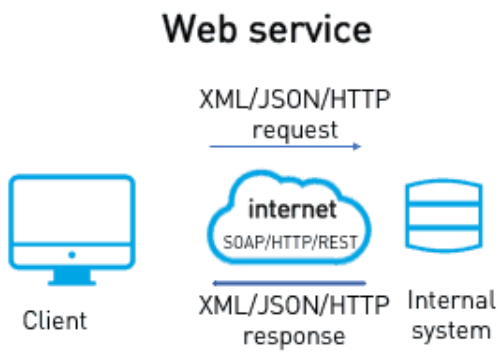
- Application Programming Interface
- Interface that allows you to build on the data and functionality of another application
- Some examples:
  - We can print our artwork of Paint ;) but Paint itself doesn't need to know how to handle the printer itself, it uses the Print-API!
  - Skyscanner, cheaptickets, momondo.... all gather information about flights, but don't have to know squat about airports ;)



What is a (web)API?

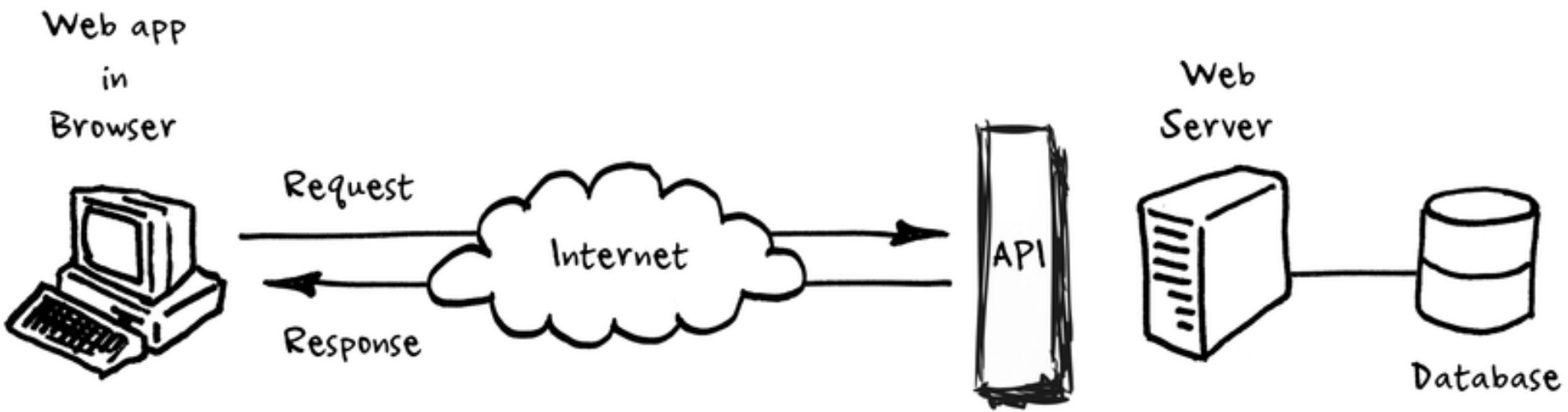
- A webservice = an API over the internet!
- All webservices are API's but now all API's are webseries!





Send a request, get a response!

- HTTP request and response
- HTTP response body consists of XML/JSON or regular text

