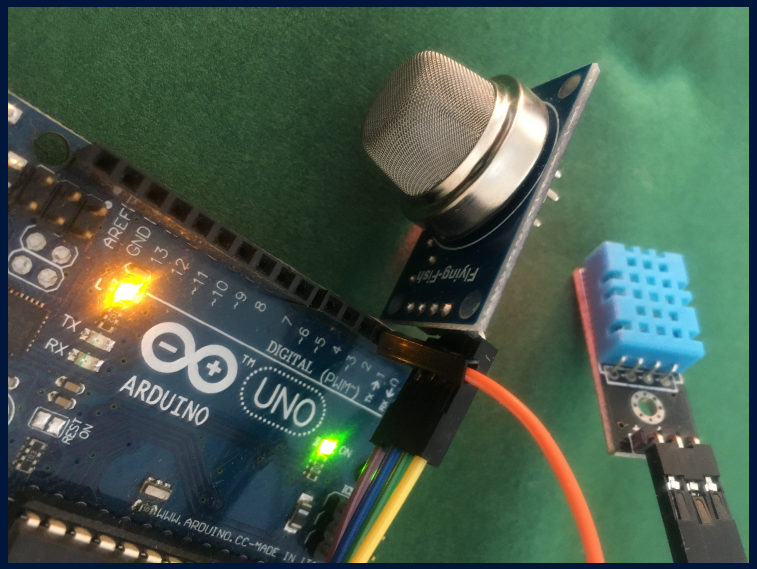
cphbusiness

COPENHAGEN BUSINESS ACADEMY

DATA ENGINEERING



cphbusiness

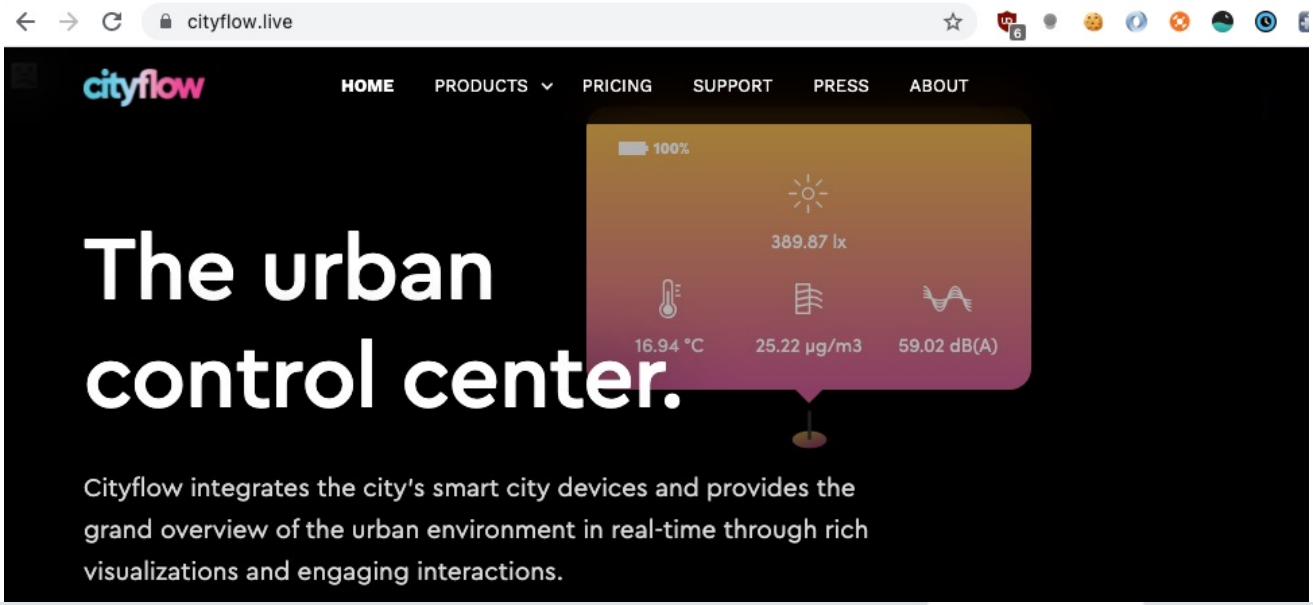
FLOW 3 – Data Engineering

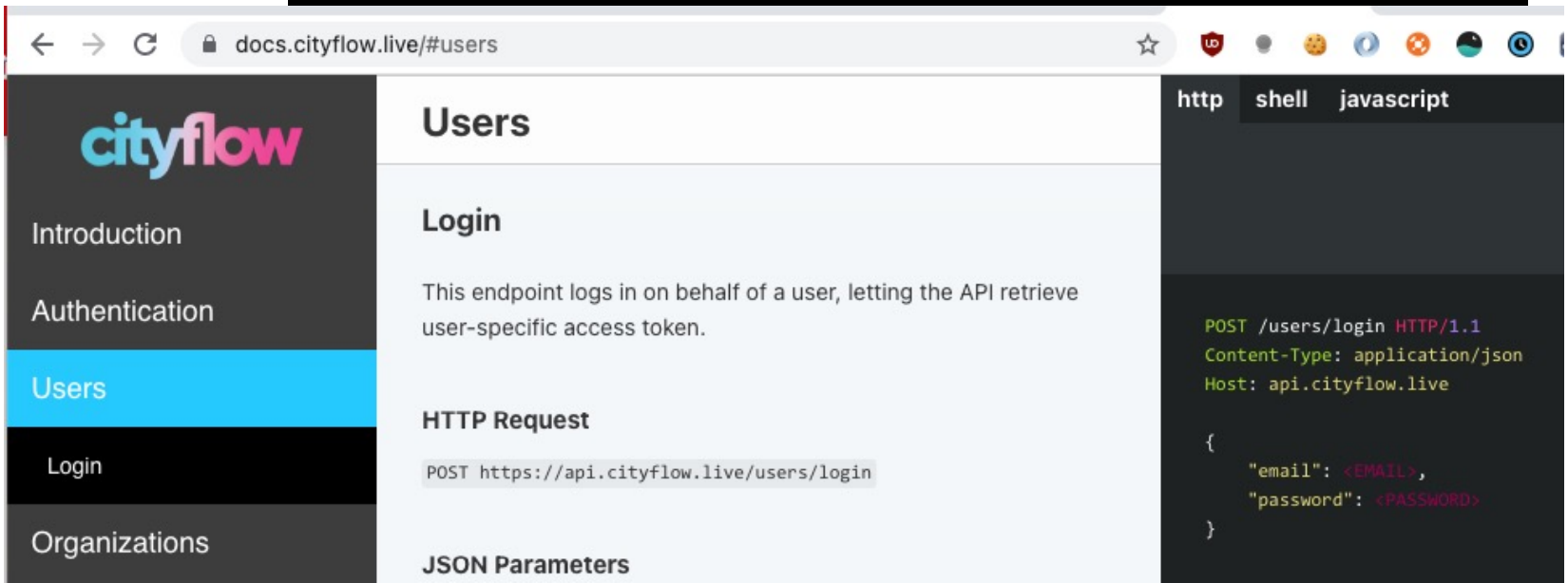
- Oplæg om de to cases i Flow 3
  - cityflow.dk – API
    - Json-dataformat
    - https request
    - python request
    - File-persistence (json, csv)
    - Database-persistence (MongoDB)
  - Bilbasen.dk – WebScraping
    - HTML/CSS/JavaScript primer
    - Scrape static (python bs4)
    - Scrape dynamic (python selenium)
- Tooling
  - GitHub / GitBash
  - Python (Anaconda) & jupyter notebook
  - AWS EC2-instance (Ubuntu med MongoDB og MySQL)
  - Arduino with serial-port driver

