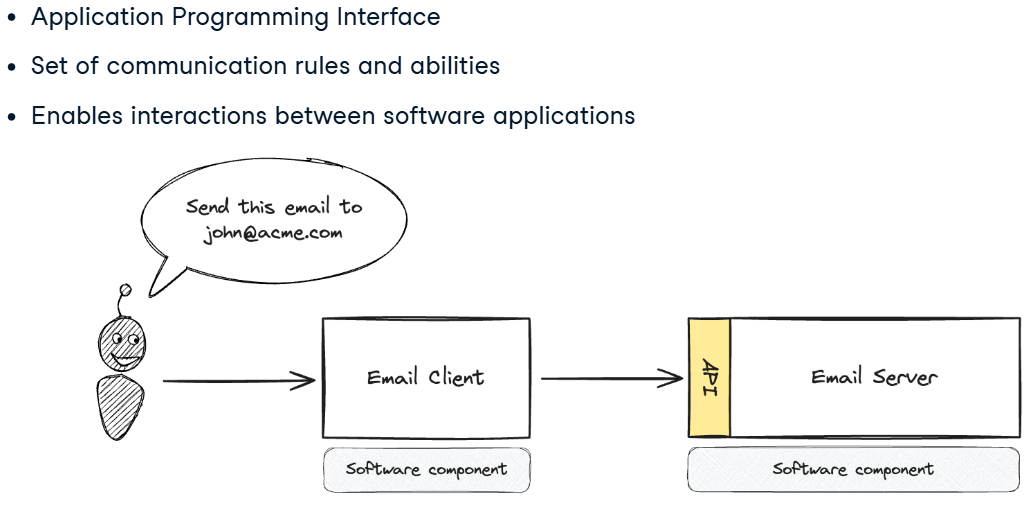
## Introduction to APIs in Python

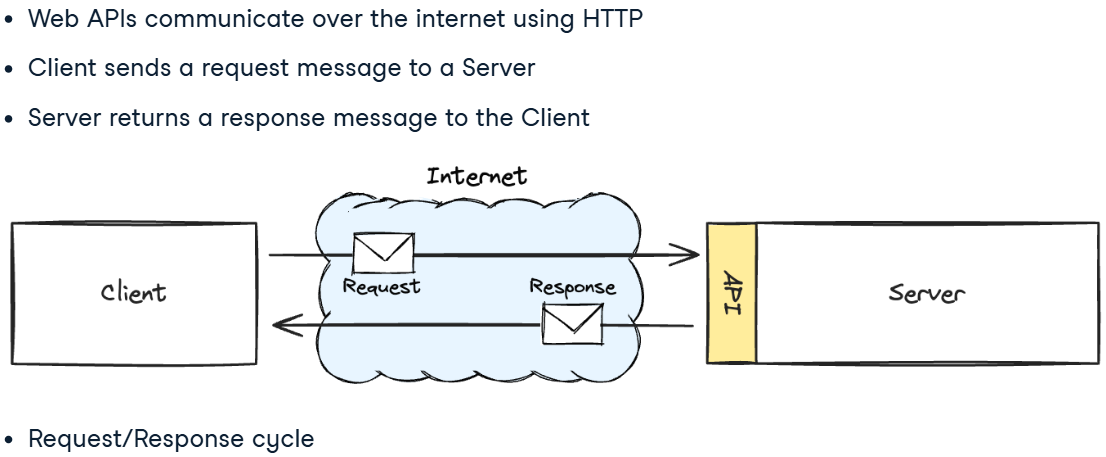
### Making API Requests with Python

#### Introduction to APIs

* What is an API:



* Web APIs, clients and servers:



* Code for **requests**:

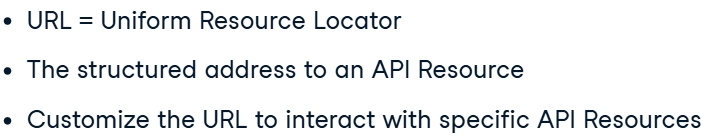
import requests

response = requests.get('http://localhost:3000/lyrics')

