Working with APIs in Python

Generating a "Morning update" using APIs

2024-11-19

datacamp

Hello!

I'm Chris Ramakers

Chris Ramakers

Engineering Manager
Developer platforms & Design System

chrisramakers

Before we begin ...

Required accounts

This course makes use of several online service that offer APIs.

You'll need to have a registered account for each of the following services:

- newsapi.org
- meteosource.com
- platform.openai.com

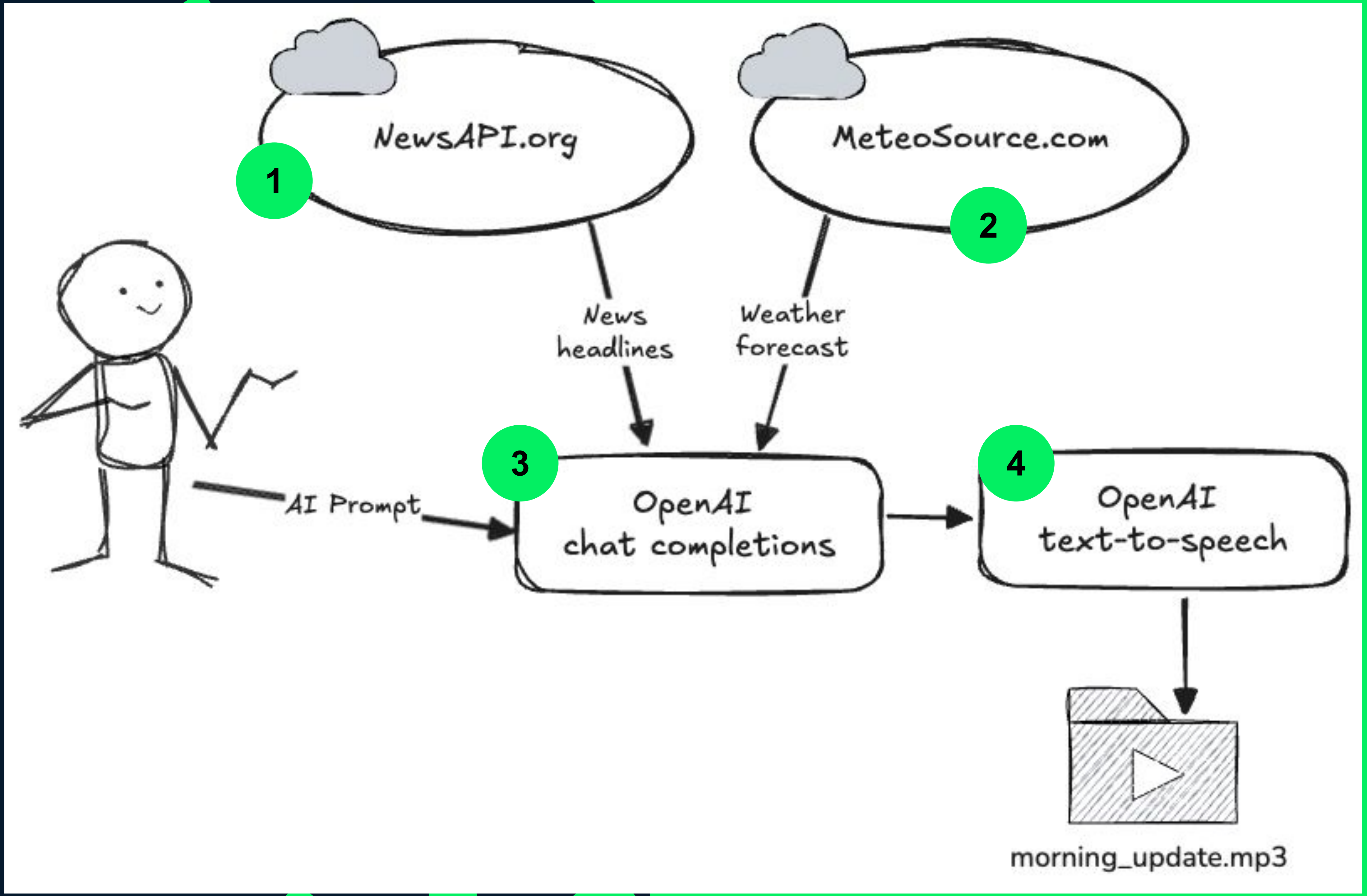
Make sure to register and confirm your email address for each of them!

On OpenAI credits

OpenAI platform is not free, you'll need to purchase credits in order to use the APIs and complete the code-along. You can find detailed instructions in the notebook!

Code-along

Generating a “Morning update” MP3 with APIs



Application Programming Interface

APIs allow different systems to communicate and interact with each other. They allow you to programmatically combine data or actions from different systems together in new and novel ways.

