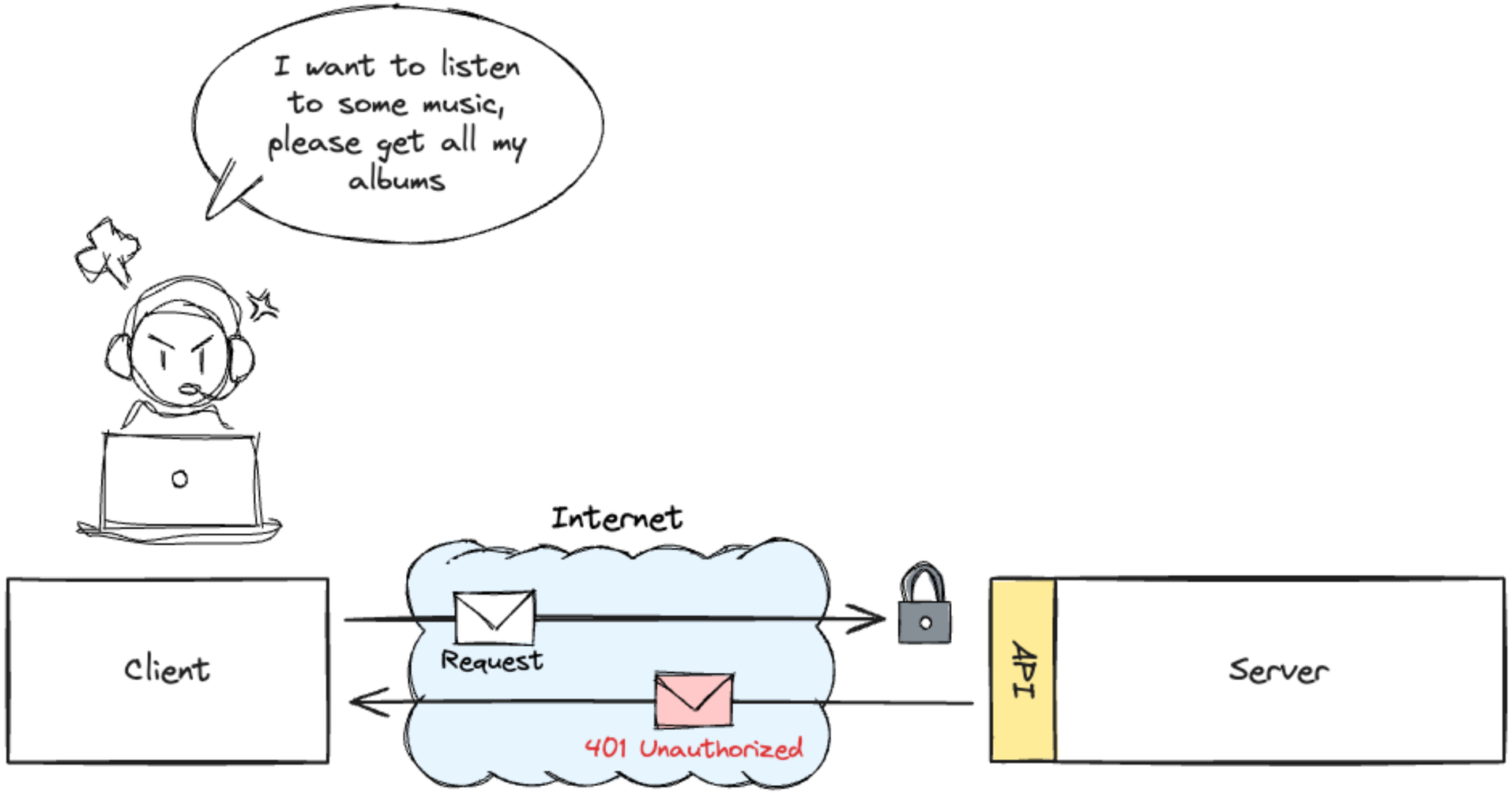
API Authentication

INTRODUCTION TO APIS IN PYTHON

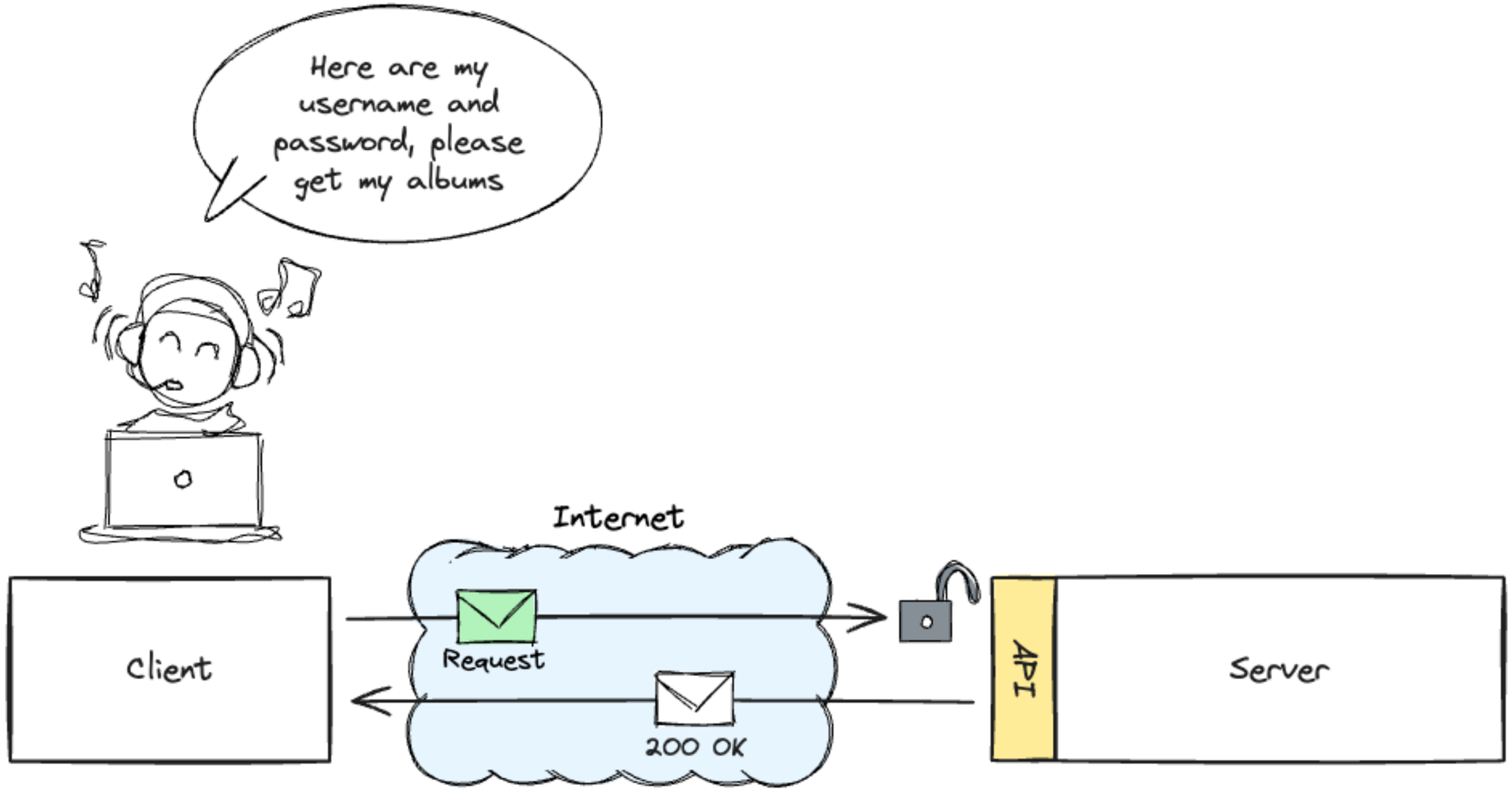
Chris Ramakers
Engineering Manager

datacamp

Accessing sensitive data



Accessing sensitive data

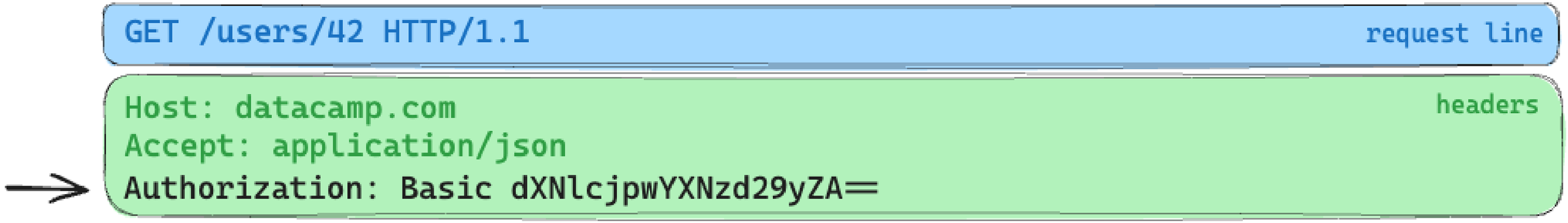


Authentication methods

Method	Ease of Implementation	Security Rating
Basic Authentication		☆ ☆ ☆ ☆
API key/token Authentication	☆	☆ ☆ ☆
JWT Authentication	☆ ☆	☆
OAuth 2.0	☆ ☆ ☆	

Tip: Check the documentation of the API you are using to learn which method to use for authentication!

Basic authentication



Basic authentication with the requests package

```
# This will automatically add a Basic Authentication header before sending the request
requests.get('http://api.music-catalog.com', auth=('username', 'password'))
```

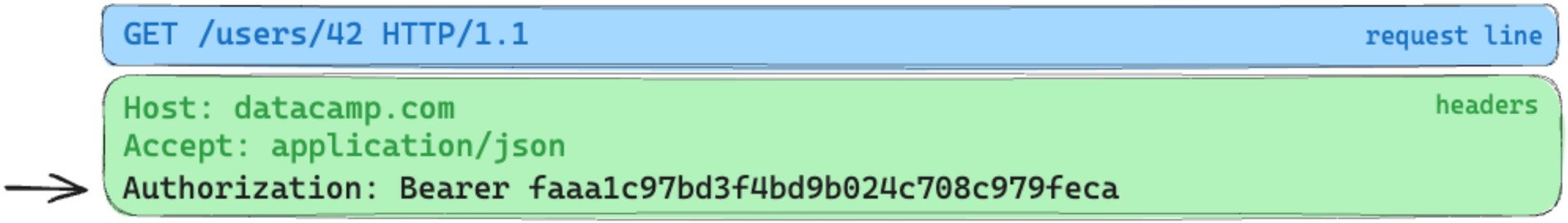
API key/token authentication

Using a query parameter

```
http://api.music-catalog.com/albums?access_token=faaa1c97bd3f4bd9b024c708c979feca
```

```
params = {'access_token': 'faaa1c97bd3f4bd9b024c708c979feca'}
requests.get('http://api.music-catalog.com/albums', params=params)
```

Using the "Bearer" authorization header




```
headers = {'Authorization': 'Bearer faaa1c97bd3f4bd9b024c708c979feca'}
requests.get('http://api.music-catalog.com/albums', headers=headers)
```

Let's practice!

INTRODUCTION TO APIS IN PYTHON

Working with structured data

INTRODUCTION TO APIS IN PYTHON

Chris Ramakers
Engineering manager

datacamp

Complex data structures

Lyric API response

```
HTTP/1.1 200 OK                                          response line
```

```
Content-Type: plain/text                                 headers
Content-Language: en-US
Last-Modified: Wed, 21 Oct 2023 07:28:00 GMT
```

```
N' I never miss Cause I'm a problem                      body
child - AC/DC, Problem Child
```

Album API response

```
HTTP/1.1 200 OK                                          response line
```

```
Content-Type: application/json                           headers
Content-Language: en-US
Last-Modified: Wed, 21 Oct 2023 07:28:00 GMT
```

```
{                                                        body
  "id": 42,
  "title": "Back in Black",
  "artist": "AC/DC",
  "tracks": [
    { "id": 1, "title": "Hells bells" },
    { "id": 2, "title": "Shoot to Thrill" },
    { "id": 3, "title": "What Do You  ..." },
    { "id": 4, "title": "Givin the Dog  ..." },
    { "id": 5, "title": "Let Me Put my  ..." }
  ]
}
```

Complex data structures: JSON

- `JSON`
  - *JavaScript Object Notation*
  - Widely supported
  - Human readable & machine usable
- Content-type, mime-type or media-type
- Other formats
  - XML
  - CSV
  - YAML

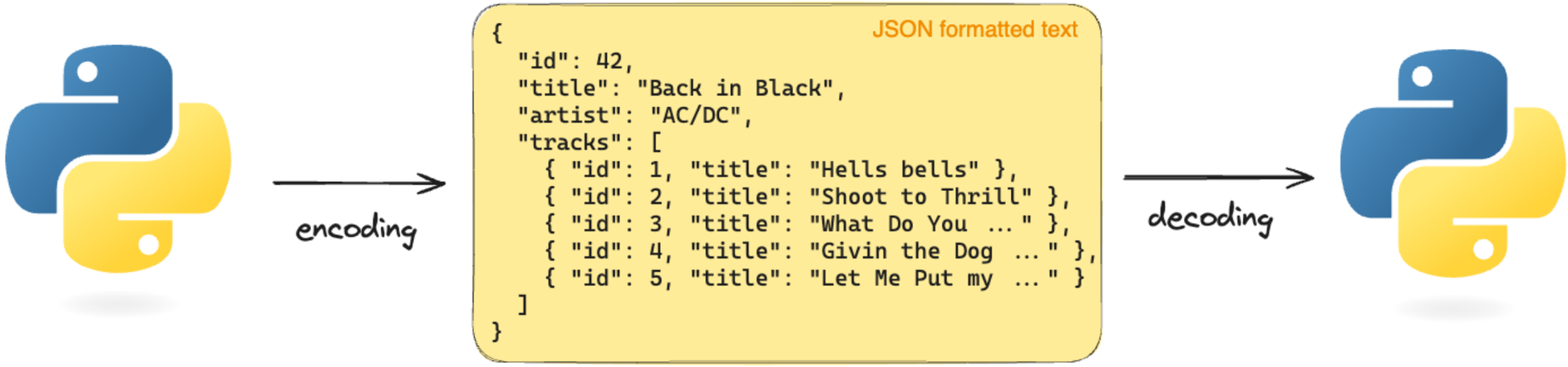
Album API response

```
HTTP/1.1 200 OK                                    response line
```

```
Content-Type: application/json                     headers
Content-Language: en-US
Last-Modified: Wed, 21 Oct 2023 07:28:00 GMT
```

```
{                                                  body
  "id": 42,
  "title": "Back in Black",
  "artist": "AC/DC",
  "tracks": [
    { "id": 1, "title": "Hells bells" },
    { "id": 2, "title": "Shoot to Thrill" },
    { "id": 3, "title": "What Do You  ..." },
    { "id": 4, "title": "Givin the Dog  ..." },
    { "id": 5, "title": "Let Me Put my  ..." }
  ]
}
```

From Python to JSON and back




```
import json
album =  {'id': 42, 'title':"Back in Black"}
string = json.dumps(album) # Encodes a python object to a JSON string
album = json.loads(string) # Decodes a JSON string to a Python object
```

Requesting JSON data

```
# GET request without headers
response = requests.get('http://api.music-catalog.com/lyrics')
print(response.text)
```

```
N' I never miss Cause I'm a problem child - AC/DC, Problem Child
```

```
# GET request with an accept header
response = requests.get('http://api.music-catalog.com/lyrics', headers={'accept': 'application/json'})

# Print the JSON text
print(response.text)

# Decode into a Python object
data = response.json()
print(data['artist'])
```

```
{'artist': 'AC/DC', 'lyric': "N' I never miss Cause I'm a problem child", 'track': 'Problem Child'}
AC/DC
```

Sending JSON data

```
import requests
playlist = {"name": "Road trip", "genre":"rock", "private":"true"}

# Add the playlist using via the `json` argument
response = requests.post("http://api.music-catalog.com/playlists", json=playlist)
```

```
# Get the request object
request = response.request

# Print the request content-type header
print(request.headers['content-type'])
```

```
application/json
```

Let's practice!

INTRODUCTION TO APIS IN PYTHON

Error handling

INTRODUCTION TO APIS IN PYTHON

Chris Ramakers
Engineering Manager

datacamp

Error status codes

`4xx` Client Errors

- Indicate issues on the client's end
- Common causes: Bad requests, authentication failures, etc ...

Resolution: Fix the request

`5xx` Server Errors

- Arises from problems on the server
- Common causes: Server overloaded, server configuration errors, internal errors

Resolution: Should be fixed by the API administrator

Error status codes: examples

`4xx` Client Errors

- `401 Unauthorized` - The request lacks valid authentication credentials for the requested resource
- `404 Not Found` - Indicates that the server cannot find the resource that was requested
- `429 Too Many Requests` - The client has sent too many requests in a given amount of time

`5xx` Server Errors

- `500 Internal Server Error` - The server experienced an unexpected issue which prevents it from responding
- `502 Bad Gateway` - The API server could not successfully reach another server it needed to complete the response
- `504 Gateway Timeout` - The server (which acts as a gateway) did not get a response from the upstream server in time

Handling errors

API errors

```
import requests

url = 'http://api.music-catalog.com/albums'

r = requests.get(url)
if r.status_code >= 400:
    # Oops, something went wrong
else:
    # All fine, let's do something
    # with the response
```

Connection errors

```
import requests
from requests.exceptions import ConnectionError
url = ''

try:
    r = requests.get(url)
    print(r.status_code)
except ConnectionError as conn_err:
    print(f'Connection Error! {conn_err}.')
    print(error)
```

raise_for_status()

```
import requests
# 1: Import the requests library exceptions
from requests.exceptions import ConnectionError, HTTPError

try:
    r = requests.get("http://api.music-catalog.com/albums")

    # 2: Enable raising exceptions for returned error statuscodes
    r.raise_for_status()

    print(r.status_code)

# 3: Catch any connection errors
except ConnectionError as conn_err:
    print(f'Connection Error! {conn_err}.')

# 4: Catch error responses from the API server
except HTTPError as http_err:
    print(f'HTTP error occurred: {http_err}')
```

Let's practice!

INTRODUCTION TO APIS IN PYTHON

Final thoughts

INTRODUCTION TO APIS IN PYTHON

Chris Ramakers
Engineering Manager

datacamp

API basics

- The role of APIs
- Different types of APIs
- URL components
- Anatomy of request & response messages
- HTTP Verbs

APIs with Python

Requests package

```
import requests
```

HTTP methods

```
# Read a resource
requests.get('https://api.my-music.com')
# Create a resource
requests.post('https://api.my-music.com', data={...})
# Update a resource
requests.put('https://api.my-music.com', data={...})
# Delete a resource
requests.delete('https://api.my-music.com')
```

URL Parameters

```
query_params = {'artist': 'Deep Purple'}
requests.get('http://api.my-music.com', params=query_params)
```

Headers

```
headers = {'accept': 'application/json'}
response = requests.get('http://api.my-music.com', headers=headers)
print(response.headers.get('content-type'))
```

Status codes

```
response = requests.get('http://api.my-music.com')
print(response.status_code)
```

Advanced topics

- **Authentication**
  - Basic Authentication

```
headers = {'Authorization':'Basic am9obkBleGF...'}
```

  - API key/token authentication

```
headers = {'Authorization': 'Bearer faaa1c9f4...'}
```

- **Structured data**
  - Requesting JSON formatted data

```
requests.get('https://api.my-music.com', headers={'accept': 'application/json'})
```

  - Sending JSON formatted data

```
playlists = [{"Name":"My favorite songs"}, {"Name":"Road Trip"}]
requests.post('https://api.my-music.com/playlists/', json=playlists)
```

Error handling

- Types of errors
  - Connection errors
  - HTTP errors
    - `4XX` Client errors
    - `5XX` Server errors
- Handling errors with status codes
  - `response.status_code`
- Handling errors with exceptions
  - `raise_for_error()`

```
import requests
from requests.exceptions import ConnectionError, HTTPError

try:
    response = requests.get("http://api.music-catalog.com/albums")
    response.raise_for_status()

except ConnectionError as conn_err:
    print(f'Connection Error! {conn_err}.')

except HTTPError as http_err:
    print(f'HTTP error occurred: {http_err}')
```

Congratulations!

INTRODUCTION TO APIS IN PYTHON