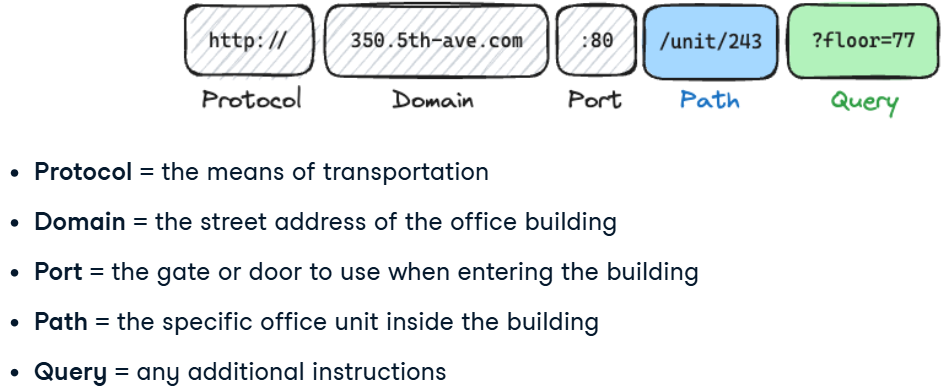
print(response.text)

#### The basic anatomy of an API request

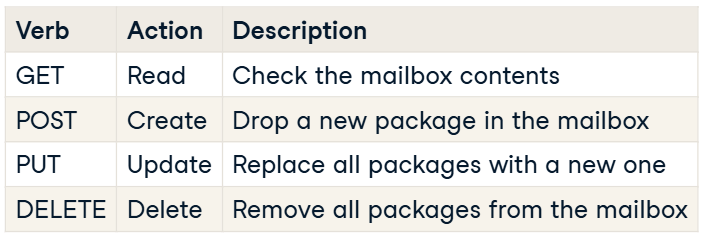
* What is an URL:



* Elements of the URL:



* HTTP verbs (actions):



* Getting content with additional parameters:

query\_params = {'artist': 'Deep Purple',

                'include\_track': True}

response = requests.get('http://localhost:3000/lyrics/random', params=query\_params)

print(response.url)

print(response.text)

* Code:

response = requests.get('http://localhost:3000/playlists')

playlist\_data = {'Name': 'Rock Ballads'}

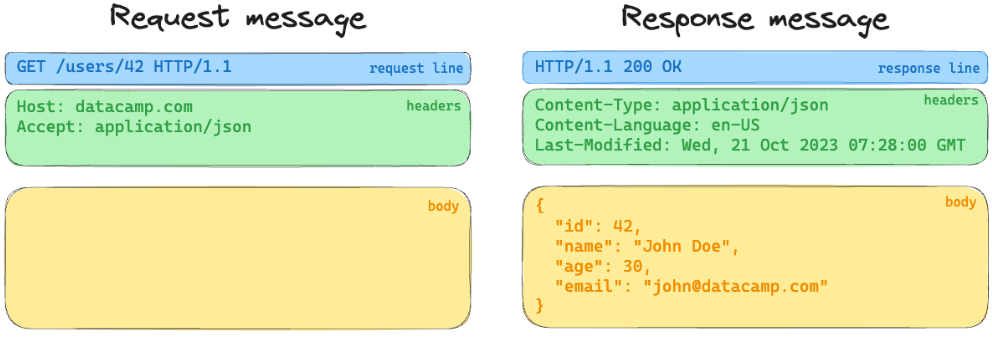
response = requests.post('http://localhost:3000/playlists', data=playlist\_data)

requests.delete('http://localhost:3000/playlists/2')



#### Headers and status codes

* An anatomy of requests:
  + “accept” and “content-type” is used for negotiations:



* Getting the request object:



* Code for getting the response:

response = requests.get('http://localhost:3000/movies')

if (response.status\_code == requests.codes.ok):

  print('The server responded succesfully!')

elif (response.status\_code == requests.codes.not\_found):

  print('Oops, that API could not be found!')