APIs are everywhere

Plug in your EV, APIs handle the data exchange between your car and the charger. Check the weather on your phone, data is retrieved from an online weather API and displayed on your device.

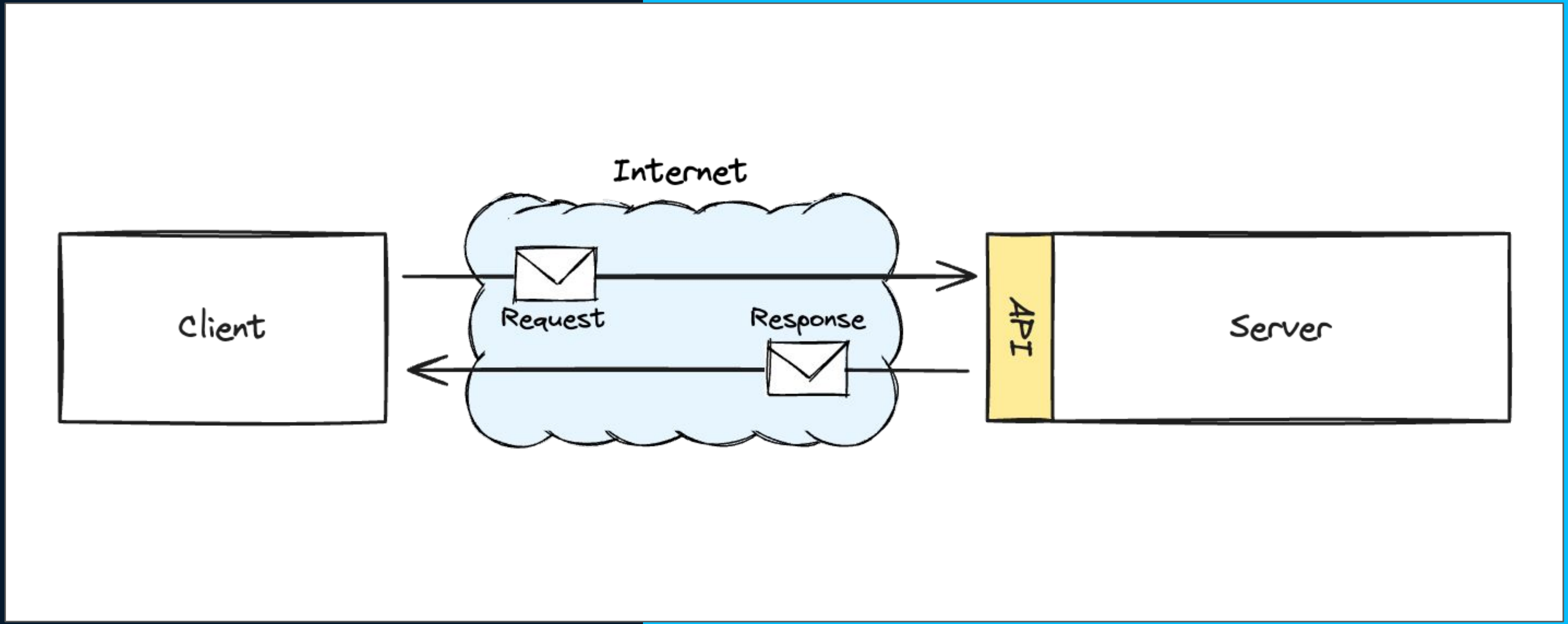
Defined set of rules

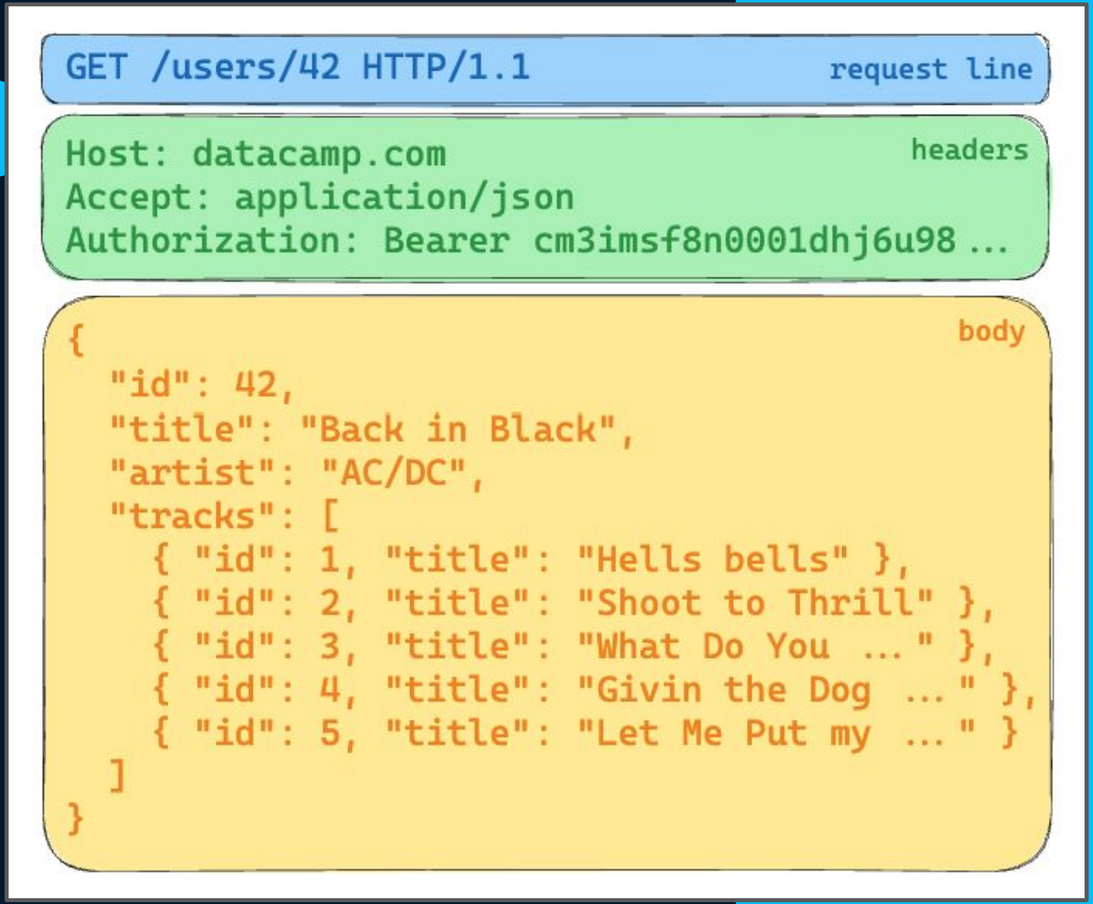
APIs define a set of rules how systems can interact with each other. These rules are made available as the *API documentation* and are a valuable resource in order to effectively integrate and use these APIs.

What is an API?

REST(-like) APIs

- Based on the HTTP protocol