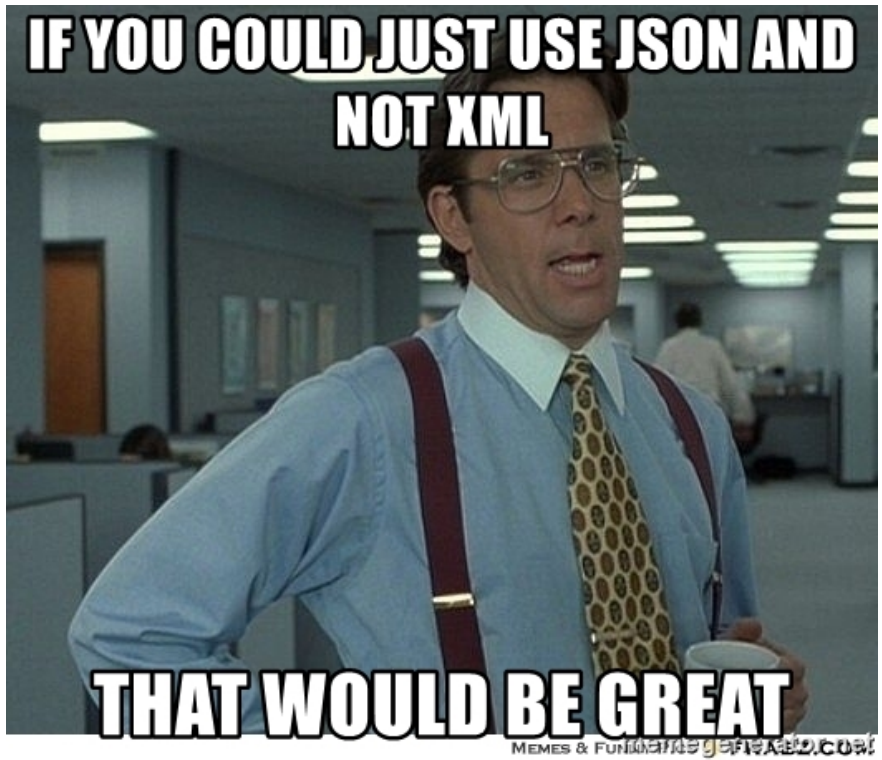


04 API

- ~~What is a (web)API?~~
- Working with JSON & XML
- Doing the fun stuff

XML & JSON

- XML: eXtensible Markup Language
  - similar to HTML, works with tags
- JSON: JavaScript Object Notation
- Both are readable & self-describing
- JSON is the 'new' kid in town!



XML

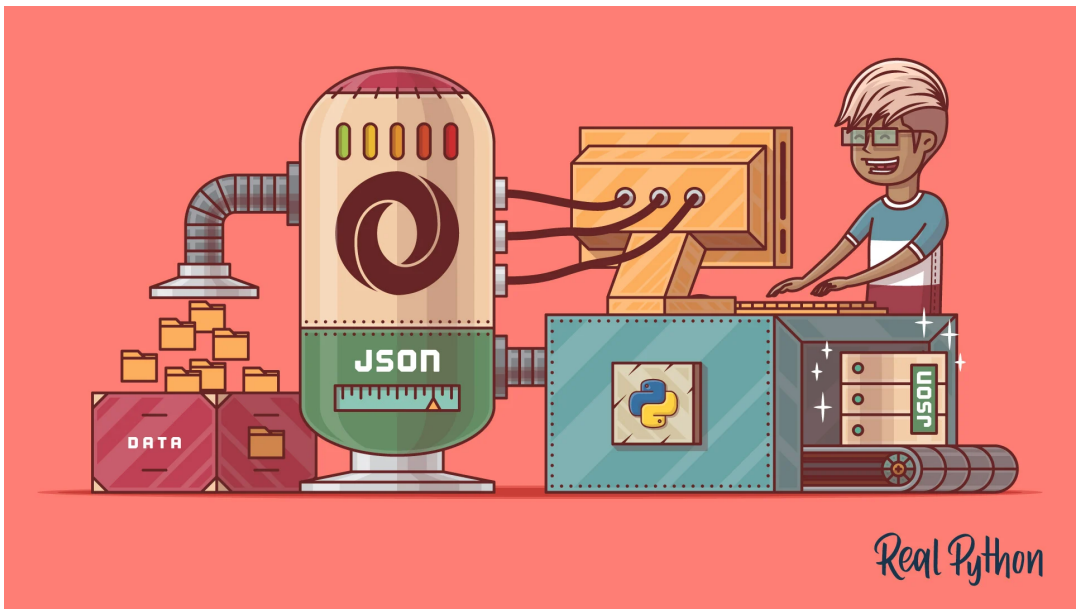
```
<empinfo>
   <employees>
      <employee>
         <name>James Kirk</name>
         <age>40></age>
      </employee>
      <employee>
         <name>Jean-Luc Picard</name>
         <age>45</age>
      </employee>
      <employee>
         <name>Wesley Crusher</name>
         <age>27</age>
      </employee>
   </employees>
</empinfo>
```