* Headers with requests (get and post):

headers = {'accept': 'application/json'}

response = requests.get('http://localhost:3000/lyrics', headers=headers)

print(response.text)

* Error checking and finding formats for server acceptance:

headers = {'accept':'application/xml'}

response = requests.get('http://localhost:3000/lyrics', headers=headers)

if (response.status\_code == 406):

  print('The server can not respond in XML')

  print('These are the content types the server accepts: ' + response.headers['accept'])

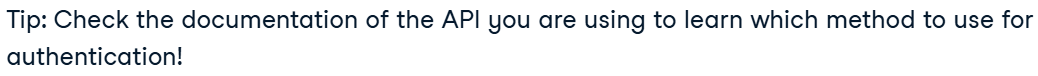
else:

  print(response.text)

### More API request concepts

#### API Authentication

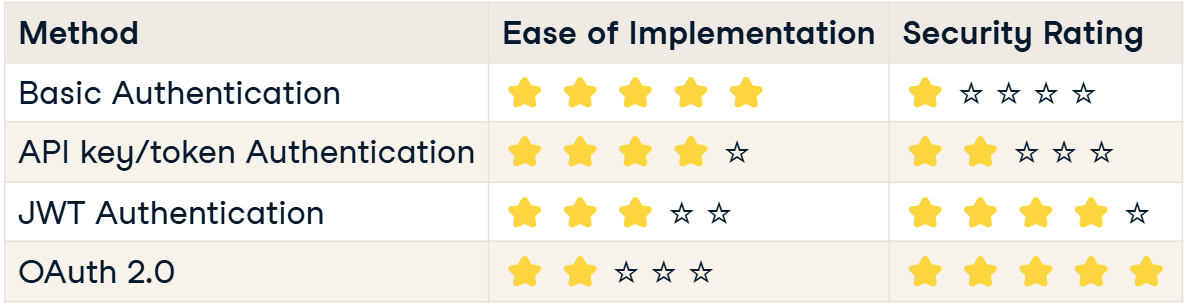
* Working with APIs when authenticate:



* Basic authentication:



* Authentication methods:



* + Basic authentication:

authentication = ('john@doe.com', "Warp\_ExtrapolationsForfeited2")

response = requests.get('http://localhost:3000/albums', auth=authentication)

* + API key-based authentication:

params = {"access\_token":'8apDFHaNJMxy8Kt818aa6b4a0ed0514b5d3'}

response = requests.get('http://localhost:3000/albums', params=params)

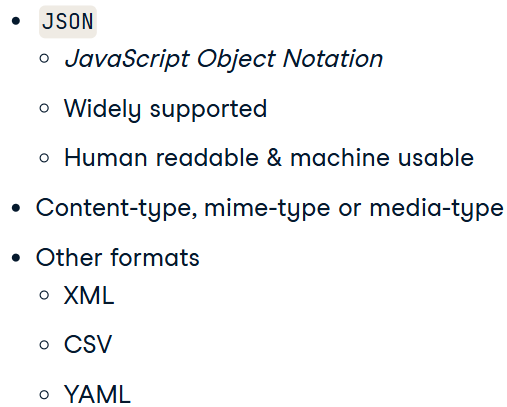
/

headers = {'Authorization': 'Bearer 8apDFHaNJMxy8Kt818aa6b4a0ed0514b5d3'}

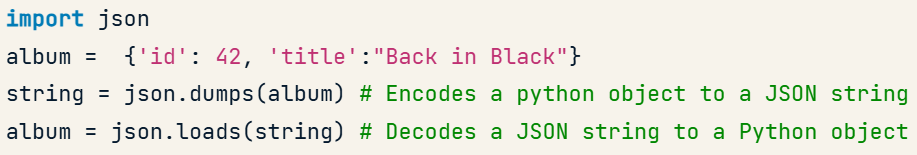
response = requests.get('http://localhost:3000/albums', headers=headers)

#### Working with structured data

* Complex data structures:



* Encoding and decoding JSON objects:



* Get JSON file:

headers = {

    'Authorization': 'Bearer ' + API\_TOKEN,

    'Accept': 'application/json'

}

response = requests.get('http://localhost:3000/albums/1/', headers=headers)

album = response.json()

print(album['Title'])

* From JSON to DataFrame:

import pandas as pd

import json

headers = {'accept': 'application/json'}

response = requests.get('http://localhost:3000/lyrics', headers=headers)

data = json.loads(response.text)

df = pd.json\_normalize(data)

df

result:



* Sending JSON data:

playlists = [

    {"Name":"Rock ballads"},

    {"Name":"My favorite songs"},

    {"Name":"Road Trip"}]