- Client sends a *request message*
- Server returns a *response message*
- Integrating an API is …
  - … creating and sending a request
  - … and evaluating the response




request line
headers
body

```
GET /users/42 HTTP/1.1

Host: datacamp.com
Accept: application/json
Authorization: Bearer cm3imsf8n0001dhj6u98 ...

{
  "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 ... " }
  ]
}
```

An example request message

4 important concepts

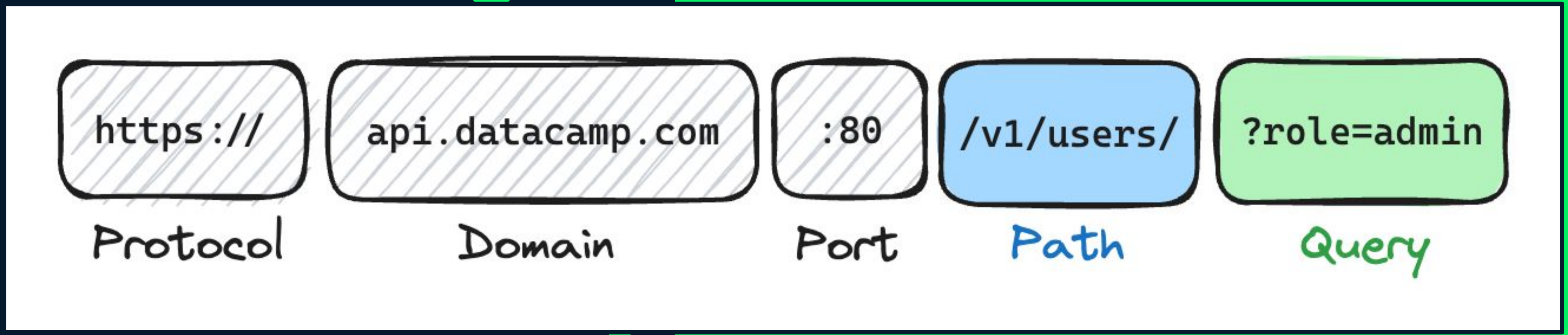
- URL & URL parameters
- HTTP Verbs
- Headers
- Status codes

Core concepts

Concept 1:

URL & URL Parameters

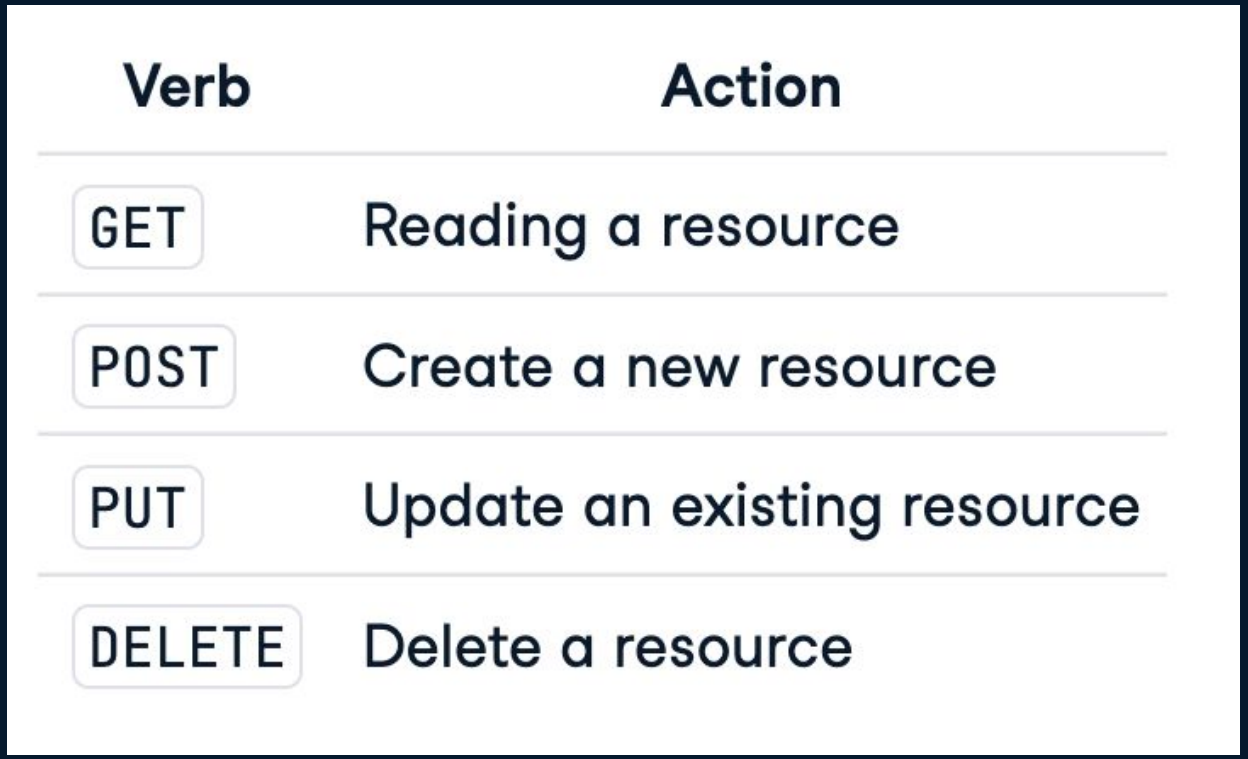
- Like an internet address
- Important elements
  - Domain
  - Path
  - Query (url parameters)
- Aka "query parameters"