JSON

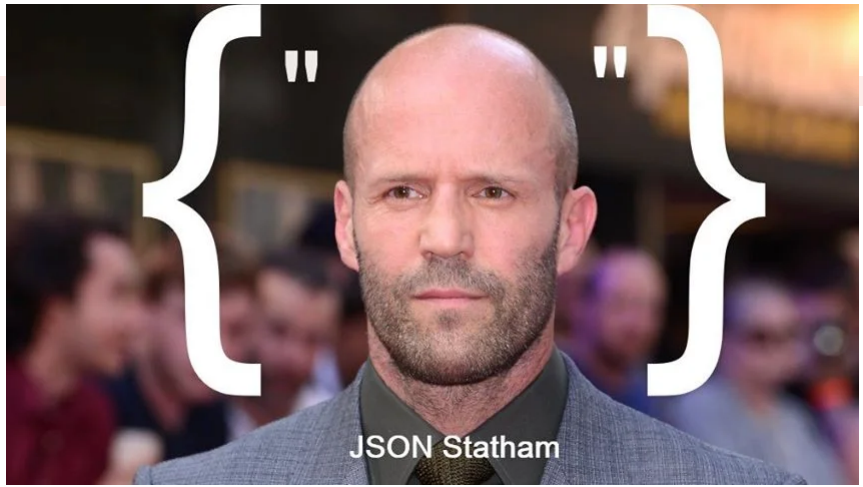
```
{  "empinfo" :
      {
            "employees" :  [
            {
                  "name" : "James Kirk",
                  "age" : 40,
            },
            {
                  "name" : "Jean-Luc Picard",
                  "age" : 45,
            },
            {
                  "name" : "Wesley Crusher",
                  "age" : 27,
            }
                              ]
      }
}
```

JSON module

- **Translation** between string with **JSON data and Python values**
- JSON can store/read the following types: strings, integers, floats, Booleans, lists, dictionaries, and NoneType, but no Python-specific objects, such as File objects, Regex objects etc...



Read JSON



- Import module **json**
- Use load() or loads()
  - load: file load
  - loads: string load

    double quotes for key/values!
- It returns the data as a Python dictionary

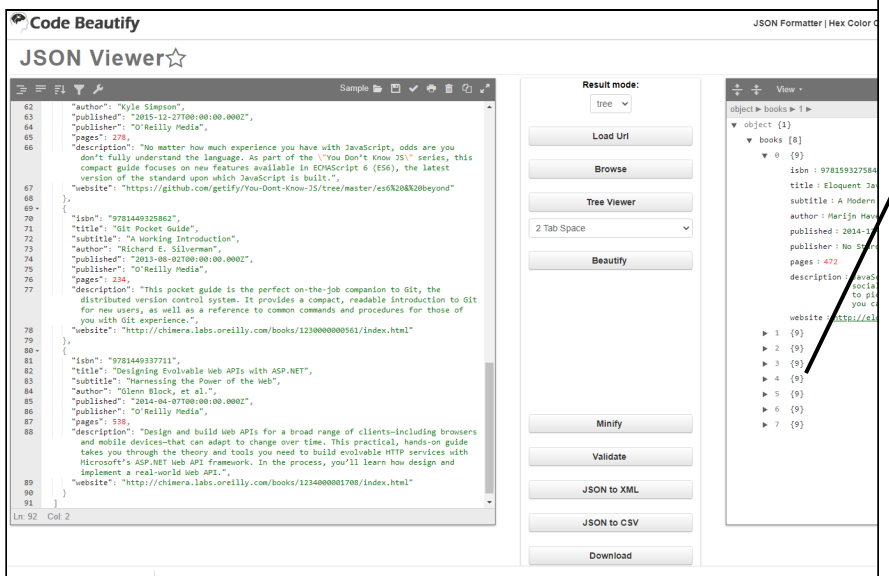
  Be aware: Python dictionaries are not ordered!

```
1 import json
2
3 #string JSON example
4 json_data =  '{ "name":"Jason Statham", "age":54, "city":"Chesterfield"}'
5 y = json.loads(json_data)
6 print(y["age"]) # result is dictionary!
```