FLOW 3 – User Stories

- Som underviser vil jeg gerne vide hvor meget den hvide bygning bliver brugt så jeg kan lægge aktiviteter i bygningen når den bliver mindst brugt
- Som studerende vil jeg gerne vide hvor det er bedst at lave en walk-and-talk i Århus i løbet af dagen på en hverdag
- Som ejer af en gammel Volvo vil jeg gerne kunne få en vurdering af hvor meget den er værd således at jeg kan planlægge hvornår jeg skal sælge den

Cityflow


cityflow.live
cityflow
HOME
PRODUCTS
PRICING
SUPPORT
PRESS
ABOUT
100%
389.87 lx
16.94 °C
25.22 µg/m3
59.02 dB(A)
The urban control center.
Cityflow integrates the city's smart city devices and provides the grand overview of the urban environment in real-time through rich visualizations and engaging interactions.


docs.cityflow.live/#users
cityflow
Introduction
Authentication
Users
Login
Organizations
Users
Login
This endpoint logs in on behalf of a user, letting the API retrieve user-specific access token.
HTTP Request
POST https://api.cityflow.live/users/login
JSON Parameters
http
shell
javascript
POST /users/login HTTP/1.1
Content-Type: application/json
Host: api.cityflow.live
{
"email": <EMAIL>,
"password": <PASSWORD>
}

cphbusiness

Cityflow – REST API

- What do I see at docs.cityflow.live?

- Authentication
- Devices
- Measurements
- Errors

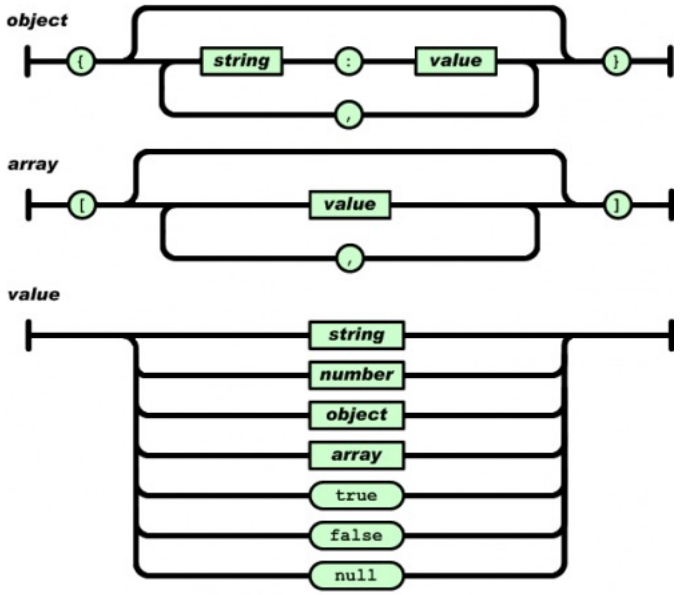
- http
- shell
- javascript

Cityflow – REST API Data Format

- ”The above command returns JSON structured like this”
- JSON – JavaScript Object Notation