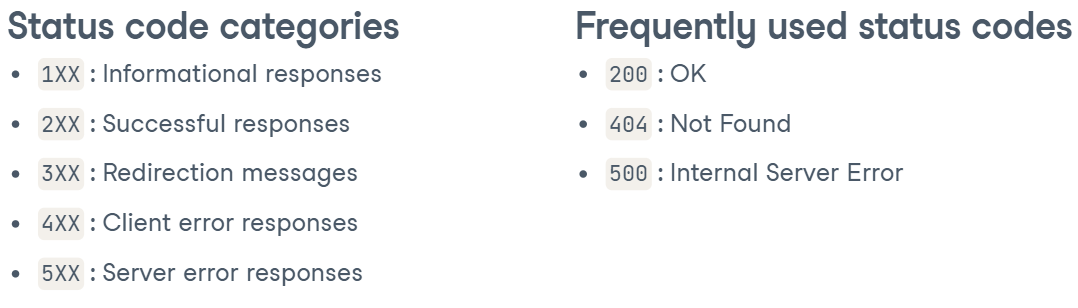
requests.post('http://localhost:3000/playlists/', json=playlists)

response = requests.get('http://localhost:3000/playlists')

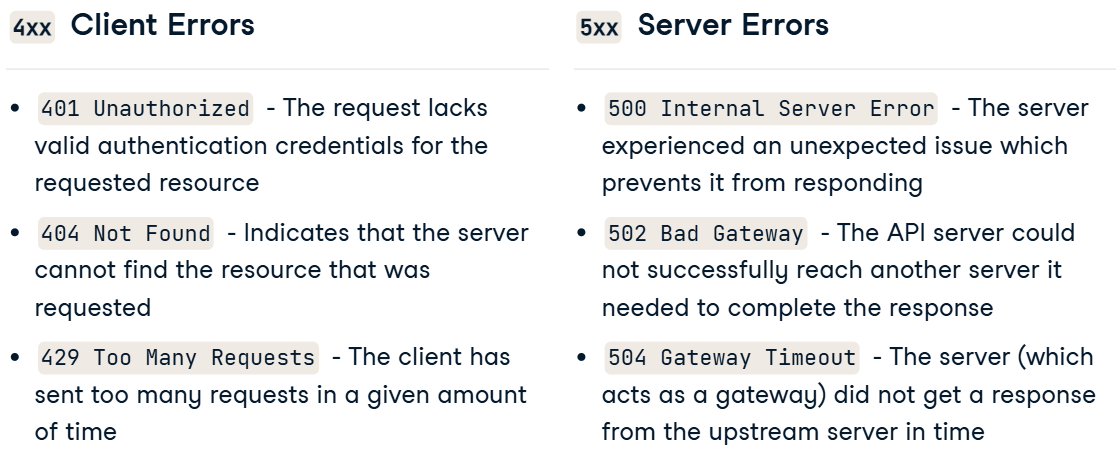
print(response.text)

#### Error handling

* Status codes:



* + Additional info:



* Code for handling the error:

from requests.exceptions import HTTPError, ConnectionError

url ="http://localhost:3000/albums/"

try:

    r = requests.get(url)

    print(r.raise\_for\_status())

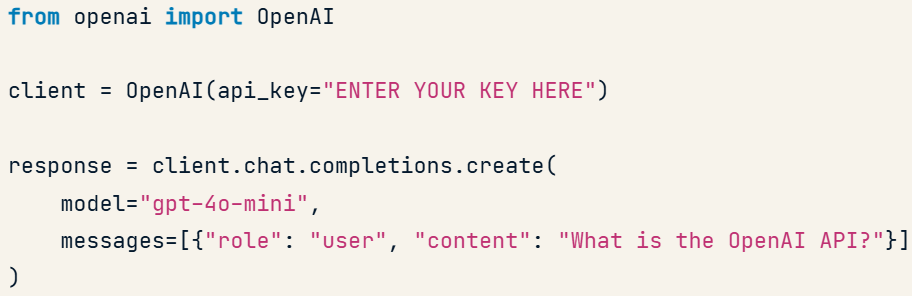
except HTTPError as http\_err:

    print(f'HTTP error occurred: {http\_err}')

## Introduction to the OpenAI API

### Making requests to the OpenAI API

* Making a request:



* + response:

print(response.choices[0].message.content)

### OpenAI's Text and Chat Capabilities

#### Generating and transforming text

* Contolling response randomness:

