Introduction to APIs

Module #:
APIs

APIs

The topic of this module

How do we retrieve data from a third party, such as Kiva, Twitter, or Google?

These services offer APIs (Application Programmable Interface), or exchanges/contracts between programs.

APIs specify the way you should communicate with the third party to retrieve data you are interested in.

Then, after you've learned how to retrieve data, what are ethical concerns in accessing data?

Module Checklist:

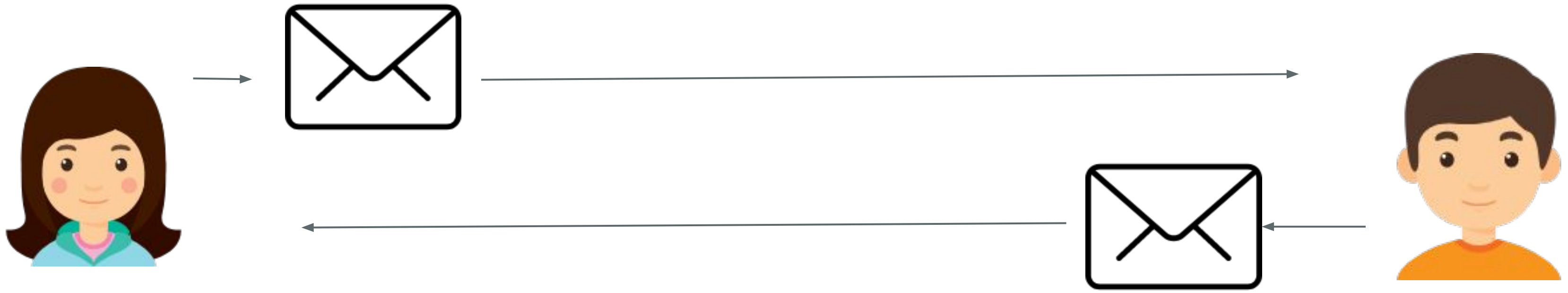
- ❑ Interfaces
- ❑ HTTP
- ❑ Data Formats (JSON)
- ❑ Consuming APIs in Python
- ❑ Data access ethics

HTTP

How do these terms relate to each other?

To better understand the terms from the previous slide, consider an analogy:

You want your mail to get delivered to person x and for person x to send you back a response by mail.



HTTP

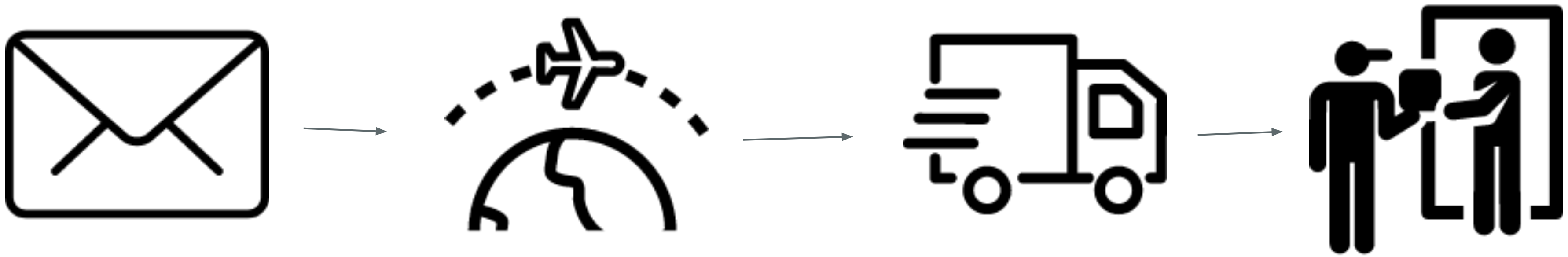
How do these terms relate to each other?

- The **API or interface** provides a contract between you and the mail delivery system, and you and person x
  - If you provide an envelope with an address, attach a stamp, and give it to a Postal Delivery worker, it will reach the destination.
  - You and person x have an agreed upon way to communicate beforehand.

HTTP

How do these terms relate to each other?

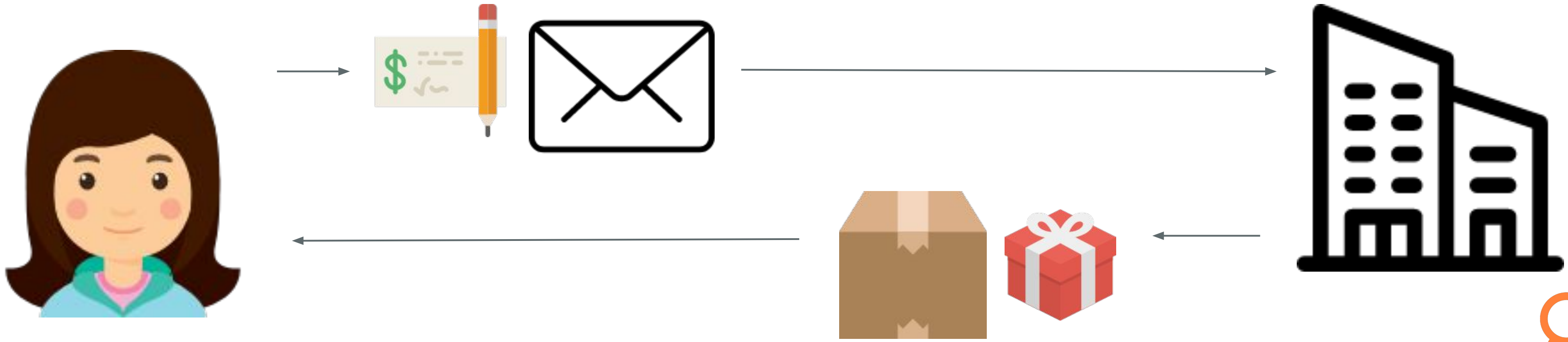
- **HTTP** is the means by which it is delivered. Your mail may be delivered by plane, by car, by foot -- it doesn't matter to you, as long as it gets to its destination.