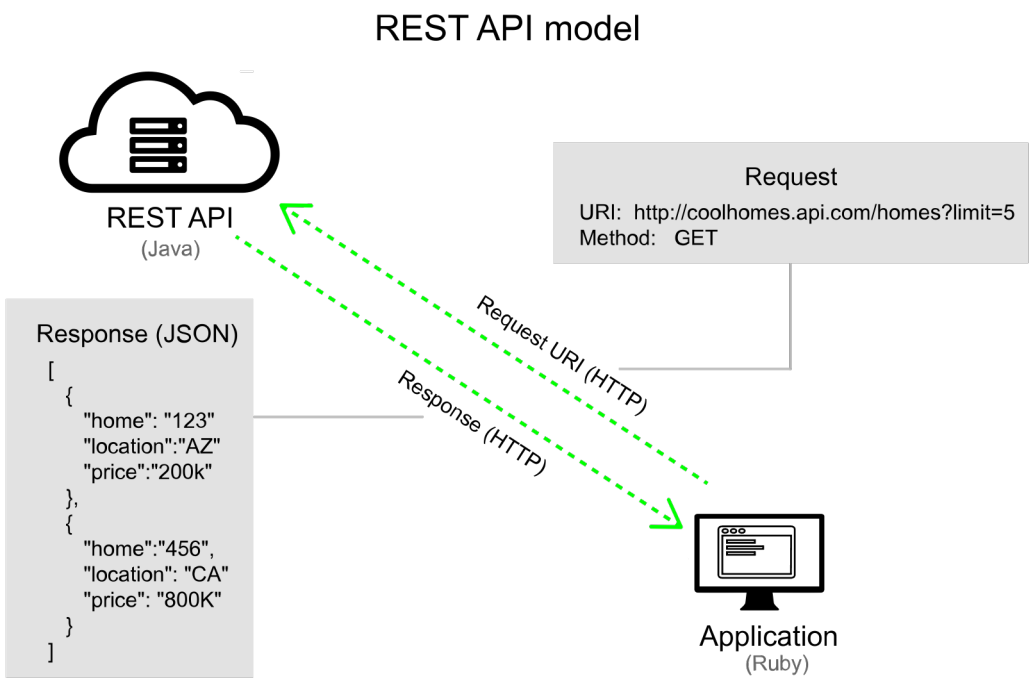
JSONLint - The JSON Validator


```
{
  "id": "e00fce68c573b4acca2089ce",
  "type": 150,
  "location": 216,
  "latitude": 56.1632767,
  "longitude": 10.2105122,
  "location_name": "Nørrebrogade",
  "city": "Aarhus",
  "country": "Denmark",
  "roles": [
      4
  ],
  "permissions": [],
  "tags": [
      "Randersvej"
  ]
}
```

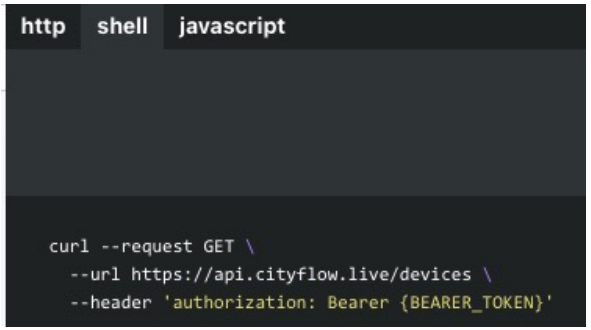



cphbusiness

Cityflow – REST API Request



- Http – hvad er det?
  - Chrome dev-tools
- Shell – hvad er det?
  - Intall Git Bash (windows)
  - Curl – hvad er det?



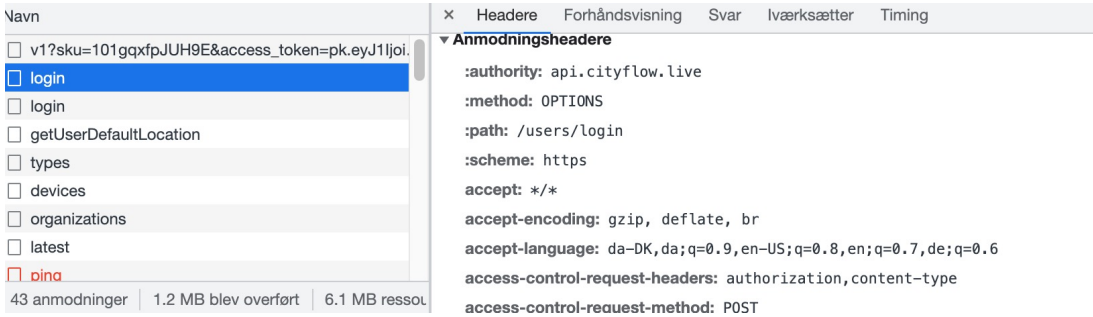
cphbusiness

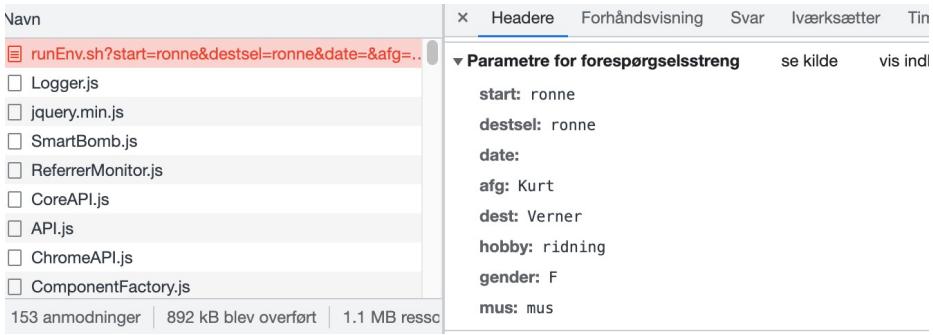
Cityflow – REST API Request

1. Introduction
2. Protocols
3. Data Formats
4. Authentication, Part 1
5. Authentication, Part 2
6. API Design
7. Real-Time Communication
8. Implementation

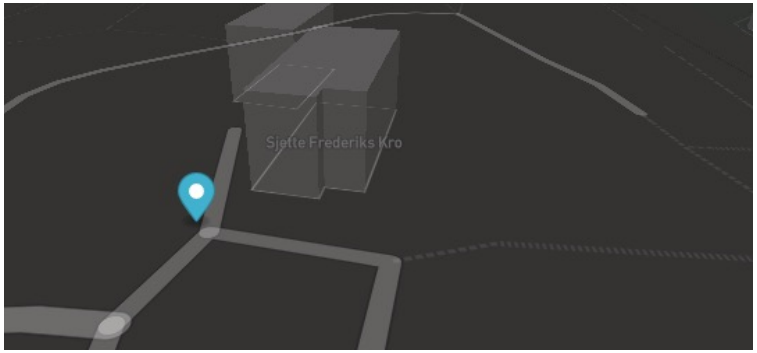
To make a valid request, the client needs to include four things:

1 **URL** (Uniform Resource Locator) [1]
2 **Method**
3 List of **Headers**
4 **Body**