54

Read JSON from a file

- First, always try to understand the structure of your JSON data
- https://codebeautify.org/jsonviewer



Read JSON from a file

- Open file
  - pass file-object to load()
  - read content and pass to loads()
- Do something with data!
  - print specific part
  - print all
  - loop dictionary
  - ...

```
Eloquent JavaScript, Second Edition
Learning JavaScript Design Patterns
Speaking JavaScript
Programming JavaScript Applications
Understanding ECMAScript 6
You Don't Know JS
Git Pocket Guide
Designing Evolvable Web APIs with ASP.NET
```

```
1  import json
2
3
4  json_file = open("books.json")
5
6  json_data = json.load(json_file)          # option 1
7  json_data = json.loads(json_file.read()) # option 2
8
9  print(json_data["books"][0]["title"])
10
11 print(json_data)
12
13
14
15
16
17 for book in json_data["books"]:
18     print(book["title"])
19
20 # don't forget closing the file
21 json_file.close()
```

```
Eloquent JavaScript, Second Edition
```

```
{'books': [{'isbn': '9781593275846', 'title': 'Eloquent JavaScript, Se
ion to Programming', 'author': 'Marijn Haverbeke', 'published': '2014
Press', 'pages': 472, 'description': 'JavaScript lies at the heart of
```

JSON write

- Start from a dictionary (or a dictionary of a list of dictionaries :))
- Use dump(): convert python data to a string
  - see any difference between python_data and json_data?

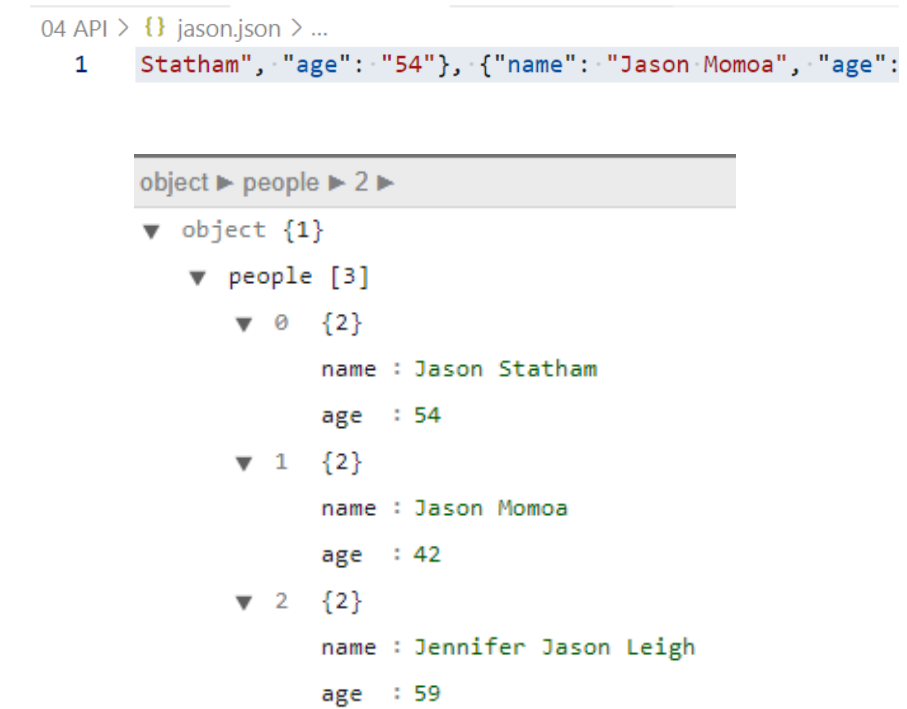
```
1 import json
2
3 python_data = { "name":"Jason Statham", "age":54, "city":"Chesterfield"}
4
5 json_data = json.dumps(python_data)
6 print(python_data)
7 print(json_data)
```

```
{'name': 'Jason Statham', 'age': 54, 'city': 'Chesterfield'}
{"name": "Jason Statham", "age": 54, "city": "Chesterfield"}
```