HTTP

How do these terms relate to each other?

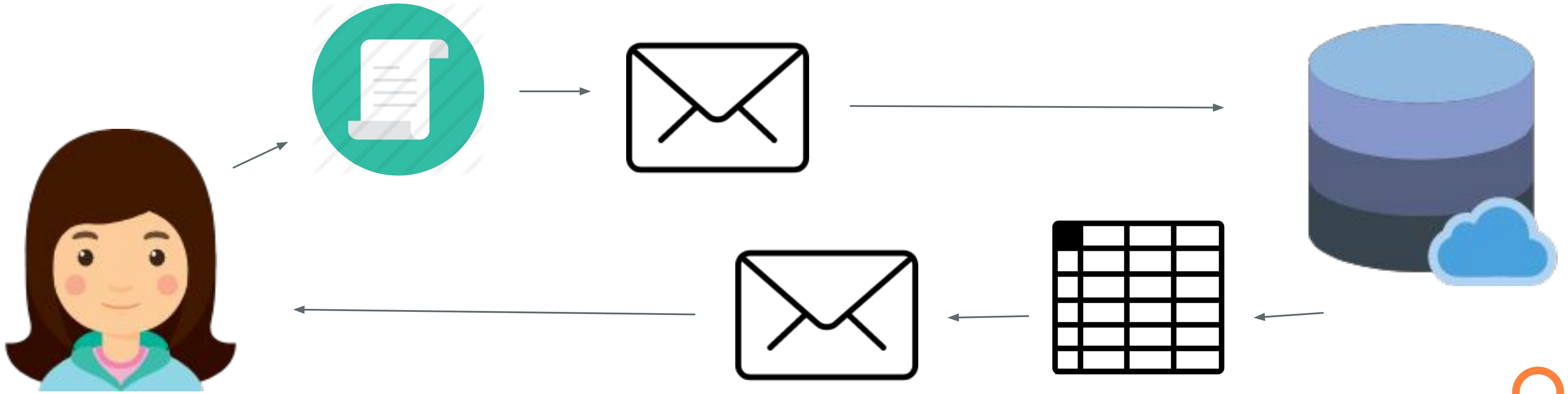
- The **data format** is an agreed upon way to enclose the message between you and person X.
    - If person X is a person at a company who will mail you a product, your data format might be an order form with a check for payment, and in response, person x will send you back a product wrapped in a cardbox box.



HTTP

How do these terms relate to each other?

- To bring it all together:
  You can use **Python** to send **HTTP** messages to a third-party service that complies with their **API**, who will then **format the data**, and send it to you.



Interfaces

Interfaces

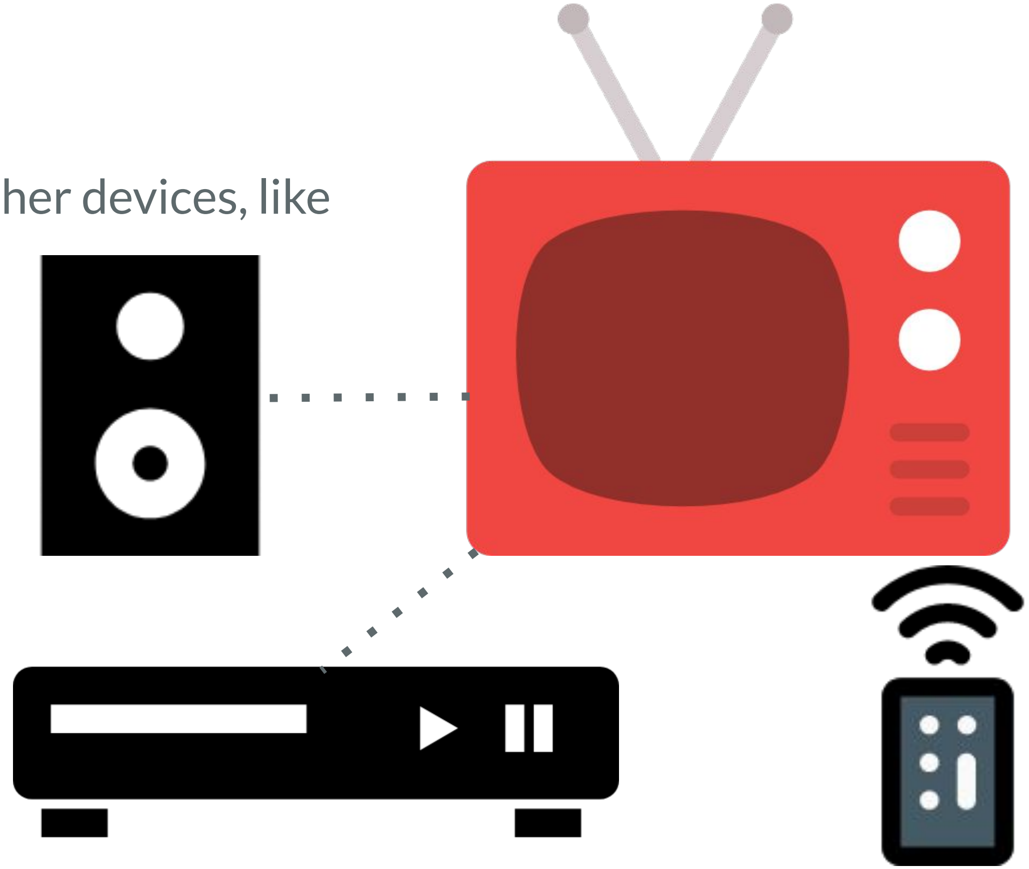
Television analogy

A TV might be connected to other devices, like

A dvd player

A remote control

Speakers

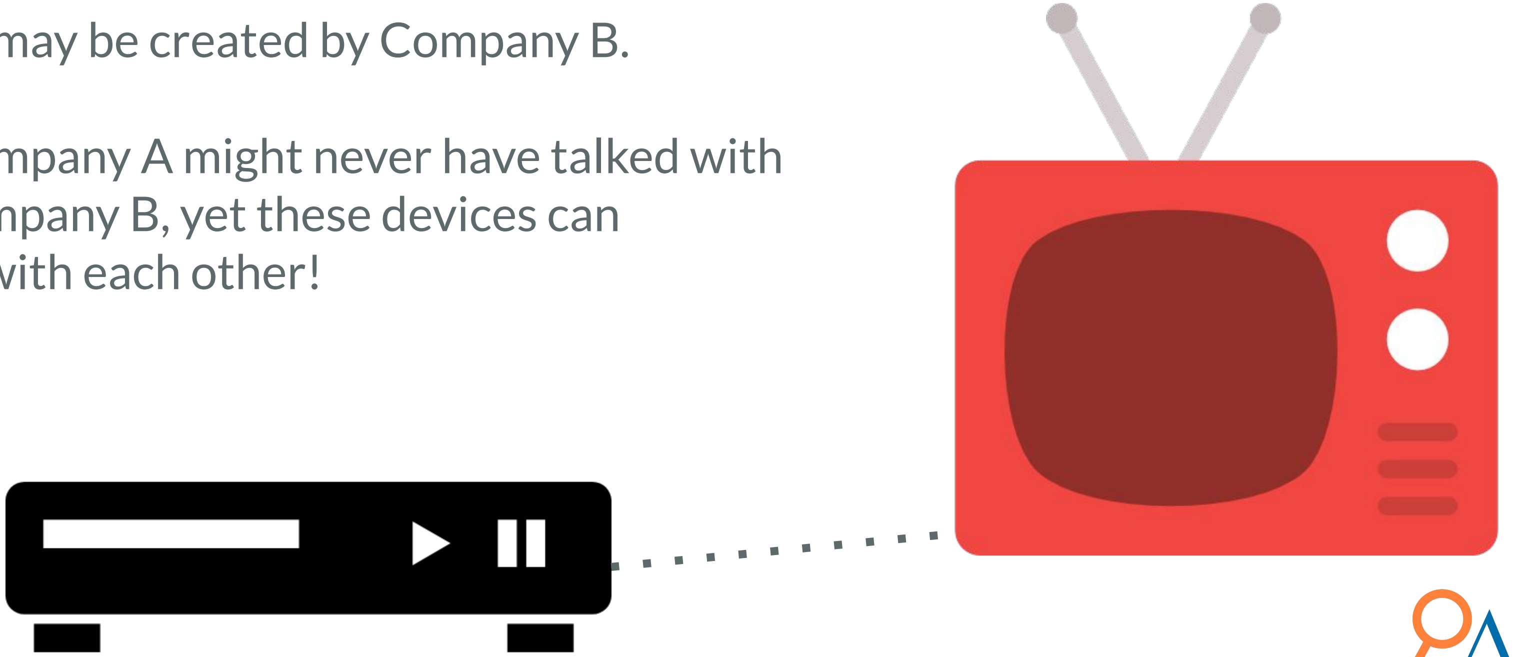


Television analogy

A DVD player may be created by Company A.

Whereas a TV may be created by Company B.

Workers at Company A might never have talked with workers at Company B, yet these devices can communicate with each other!

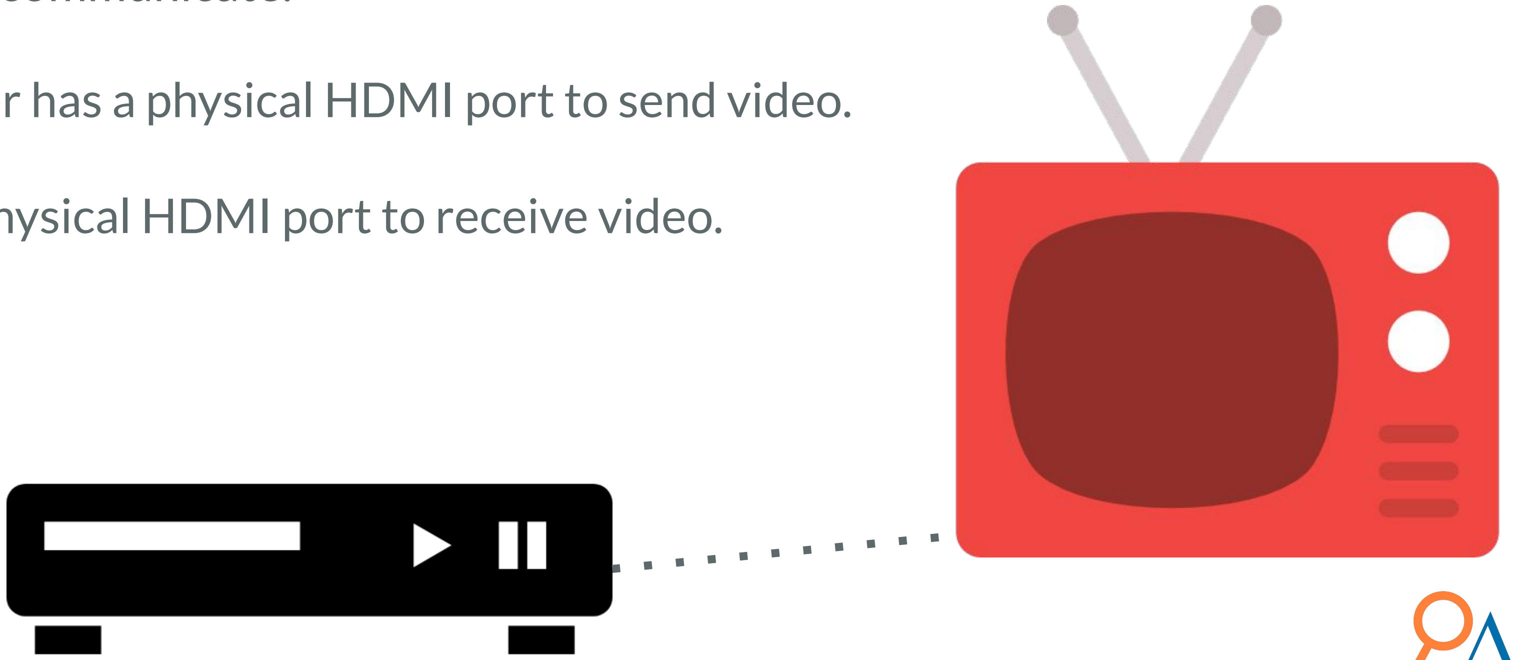


Television analogy

Think of HDMI as a common interface that two devices use in order to communicate.

The DVD player has a physical HDMI port to send video.

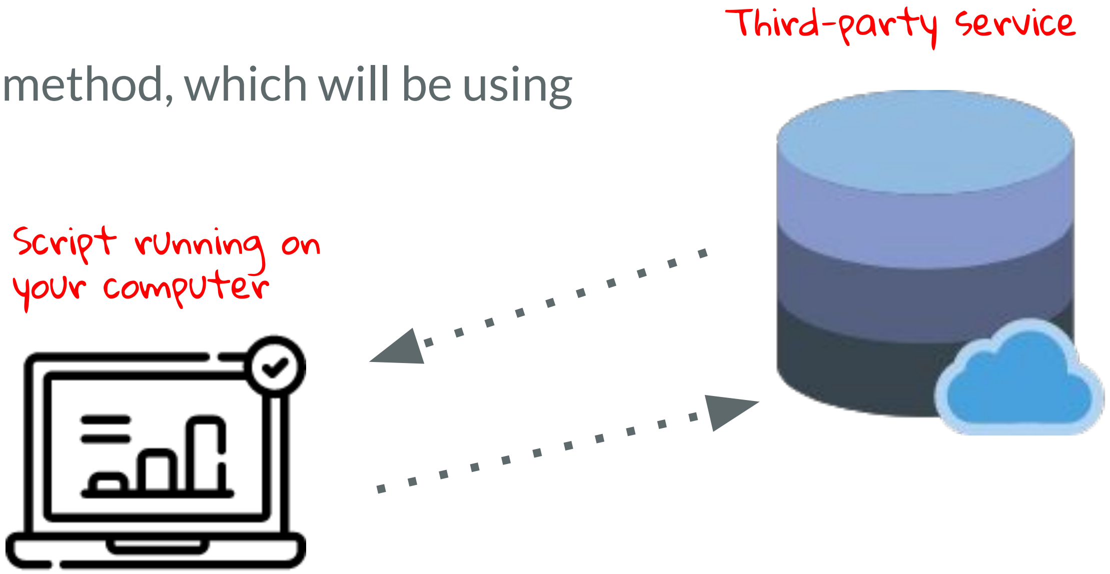
The TV has a physical HDMI port to receive video.



Television analogy

Likewise, you can send a message to a service over the web, and they will send you a message with the data you requested.

We will be discussing this method, which will be using HTTP.



HTTP

HTTP

HTTP stands for Hypertext Transfer Protocol.

HTTP encompasses sets of rules and processes for sending and receiving data on the world wide web.

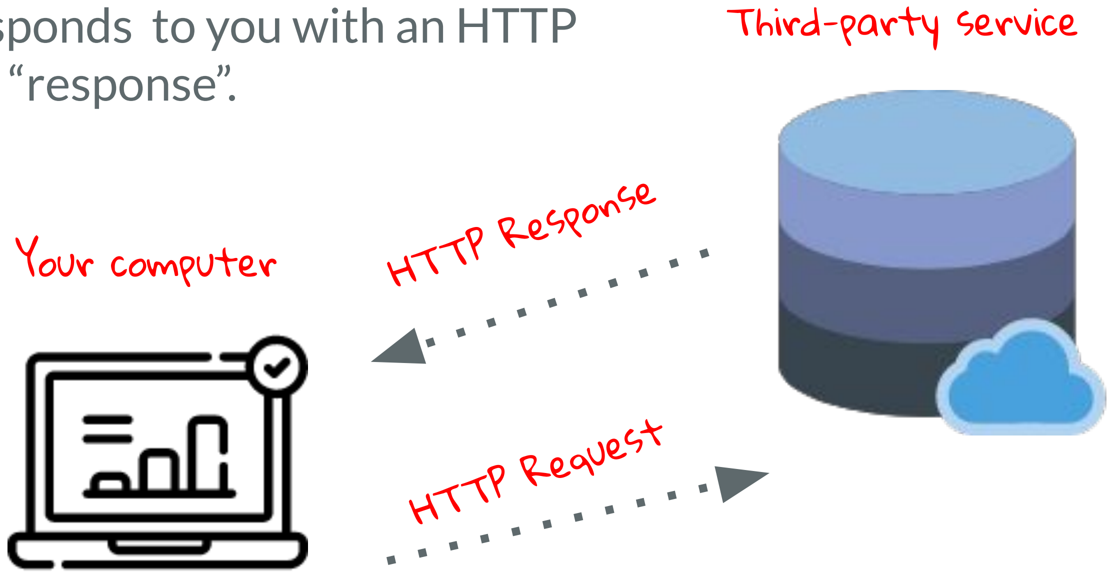
You typically don’t craft these messages yourself -- your web browser produces them, or you write a program that will produce them.

HTTP

HTTP messages

When you send an HTTP message to the third-party server, it's called a "request".

The third-party server responds to you with an HTTP message, which is called a "response".