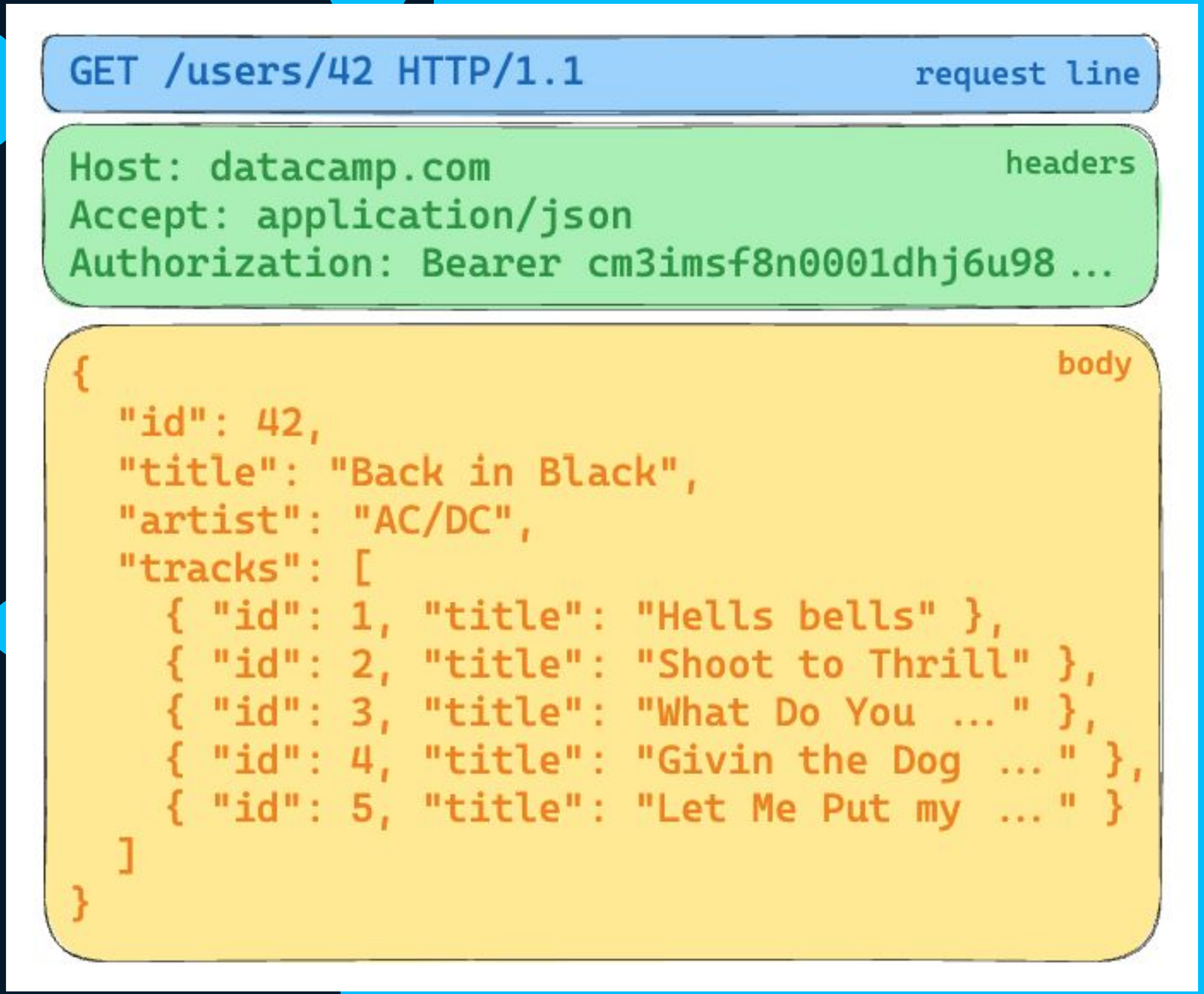
Concept 2:

HTTP Verbs

- Request message
- Describes the action to perform



Verb	Action
`GET`	Reading a resource
`POST`	Create a new resource
`PUT`	Update an existing resource
`DELETE`	Delete a resource



```
GET /users/42 HTTP/1.1
```

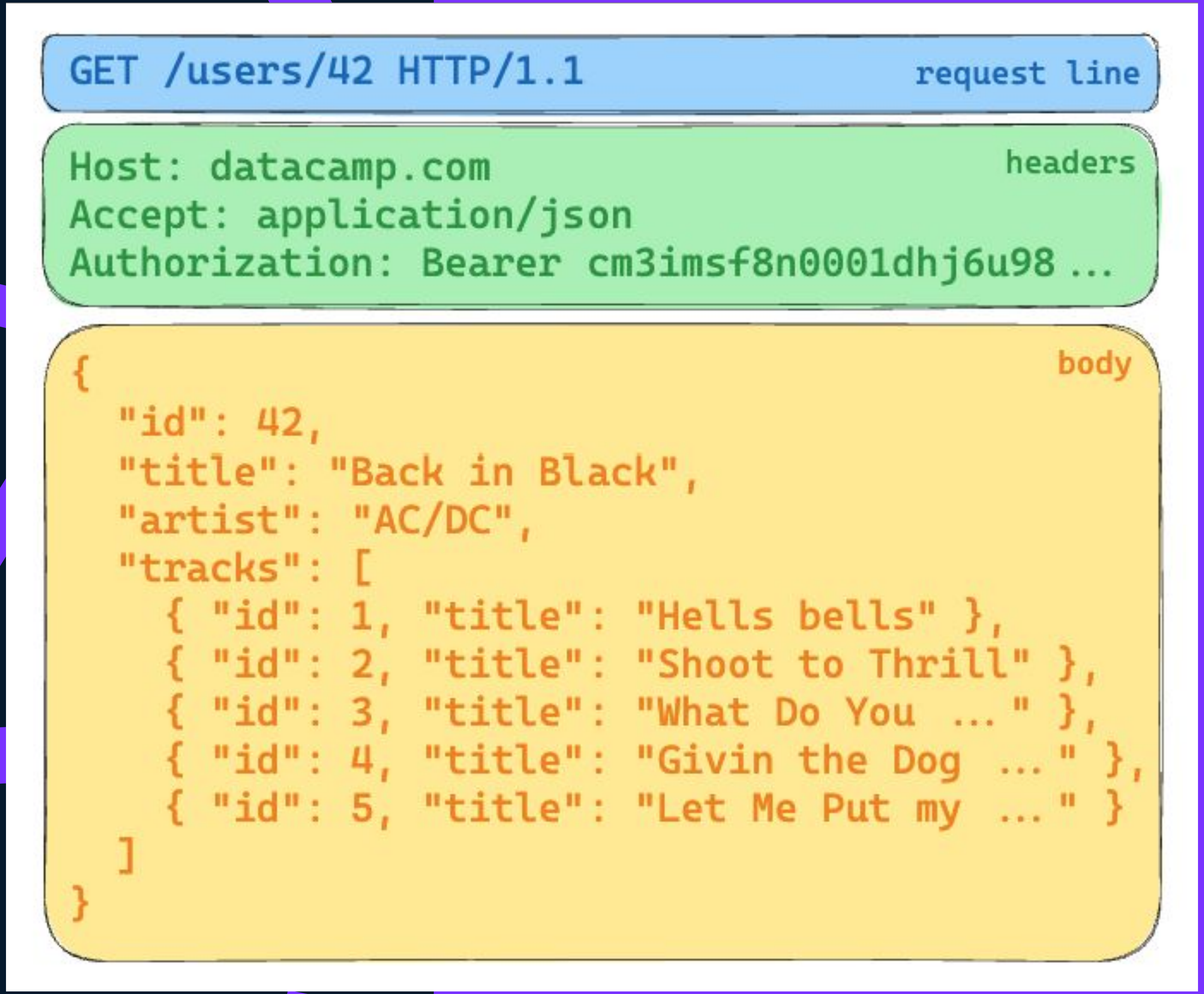
```
Host: datacamp.com
Accept: application/json
Authorization: Bearer cm3imsf8n0001dhj6u98 ...
```

```
{
  "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 ... " }
  ]
}
```