JSON write

- Start from a dictionary (or a dictionary of a list of dictionaries :))
- Use dumps(): convert python data to a string and write it to JSON file

```python
import json

#creating a dictionary of a list of dictionaries
python_data = {}
python_data['people'] = []
python_data['people'].append({
    'name': 'Jason Statham',
    'age': '54'
})
python_data['people'].append({
    'name': 'Jason Momoa',
    'age': '42'
})
python_data['people'].append({
    'name': 'Jennifer Jason Leigh',
    'age': '59'
})

json_file = open('jason.json', 'w')
json_data = json.dump(python_data, json_file)
json_file.close()
```

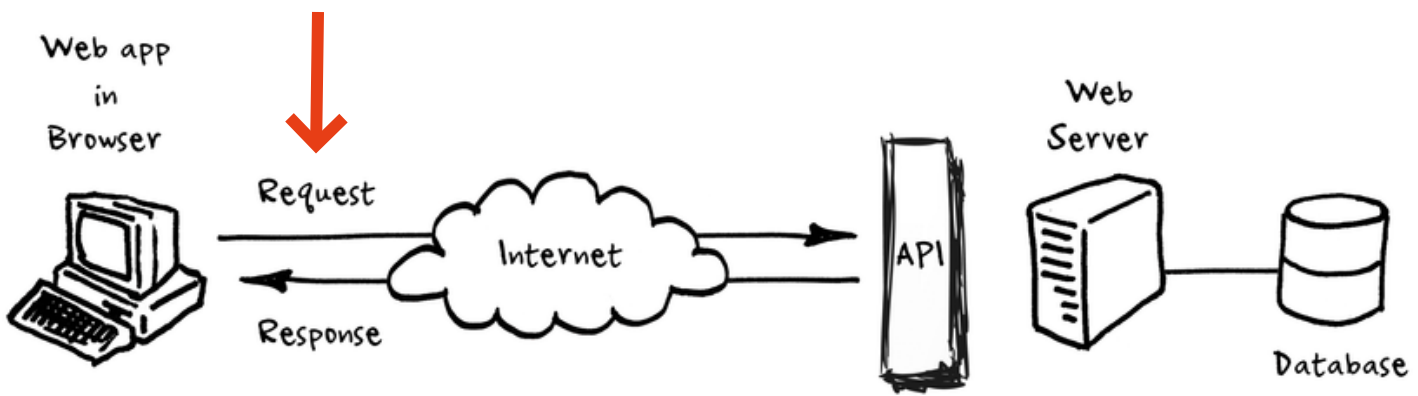


04 API

- ~~What is a (web)API?~~
- ~~Working with JSON & XML~~
- Doing the fun stuff

We have a request!

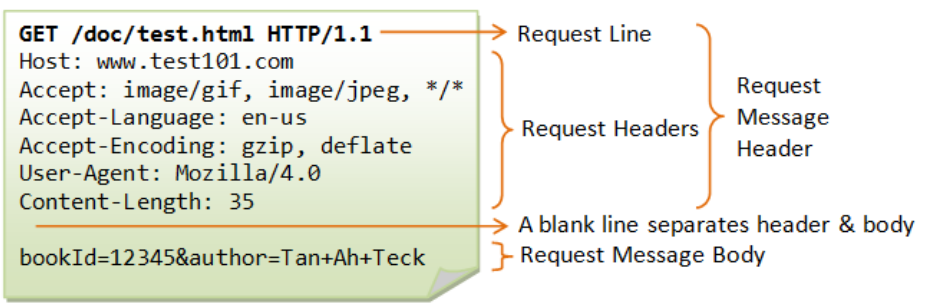
- Asking for data from an API = making a request!
- Install Python package: **requests**
  - pip install requests
- Requests module sends HTTP requests (and gets responses back!)



We have a request!