Cityflow – REST API Get Data I



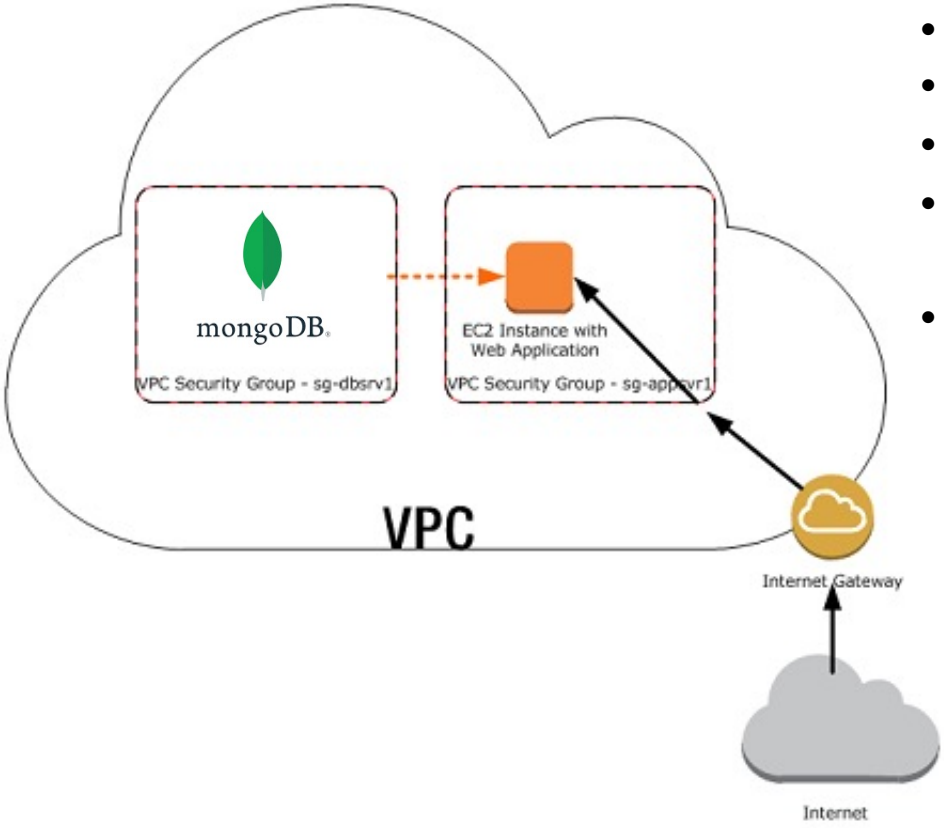
http shell javascript

```
GET /devices HTTP/1.1
Authorization: Bearer {BEARER_TOKEN}
Host: api.cityflow.live
```


```
{
  id: e00fce689f02a96799f34fc2,
  type: 150,
  location: 190,
  latitude: 56.1770897,
  longitude: 10.2296247,
  location_name: Salonvejen,
  city: Risskov,
  country: Denmark,
  roles: [
    4
  ],
  permissions: [],
  tags: [
    Risskov
  ]
},
```

cphbusiness

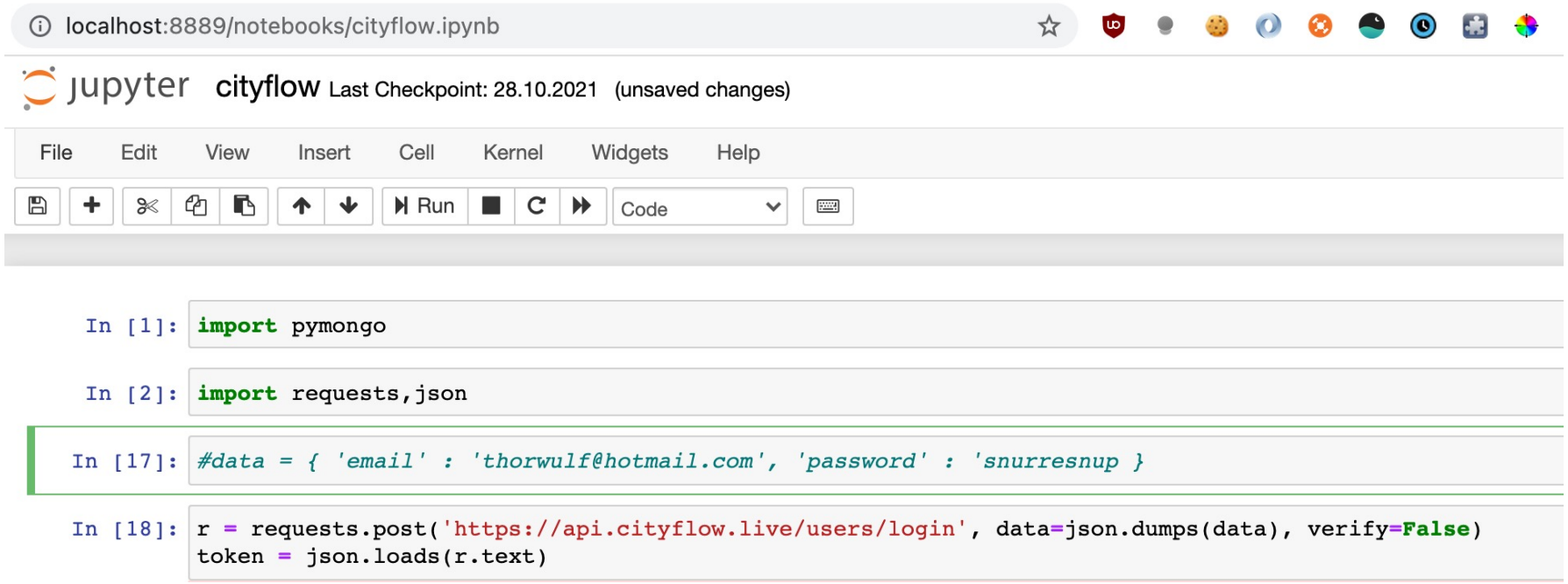
Cityflow – Save data Infrastructure



- Create EC2 instance (ubuntu)
- Get ssh-access
- Install mongodb
- Client access
  - Robo 3T
- Dev access
  - pymongo

cphbusiness

Cityflow – python from jupyter notebook



<class 'requests.models.Response'>

cphbusiness

Cityflow – opgave: udfyld skabelonen

importer de nødvendige moduler

In [ ]:

lav en post-request til https://api.cityflow.live/users/login

In [ ]:

lav en get-request der henter listen af devices

In [ ]:

lav en get-request der henter de nyeste målinger https://api.cityflow.live/measurements/latest Du gemmer resultatet af din request i en variabel som så er af type response-objekt.