HTTP

HTTP messages are like letters and envelopes

Letter in envelope

Your address Stamp

Destination Address

Dear ___,

…
…
…

HTTP message

Start line (Destination address, method)

Headers

Message body

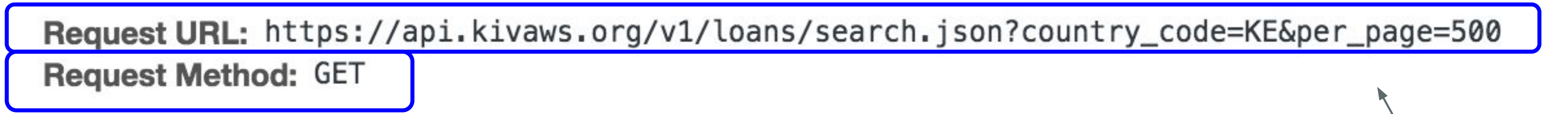
HTTP Request

HTTP

HTTP Request

Anatomy of an HTTP request

Start line

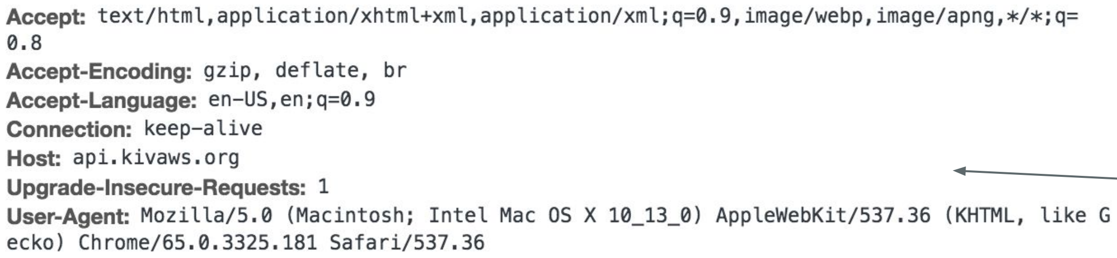


```
Request URL: https://api.kivaws.org/v1/loans/search.json?country_code=KE&per_page=500
Request Method: GET
```

"GET" is the HTTP method that tells the server that you want to retrieve data.

The URL tells the server what you want to retrieve. We'll look at this more carefully next.

Headers



```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=
0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
Host: api.kivaws.org
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_0) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/65.0.3325.181 Safari/537.36
```

You only need to know that HTTP request headers exist. For the most part, you won't have to worry about them.

No request body. GET requests do not have a request body.

Anatomy of a url

https://api.kivaws.org/v1/loans/search.json?country_code=KE&per_page=500

Protocol	https
Domain name	api.kivaaws.org
Path	/v1/loans/search.json
Query parameters	?country_code=KE&per_page=500

HTTP | HTTP Request

Url as a function

https://api.kivaws.org/v1/loans/search.json?country_code=KE&per_page=500

Now that you understand all the parts that make up a url, let's re-arrange these parts into another concept you probably already know: the function.

Think of a url like a function that takes parameters:

```
from api.kivaws.org import search_loans

search_loans(country_code='KE', per_page=500)
```

Query parameters

Query parameters contain extra data needed for this request. Sometimes it is not needed to provide this -- check the api documentation for what parameters a url needs.

It specifies key value pairs.

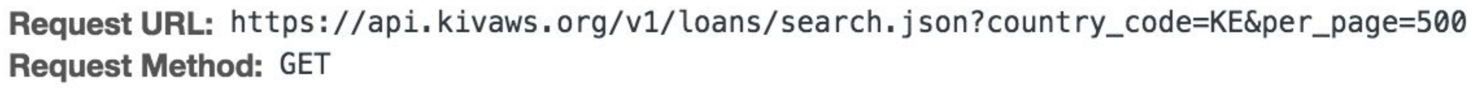
Example:
?country_code=KE&per_page=500

- Query parameters start with "?"
- It's followed by a a key (country_code)
- "=" separates the key from the value (KE)
- Then an "&" separates one key/value pair. from another key/value pair

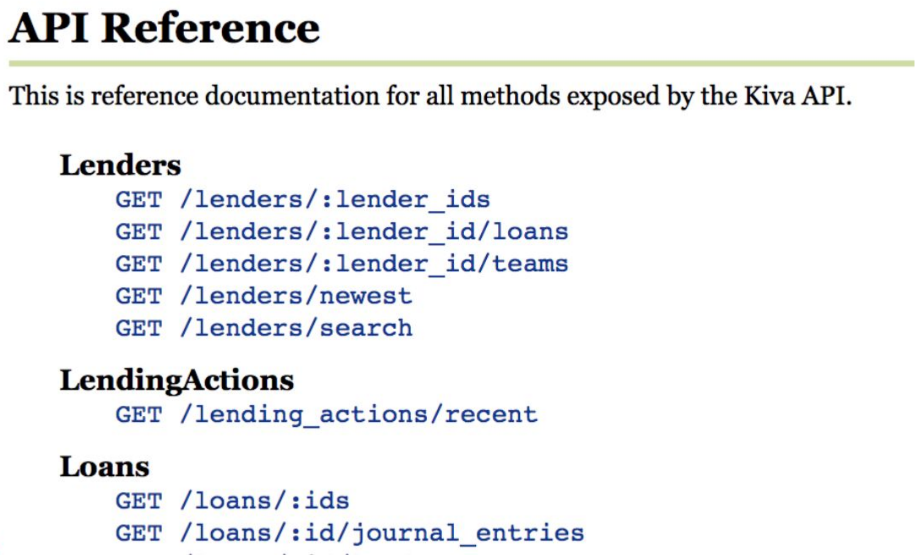
HTTP

HTTP Request

How to find API URLs



Request URL: `https://api.kivaws.org/v1/loans/search.json?country_code=KE&per_page=500`
Request Method: GET



API Reference

This is reference documentation for all methods exposed by the Kiva API.