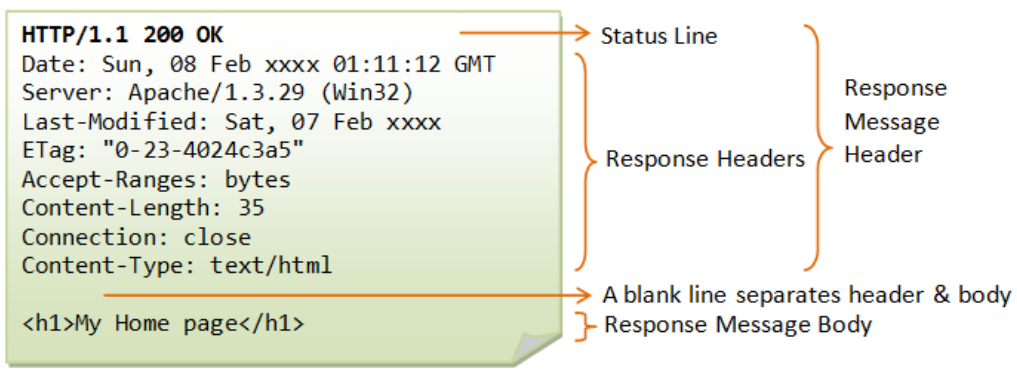
- 'Only' reading data from API = **get**!
- HTTP request
  - text-message to server
  - important! https, eg: 'scramble text' (s not important when reading data, unless it is FBI data ;))
- Check usage in API docs!

Method	Meaning
GET	Read data
POST	Insert data
PUT or PATCH	Update data, or insert if a new id
DELETE	Delete data



We get a response

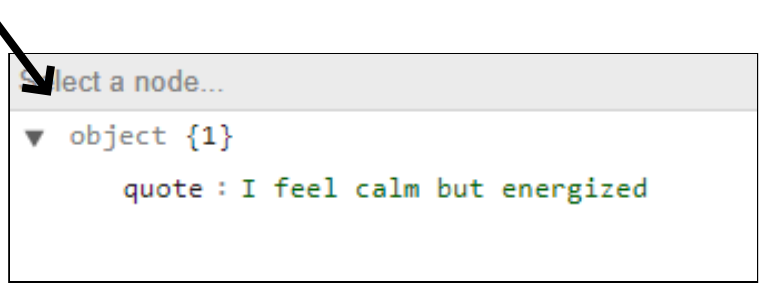
- Get response back from webserver
- HTTP response
  - status = number
  - message body --> this is where the magic happens with JSON/XML



STATUS CODE	EXPLANATION
200 - OK	The request succeeded.
204 - No Content	The document contains no data.
301 - Moved Permanently	The resource has permanently moved to a different URI.
401 - Not Authorized	The request needs user authentication.
403 - Forbidden	The server has refused to fulfill the request.
404 - Not Found	The requested resource does not exist on the server.
408 - Request Timeout	The client failed to send a request in the time allowed by the server.
500 - Server Error	Due to a malfunctioning script, server configuration error or similar.

Kanye.rest

- Check https://kanye.rest/ & check usage
- A useless free API for random Kanye West quotes ;)
- Load URL in JSON viewer to check JSON tree!



API Usage

GET https://api.kanye.rest Tweet Refresh

```
{
    "quote": "I honestly need all my Royeres to be museum quality... if I see a fake Royere
    Ima have to Rick James your couch"
}
```

Extra: https://api.kanye.rest?format=text

Kanye.rest

- Import requests and json
- Use get-request!
- Check response!
  - what if URL is wrong?
  - exception handling!

```python
import requests
import json

response = requests.get("https://api.kanye.rest/")
print(response.status_code)
print(response.headers)
print(response.text)
print(response.json)

json_data = json.loads(response.text)
print(json_data["quote"])
```