Concept 3:

Headers

- Request & response messages
- Context about the message
- Used for authorization, etc …



```
GET /users/42 HTTP/1.1
```

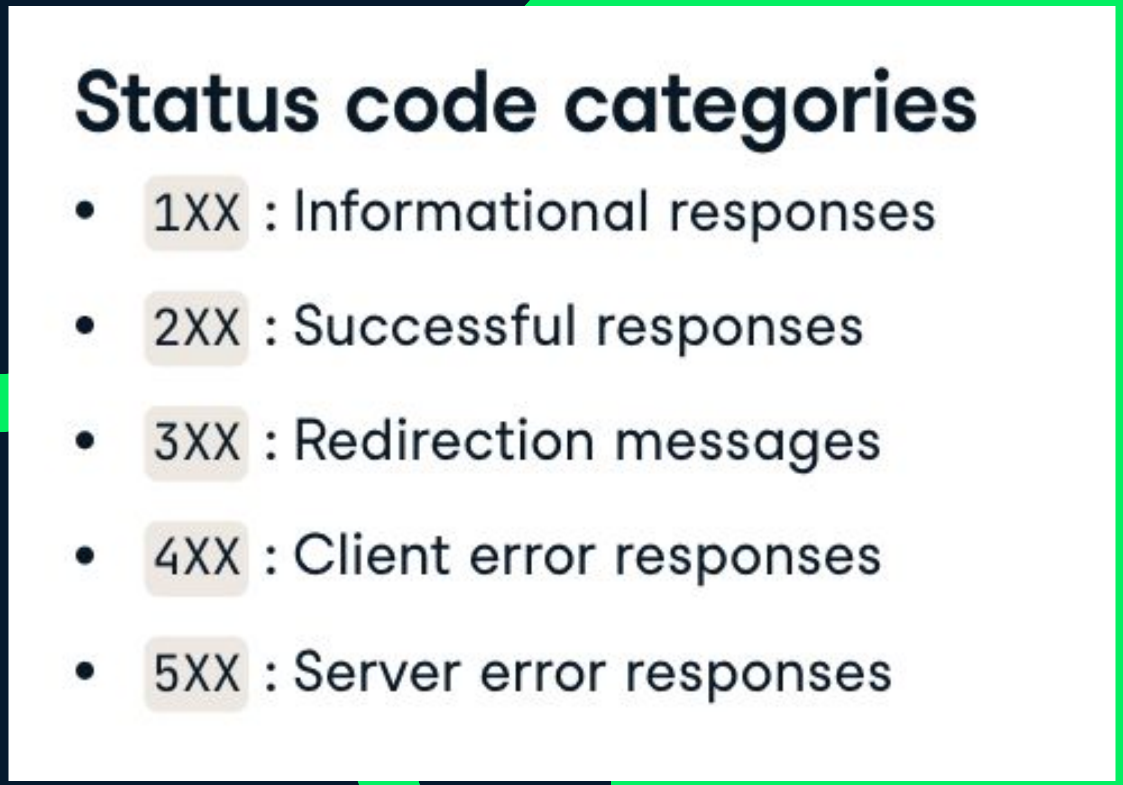
```
Host: datacamp.com
Accept: application/json
Authorization: Bearer cm3imsf8n0001dhj6u98 ...
```

```
{
  "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 ... " }
  ]
}
```