In [ ]:

Du henter nu en datastruktur ud af response-objektet (response.json()) Undersøg strukturen af de returnerede data så du kan nå ind til listen af målinger Du får nemlig noget a la { '150': [{ 'aPS': 0.69, 'b': 100,

In [ ]:

Nu kan du hente pretty-print så dit resultat står pænt

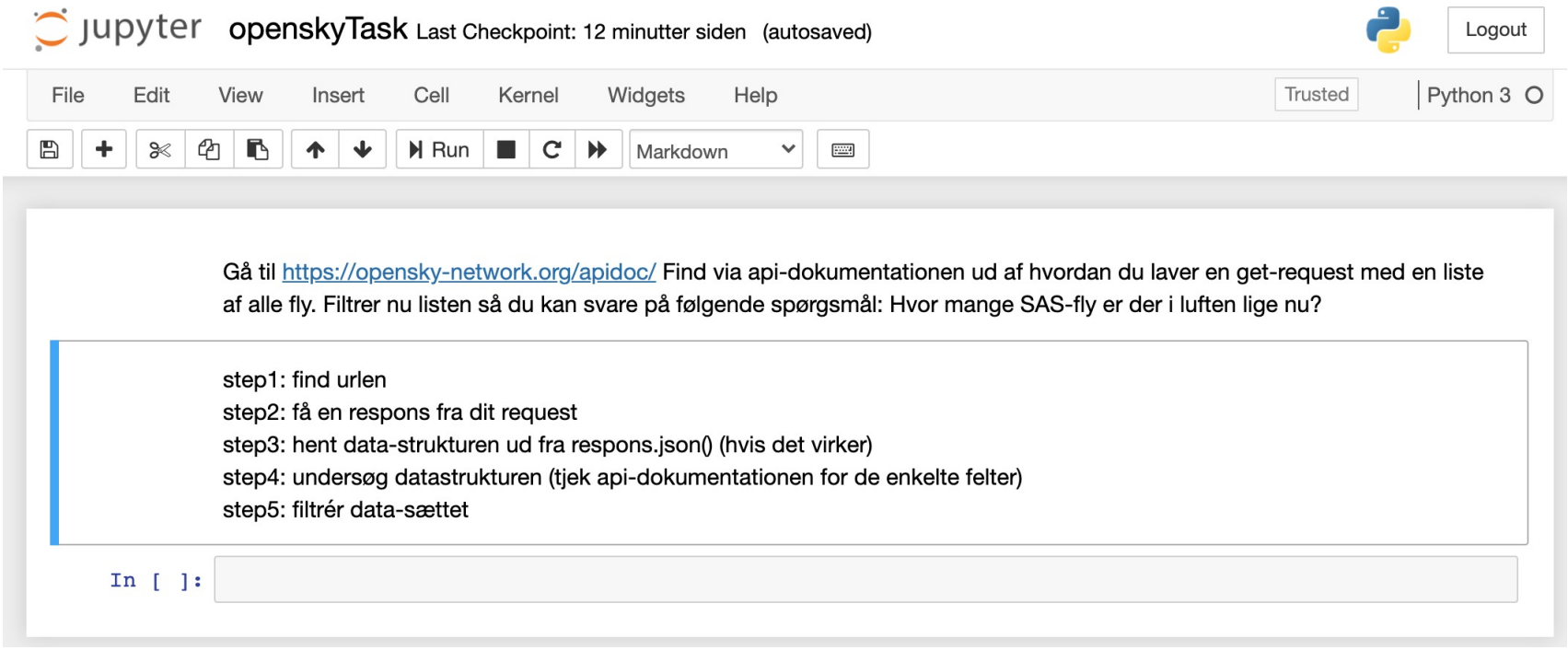
In [ ]:
```
import pprint as pp
dineMeasures=[]
pp.pprint(dineMeasures)
```