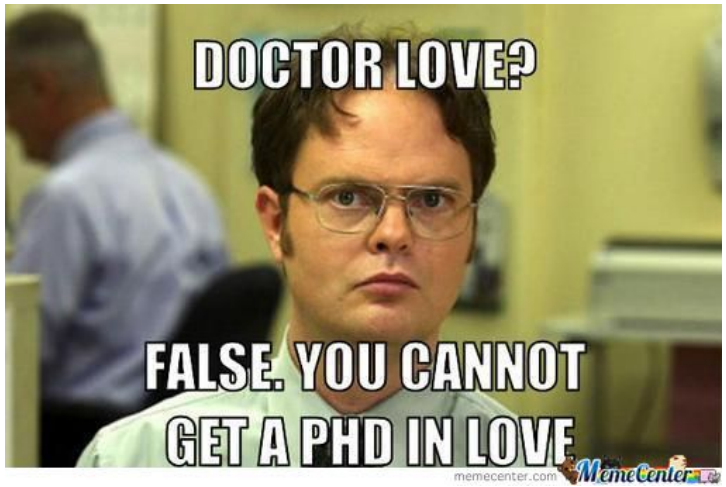
```
200
{'Date': 'Mon, 22 Mar 2021 14:51:16 GMT',
'Set-Cookie': '__cfduid=d65c3405cd644c921e
est; HttpOnly; SameSite=Lax', 'Access-Cont
-Allow-Methods': 'GET', 'cf-request-id': '
port-uri.cloudflare.com/cdn-cgi/beacon/exp
s:\\/\\/a.nel.cloudflare.com\\/report?s=bY
l43Ao9Gxah6bVIFmfThPTLj%2BaurBChOw%2F7g%3D
'Server': 'cloudflare', 'CF-RAY': '63403cb
3"; ma=86400'}
{"quote":"I'm the new Moses"}
I'm the new Moses
```

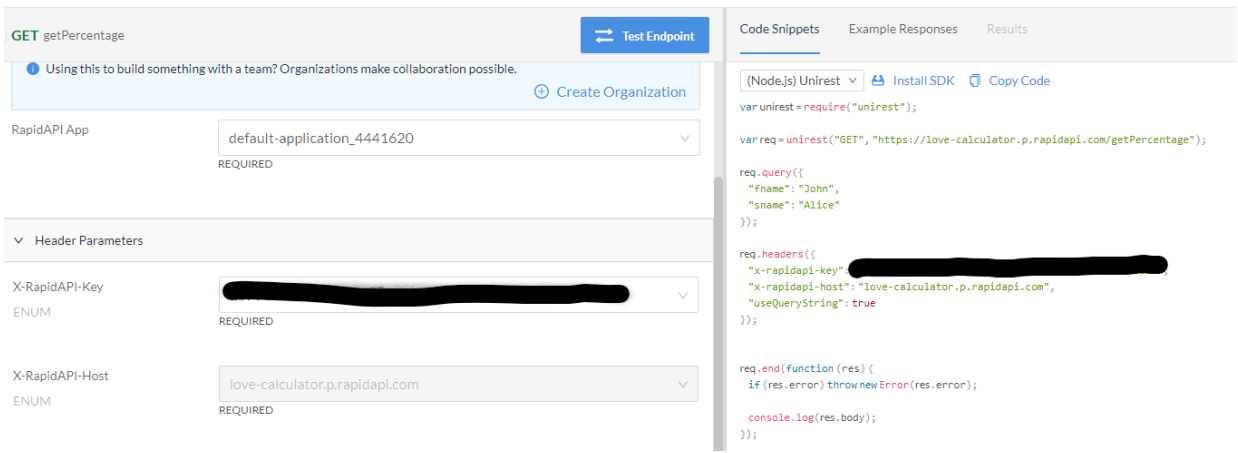
```python
import requests
import json

try:
    r = requests.get("https://api.kanyee.rest/")
except Exception as e:
    print(e)
```

```
HTTPSConnectionPool(host='api.kanyee.rest', port=443):
onnection.HTTPSConnection object at 0x000002AFBA7E65B0
```

Dr. Love aka the love calculator

- Sometimes you need an API-key!
- Create an account on https://rapidapi.com/
- The love calculator!
  - https://rapidapi.com/ajith/api/love-calculator
  - Check credentials & usage!