Concept 4:

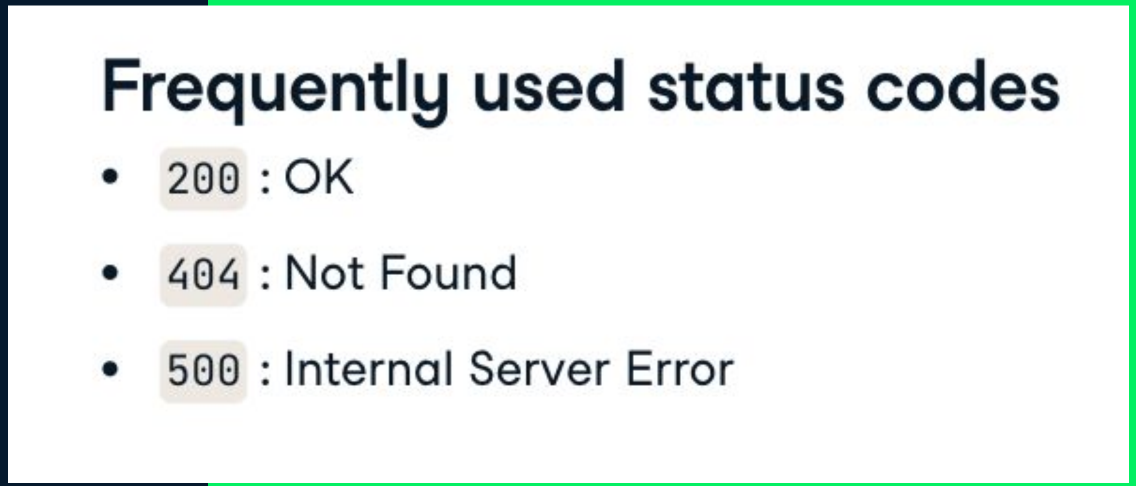
Status codes

- Part of the *response* message
- Indicate how the server has evaluated the message



Status code categories

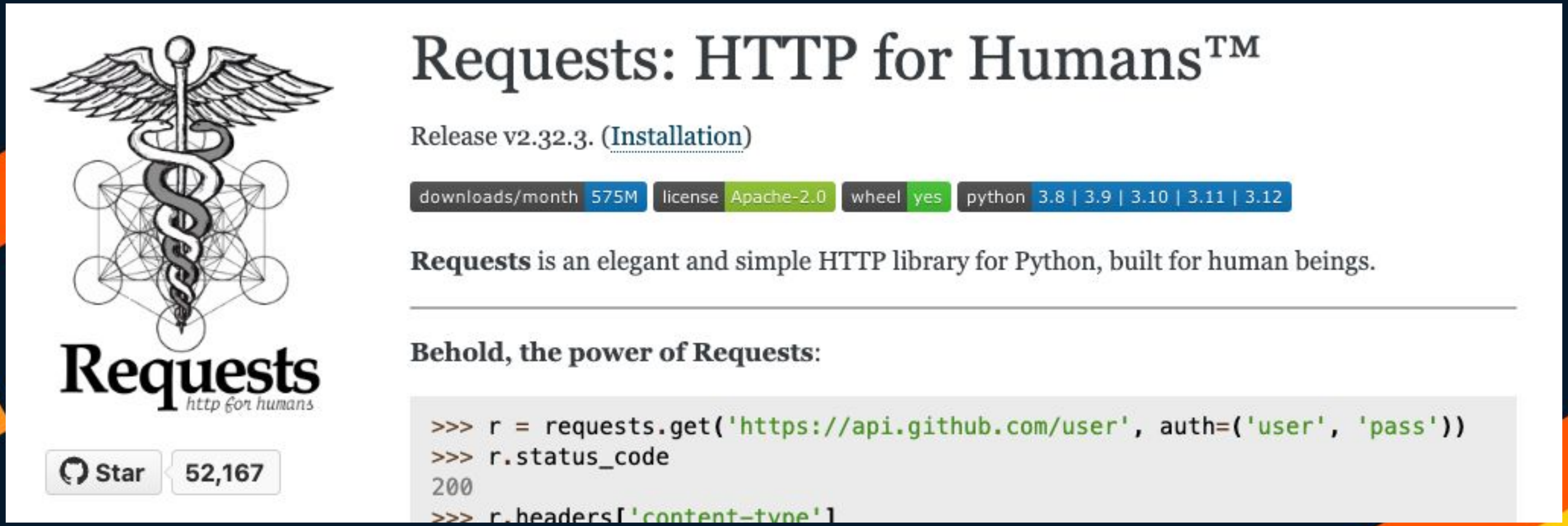
- `1XX` : Informational responses
- `2XX` : Successful responses
- `3XX` : Redirection messages
- `4XX` : Client error responses
- `5XX` : Server error responses