Nu looper du igennem din liste af målinger vha et for-loop

In [ ]:
```
for item in dineMeasures:
    pp.pprint(item)
```

cphbusiness

Cityflow – opgave: Antal SAS-fly i luften



cphbusiness

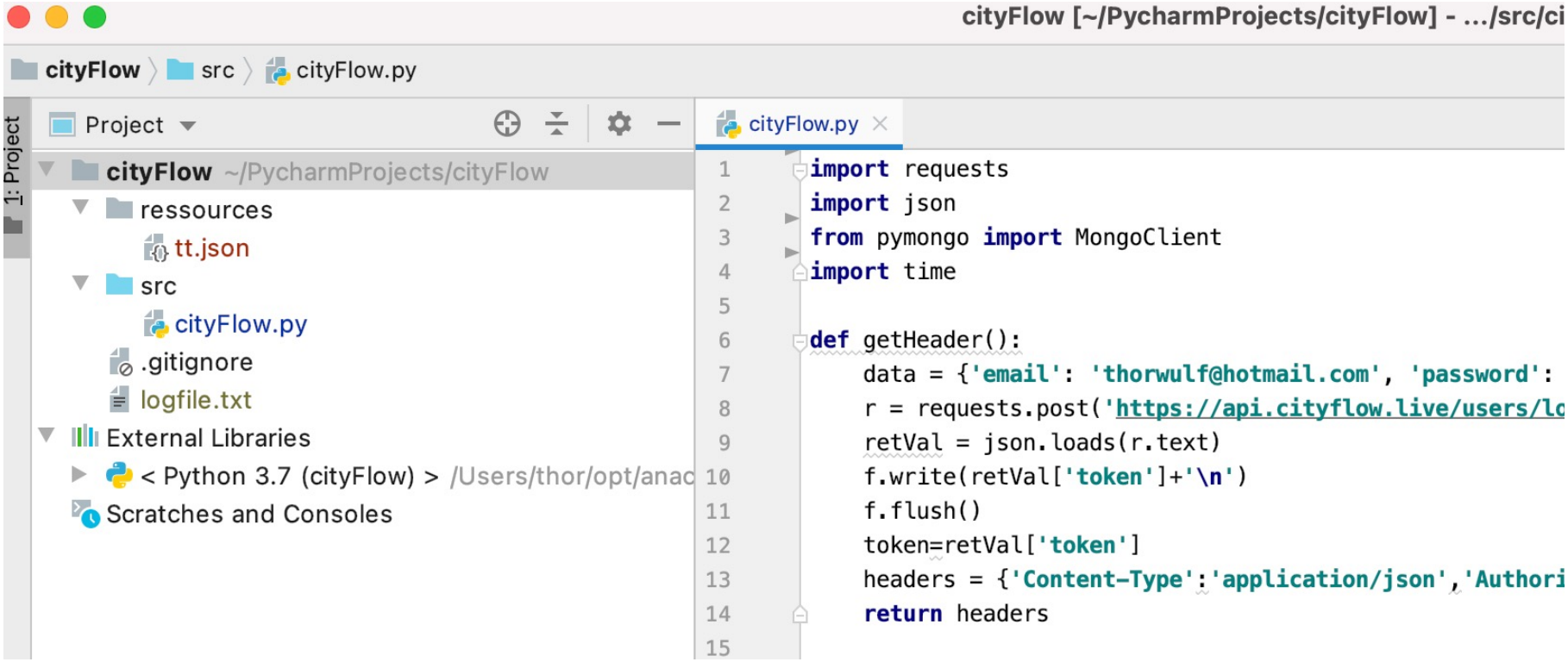
Cityflow – python from jupyter notebook

<class 'requests.models.Response'>

Properties and Methods

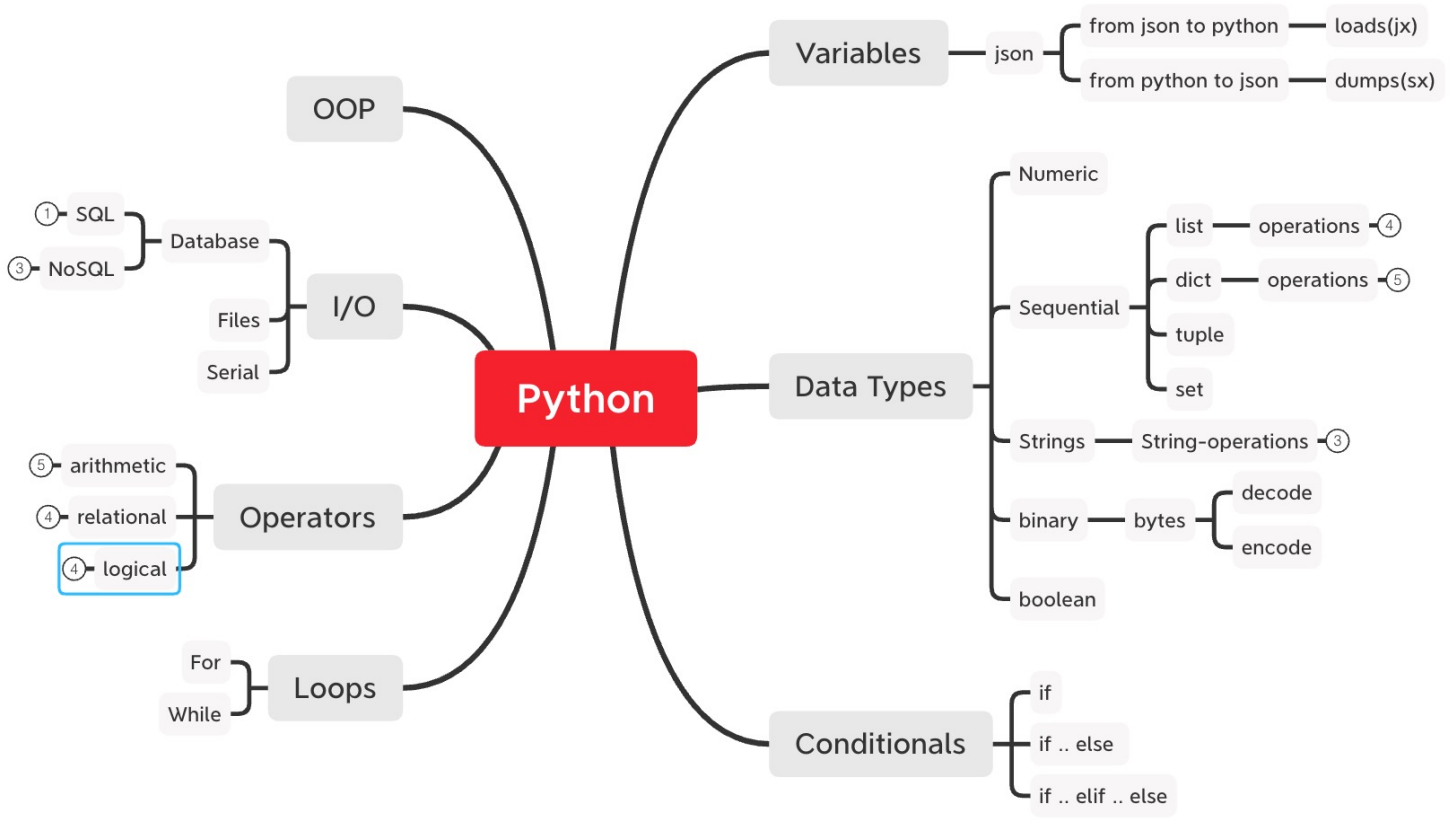
Property/Method		Description
apparent_encoding	Try it	Returns the apparent encoding
close()	Try it	Closes the connection to the server
content	Try it	Returns the content of the response, in bytes
cookies	Try it	Returns a CookieJar object with the cookies sent back from the server
elapsed	Try it	Returns a timedelta object with the time elapsed from sending the request to the arrival of the response
encoding	Try it	Returns the encoding used to decode r.text
headers	Try it	Returns a dictionary of response headers
history	Try it	Returns a list of response objects holding the history of request (url)
is_permanent_redirect	Try it	Returns True if the response is the permanent redirected url, otherwise False
is_redirect	Try it	Returns True if the response was redirected, otherwise False
iter_content()	Try it	Iterates over the response
iter_lines()	Try it	Iterates over the lines of the response
json()	Try it	Returns a JSON object of the result (if the result was written in JSON format, if not it raises an error)
links	Try it	Returns the header links
next	Try it	Returns a PreparedRequest object for the next request in a redirection
ok	Try it	Returns True if status_code is less than 400, otherwise False
raise_for_status()	Try it	If an error occur, this method returns a HTTPError object
reason	Try it	Returns a text corresponding to the status code
request	Try it	Returns the request object that requested this response
status_code	Try it	Returns a number that indicates the status (200 is OK, 404 is Not Found)
text	Try it	Returns the content of the response, in unicode
url	Try it	Returns the URL of the response

Cityflow – CRUD from python IDE



cphbusiness

Cityflow – python primer



cphbusiness

Cityflow – PyMongo CRUD primer

C	• insert()
R	• find()
U	• update()
D	• remove()

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]

mydict = { "name": "John", "address": "Highway 37" }

x = mycol.insert_one(mydict)
```

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]

for x in mycol.find({},{ "address": 0 }):
  print(x)
```

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]

for x in mycol.find():
  print(x)
```

```
import pymongo

myclient = pymongo.MongoClient("mongodb://
mydb = myclient["mydatabase"]
mycol = mydb["customers"]

myquery = { "address": { "$gt": "S" } }

mydoc = mycol.find(myquery)

for x in mydoc:
  print(x)
```

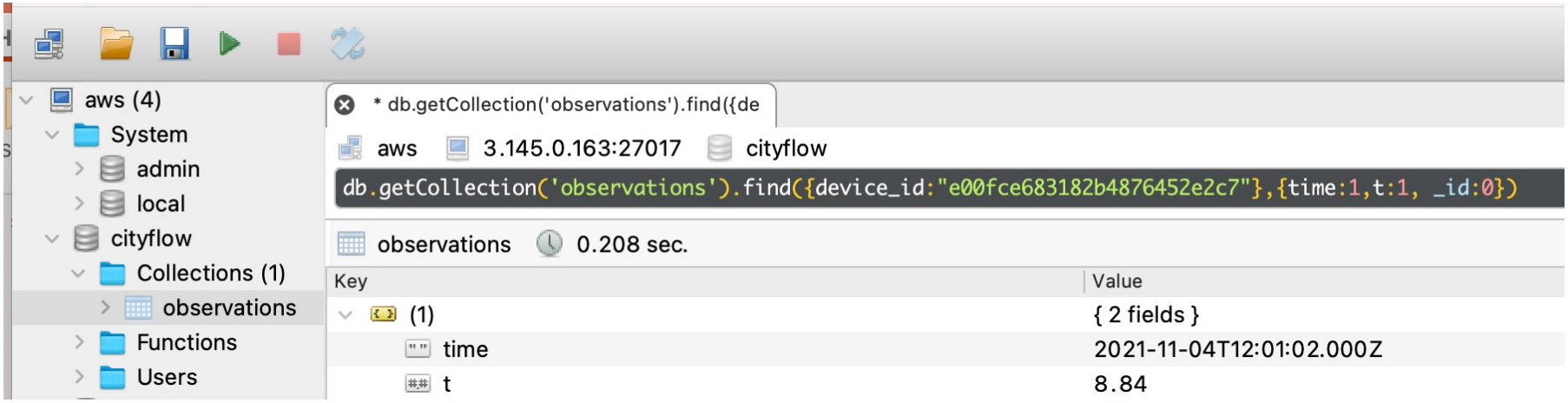
cphbusiness

Cityflow – mongodb CRUD primer

C	• insert()
R	• find()
U	• update()
D	• remove()

cphbusiness

Cityflow – inspect data from Mongo

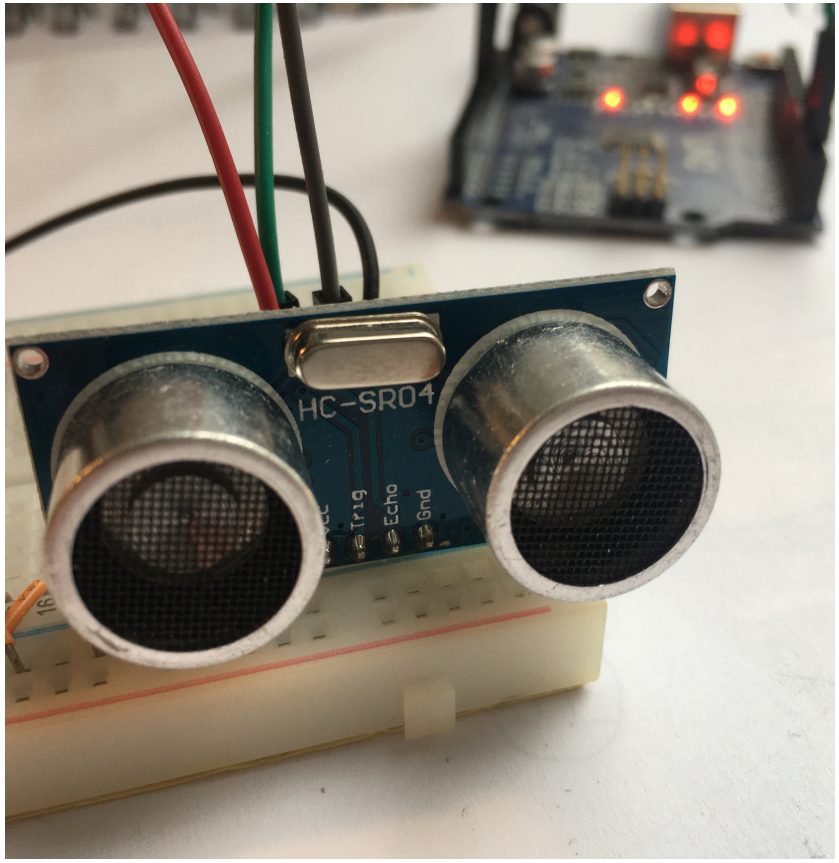


Cityflow – CRUD-operations

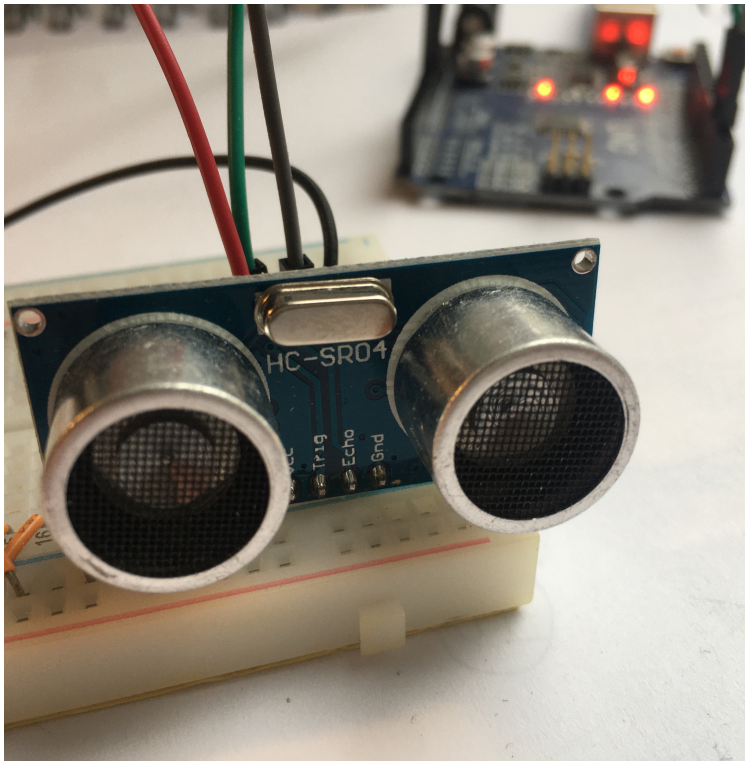
- C • insert()
- R • find()
- U • update()
- D • remove()

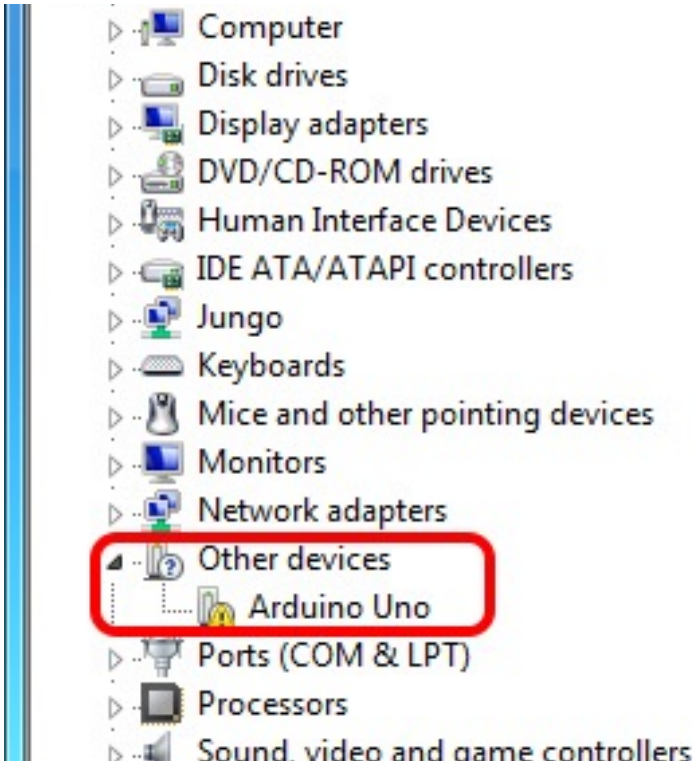
Den hvide bygning – Data-retrieval

- Design af IO-device
- Kodning af device
  - Arduino
  - Python
- Analyse af data
- Design et API til forespørgslen: hvor mange er i bygningen <yyyy-mm-dd hh:mm>

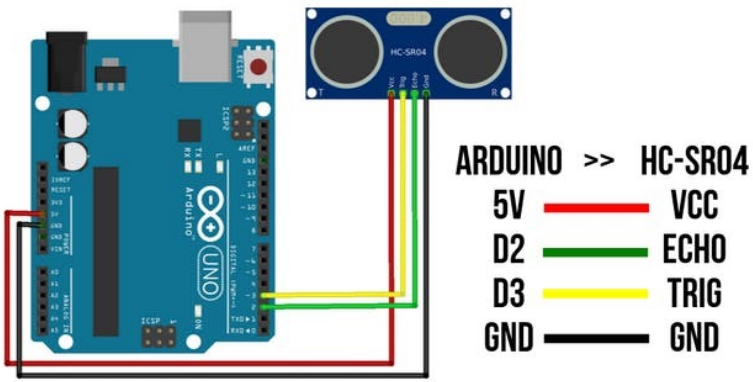


Den hvide bygning – Data-retrieval via USB





HC-SR04



- Installér Arduino
- Plug boardet
- Tjek at din device dukker op i enhedshåndteringen

cphbusiness

HC-SR04 - PySerial