Dr. Love aka the love calculator

- Build http-request with credentials!
  - url
  - headers with key
  - query-data!
- Check the URL!

```
1  import requests
2  import json
3
4  api_url = "https://love-calculator.p.rapidapi.com/getPercentage"
5  api_headers = {
6          "x-rapidapi-key": "***",
7          "x-rapidapi-host": "love-calculator.p.rapidapi.com"}
8  api_query = {
9          "fname": "Elke",
10         "sname": "Christophe" }
11
12 response = requests.get(api_url, headers = api_headers, params=api_query)
13 print(response.url)
14 print(response.text)
```

https://love-calculator.p.rapidapi.com/getPercentage?fname=Elke&sname=Christophe

Dr. Love aka the love calculator

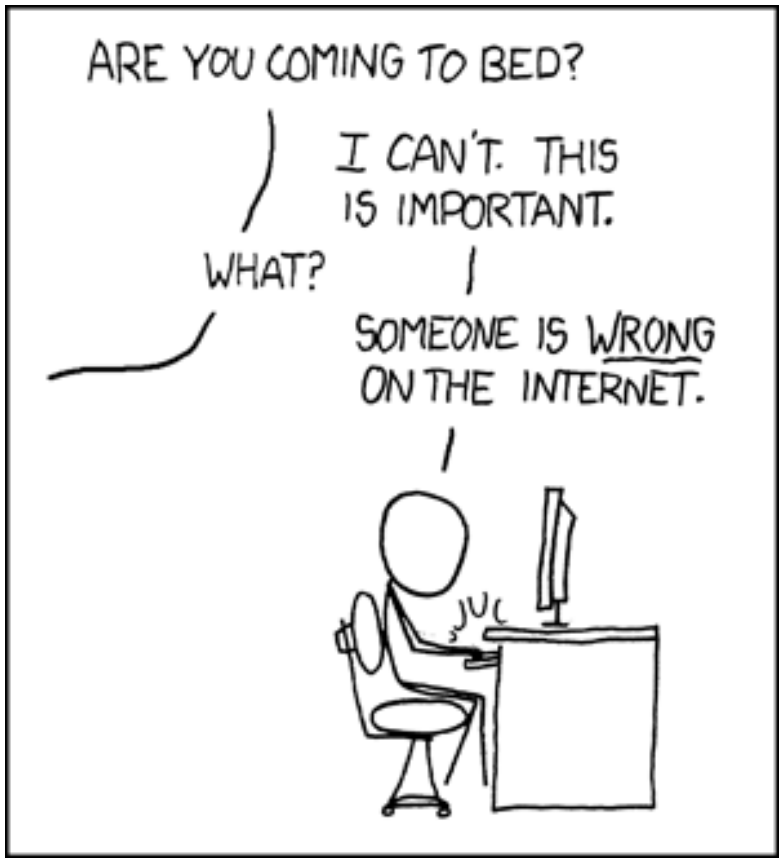
- Go wild ;)

¢σηgяατυℓατι σηs! gσσ∂ ¢нσι¢є,тнιs ιs sαмρℓє ƒαη¢y тєχт.

```
1  import requests
2  import json
3
4  api_url = "https://love-calculator.p.rapidapi.com/getPercentage"
5  api_headers = # your own credentials
6
7  fname = input("Enter first name")
8  sname = input("Anter second name")
9
10 api_query = {
11         "fname": fname,
12         "sname": sname }
13
14 response = requests.get(api_url, headers = api_headers, params=api_query)
15 json_data = json.loads(response.text)
16 result = json_data["result"]
17
18
19 api_url = "https://ajith-fancy-text-v1.p.rapidapi.com/text"
20 api_headers = # your own credentials
21
22 api_query = {
23         "text": result}
24
25 response = requests.get(api_url, headers = api_headers, params=api_query)
26 json_data = json.loads(response.text)
27 print(json_data["fancytext"])
```

API's or packages?

- Some free APIs also have Python packages
- eg: XKCD (webcomics)
  - API : https://xkcd.com/json.html
  - Python package: pip install xkcd (https://pypi.org/project/xkcd/)
- Always go for the API (no installation needed!) because most of the time the package is just a wrapper!



Practice makes perfect!

- *Check out the extra (not graded) exercises in the notebooks!* *See Github*
- Do your exercises, spend the hours!

Say what? How many hours?

6 SP = 6 * 28 hours = 168 hours

Lessons = 12 * 3 hours = 36 hours

Take-home exam = 16 hours

Exercise = 168-36-10 = 116 hours

codegrade