Frequently used status codes

- `200` : OK
- `404` : Not Found
- `500` : Internal Server Error

🤯 But, that all seems so complex



Requests: HTTP for Humans™

Release v2.32.3. (Installation)

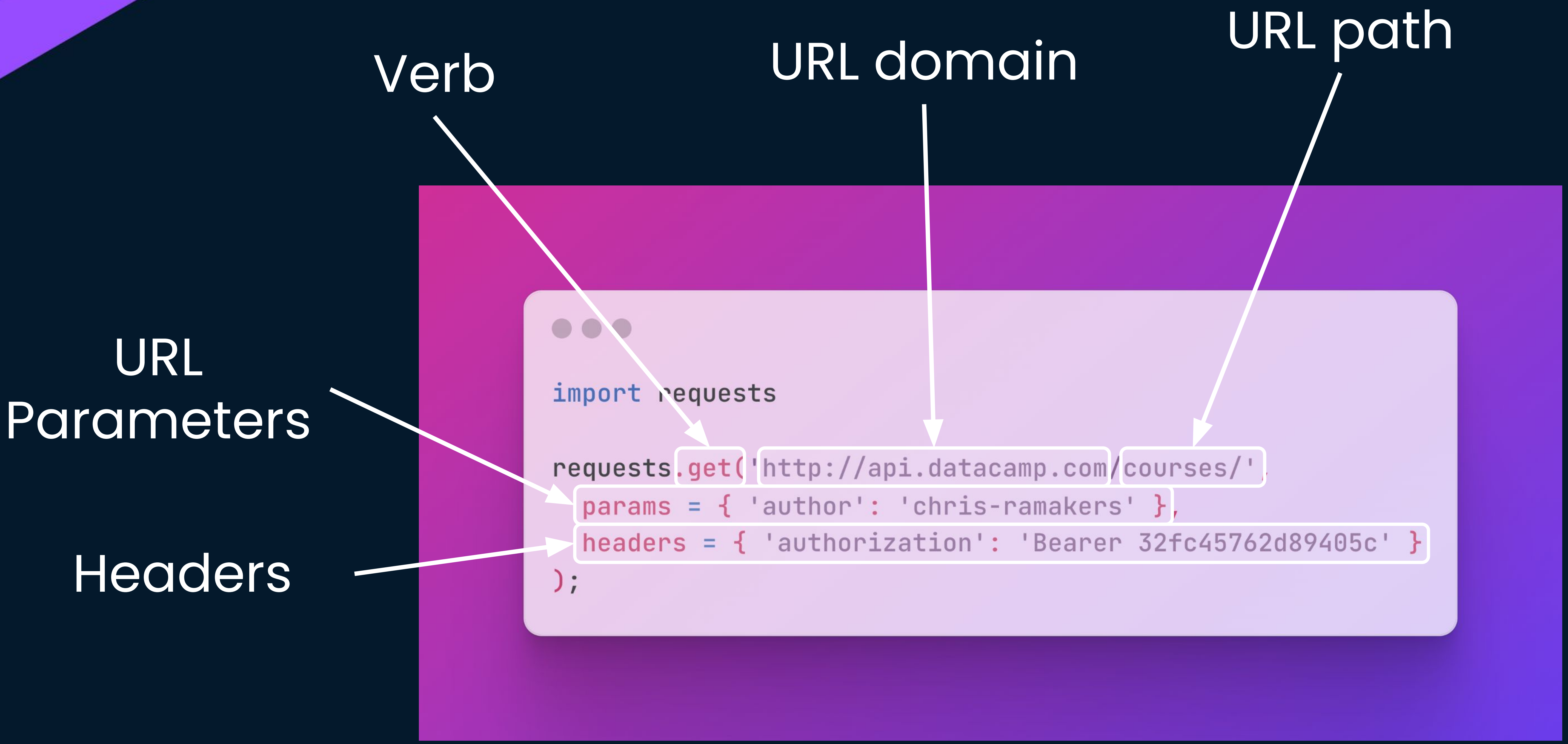
downloads/month 575M license Apache-2.0 wheel yes python 3.8 | 3.9 | 3.10 | 3.11 | 3.12

Requests is an elegant and simple HTTP library for Python, built for human beings.

Behold, the power of Requests:

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
```

HTTP requests with requests™



```
import requests

requests.get('http://api.datacamp.com/courses/',
 params = { 'author': 'chris-ramakers' },
 headers = { 'authorization': 'Bearer 32fc45762d89405c' }
);
```

Let’s get coding!

clear as data

Thank you

Chris Ramakers
Engineering Manager @ DataCamp
chris.ramakers@datacamp.com