Lenders

```
GET /lenders/:lender_ids
GET /lenders/:lender_id/loans
GET /lenders/:lender_id/teams
GET /lenders/newest
GET /lenders/search
```

LendingActions

```
GET /lending_actions/recent
```

Loans

```
GET /loans/:ids
GET /loans/:id/journal_entries
```

Services typically publicize urls for accessing data.

Search for "{Company name} API documentation".

URLs are typically organized by resource. E.g. lenders, lending actions, and loans are resources available on the screenshot on the left.

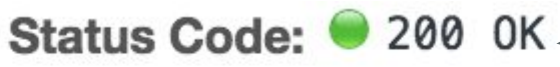
HTTP Response

HTTP

HTTP Response

Anatomy of an actual HTTP response

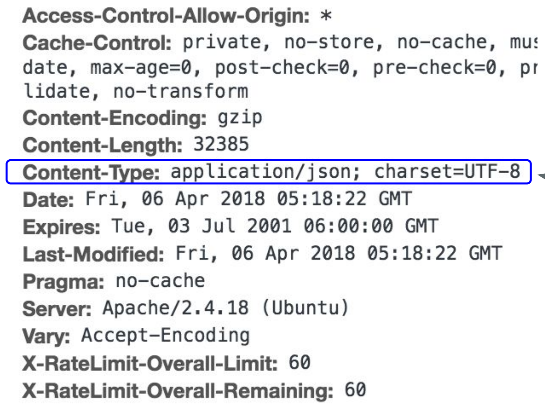
Start line



```
Status Code: ● 200 OK
```

The status code tells you whether the server has responded to your request successfully. We'll explore this next.

Headers



```
Access-Control-Allow-Origin: *
Cache-Control: private, no-store, no-cache, mus
date, max-age=0, post-check=0, pre-check=0, pr
lidate, no-transform
Content-Encoding: gzip
Content-Length: 32385
Content-Type: application/json; charset=UTF-8
Date: Fri, 06 Apr 2018 05:18:22 GMT
Expires: Tue, 03 Jul 2001 06:00:00 GMT
Last-Modified: Fri, 06 Apr 2018 05:18:22 GMT
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
X-RateLimit-Overall-Limit: 60
X-RateLimit-Overall-Remaining: 60
```

The content-type, or format which the message body is formatted, is JSON. We'll examine that more closely in a later slides.

Message Body

```
{
  "paging": {
    "page": 1,
    "total": 150934,
    "page_size": 500,
    {
      "paging": {
        "page": 1,
        "total": 150934,
        "page_size": 500,
        "pages": 302
      },
      "loans": [{
        "id": 1501869,
        "name": "George",
        "description": {
          "languages": ["en"]
        },
    ...
```

HTTP

HTTP Response

HTTP response status codes

2xx	Success!
3xx	Redirect
4xx	You did something wrong
5xx	The server did something wrong

HTTP → HTTP Response

Common HTTP Status codes

200	Success!
301	The server is redirecting you. This shouldn't be a problem for you.
400	The server thinks that you're making a bad request. Check your url and query parameters.
401/403	You don't have the proper authentication for the resource you're requesting. Check that you're providing it correctly.
404	The resource doesn't exist. Check your url.
500	Something has gone wrong on the server, so it can't provide you what you're looking for.

HTTP

In the preceding slides, you learned about urls and headers.

They tell an HTTP message

1. Where to be delivered and
2. Information about the resource

What we'll focus on next is the message body itself, where the data we're actually interested in resides!

Data formats (JSON)

HTTP

Data formats

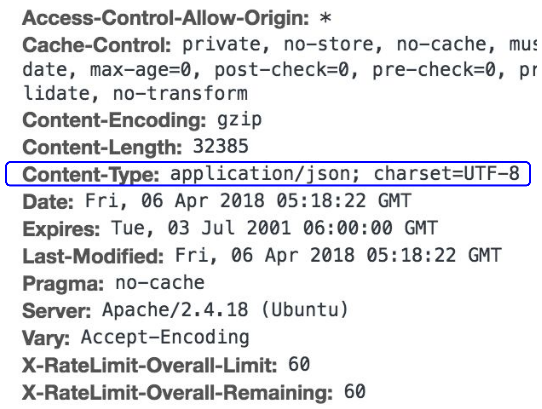
Data formats

A data format specifies how a data model is represented in a computer's memory.

We'll look at a concrete example of this in the next slide.

By far, the most popular data format used on the web to transfer data today is JSON.

Headers



```
Access-Control-Allow-Origin: *
Cache-Control: private, no-store, no-cache, mus
date, max-age=0, post-check=0, pre-check=0, pr
lidate, no-transform
Content-Encoding: gzip
Content-Length: 32385
Content-Type: application/json; charset=UTF-8
Date: Fri, 06 Apr 2018 05:18:22 GMT
Expires: Tue, 03 Jul 2001 06:00:00 GMT
Last-Modified: Fri, 06 Apr 2018 05:18:22 GMT
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
X-RateLimit-Overall-Limit: 60
X-RateLimit-Overall-Remaining: 60
```

Anatomy of a JSON object

```
{
"loans": [{
      "id": 1505488,
      "description": {
        "languages": ["en"]
      },
      "status": "fundraising",
      "funded_amount": 0,
      "sector": "Food",
      "use": "to buy varieties of cereals to sell so she can earn extra income to support her children.",
      "location": {
        "country_code": "KE",
        "town": "Chuka",
      },
      "posted_date": "2018-04-12T05:10:06Z",
      "bonus_credit_eligibility": true,
   }]
}
```

Object (key and value pair/dictionary)

Array (list/sequence)

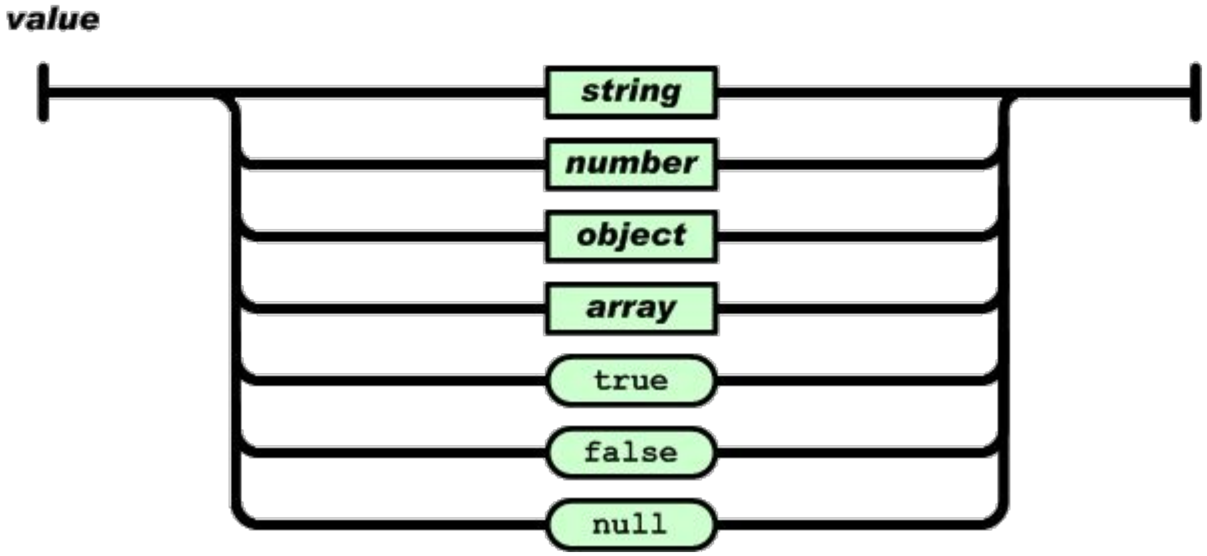
Number (ints, floats)

String

Boolean (true/false)

Values

A value can be any of these types:



HTTP

Data formats

Objects

An object is a key value pair. It is represented by:
- two curly braces, an opening and a closing bracket { }
- a key, which must be a string
- a colon separates the key from
- a value

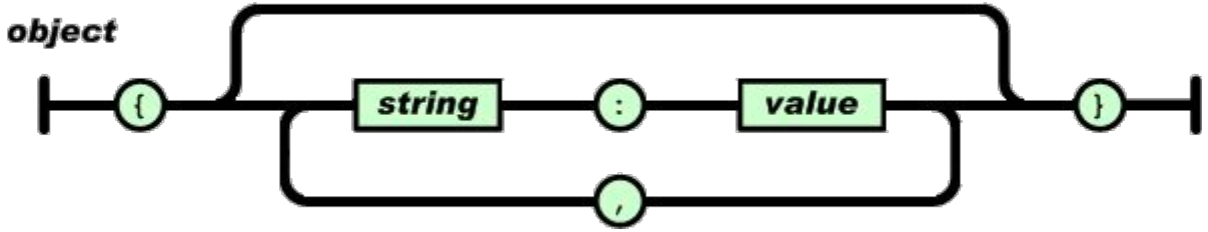


Image credit: https://www.json.org

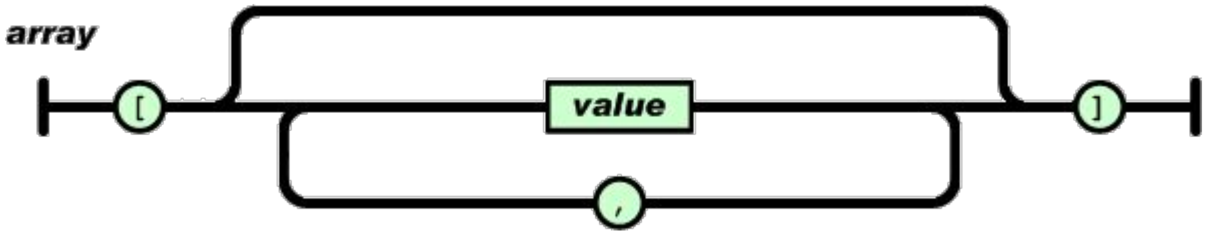
HTTP

Data formats

Array

An array, or a list of values, is represented by:

- An opening and closing bracket
- A value
- Commands separate values in the list



Consuming APIs in Python

Requests package

The “Requests” package is the recommended package for general use by the official Python documentation.

Its documentation can be found here.

Installation instructions will be included in the Jupyter notebook.

APIs in Python

How to use Requests

```
import requests
```

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
```

Send an HTTP message and save the response as r

```
>>> r.status_code
200
```

Check that the request was successful

```
>>> r.headers['content-type']
'application/json; charset=utf8'
```

Check the data format of the message body

```
>>> r.json()
{u'private_gists': 419, u'total_private_repos': 77, ...}
```

Examine the message body

In past slides, we learned about interfaces, http, json, and using the requests package in python.

You have gained a superpower! With great power comes great responsibility.

Let’s discuss data access ethics.

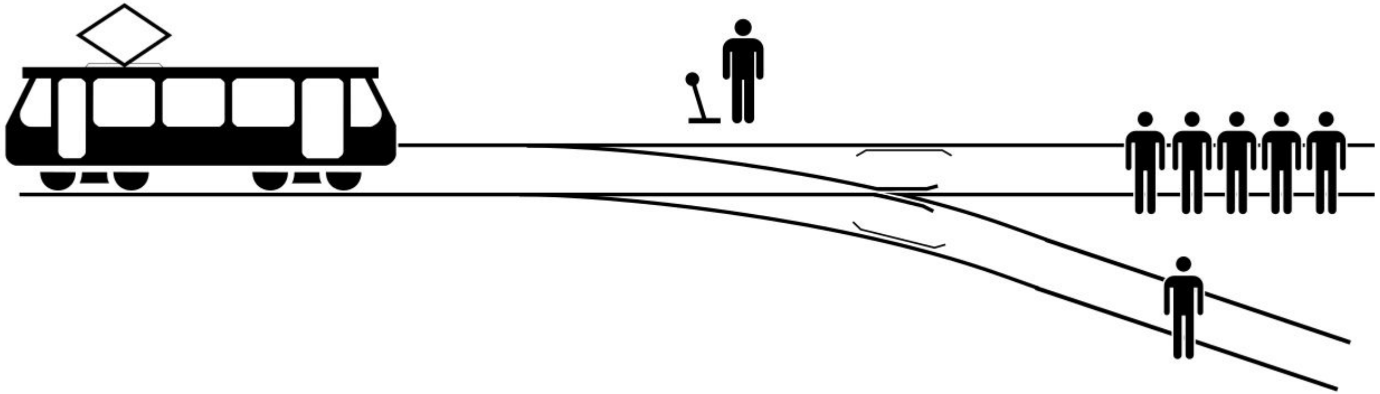
Data Access Ethics

Data Ethics

Ethics

What are ethics?

Ethics are shared social rules of a society that people follow in order to benefit from.



Data science ethics

Topics in Data Science ethics may include

- Informed consent
- Data ownership
- Privacy
- Algorithmic fairness

We won’t be discussing all of these, as this module focuses on retrieving data. To that end, we will focus on informed consent and privacy in regards to data collection methods.

(Source https://www.coursera.org/learn/data-science-ethics)

Real-life case study

- The question posed by this case study is: When the exact data you want is unavailable, but is made available by hackers, can you use that data ethically?

- We will look at the backstory, the debate, and the real-life decisions chosen by researchers. Then we will present to you a series of guidelines that will help you make decisions if you encounter such issues in the future.

(Source: https://bdes.datasociety.net/wp-content/uploads/2016/10/Patreon-Case-Study.pdf)

Data Ethics

Backstory and facts

- A group of data-scientists wanted to analyze crowdfunding.
- They were interested in data from Patreon, a crowdfunding website.
- Patreon didn't offer an API. The researchers considered scraping webpages from Patreon, because there would be tens of thousands pages.
- However, they couldn't guarantee they would have scraped all project pages. This means that their data would have been a convenience sample and researchers could not be certain of conclusions based on that sample.

(Source: https://bdes.datasociety.net/wp-content/uploads/2016/10/Patreon-Case-Study.pdf)

Backstory (continued)

- Just as they were considering this, Patreon was hacked in October 2015. All data was made available, including public data on projects (which the researchers were interested in), private messages, emails, and passwords.
- The method the data was gathered is not legal under US law.
- However, the data that the researchers were interested in was already public before the hack, and now made easily accessible.
- Some researchers felt that because the data was now public, they could ethically use it, while others disagreed.

(Source: https://bdes.datasociety.net/wp-content/uploads/2016/10/Patreon-Case-Study.pdf)

Past cases

- Upon disagreement, the researchers looked at past cases in journalism.

- Some examples were:
    - A New York state newspaper published names and addresses of gun owners in the readership area, having retrieved the data publically. Those people lashed out against the newspaper, so making public data more widely public is not always seen as appropriate.

    - Mark Burnett, a security researcher, collected usernames and passwords from illegal data dumps. He publically released his collection in order to facilitate research. He claimed that no further harm could result from this because the data was already publicly available, and now could be more easily used for research purposes.

- After examining past cases, the researchers debated.

(Source: https://bdes.datasociety.net/wp-content/uploads/2016/10/Patreon-Case-Study.pdf)

Data Ethics

Debate

Arguments in Favor of Use

1. Data is public, like a newspaper.
2. We hope to serve the public good via our work.
3. This is the data we want, but we can’t get it via other methods.

Arguments Against Use

1. Researchers have a limited capability to distinguish between public and private information within the hacked data.
2. May see private data when cleaning the data.
3. Perhaps legitimizing criminal activity.
4. Violating users’ expectation of privacy.
5. Using people’s data without consent.

(Source: https://bdes.datasociety.net/wp-content/uploads/2016/10/Patreon-Case-Study.pdf)

Decision

- In the end, the researchers decided not the use the data released by the hackers.
- They felt that the arguments against the use of the data were stronger than the arguments for.
- Even though they would use the data for good and only use parts of the data that were already public, they decided the negatives outweigh the positives.

(Source: https://bdes.datasociety.net/wp-content/uploads/2016/10/Patreon-Case-Study.pdf)

Data Ethics

Data science ethics laws/guidance

- Laws are beginning to catch up to practice, as the EU's General Data Protection Regulation (GDPR) laws come into effect in May 2018.
  - These laws aim to protect the privacy of all individuals in the EU. For example, entities that collect data must specify a purpose for doing so and use it for that purpose only.

- Data science as an industry lacks a universally accepted code of ethics. There are individuals and companies that are creating "data ethics manifestos". The next slide will present one set of principles.

Two-point code of conduct

- Professor H.V. Jagadish, from the University of Michigan, came up with a two point code of conduct for data scientists:

1. Do not surprise the subject in collection and use of the data.
2. “Own” the outcome. If the process leads to an undesirable outcome, fix it.

(Source: https://www.coursera.org/learn/data-science-ethics)

Checklist:

- ✓ Interfaces
- ✓ HTTP
- ✓ Data Formats (JSON)
- ✓ Consuming APIs in Python
- ✓ Data access ethics

Advanced resources

Want to take this further? Here are some resources we recommend:

- HTTP Overview from Mozilla
- JSON overview from Mozilla
- Requests package
- Data Science Ethics course