

# Scraping Media from the Web with Python

---

## DOWNLOADING MEDIA FILES



**Allen O'Neill**

ENGINEER

@databytzai

# Overview/ Summary



**Investigation & preparation**

**Download different media types**

**Learn to optimise downloads**

# Investigation & Preparation

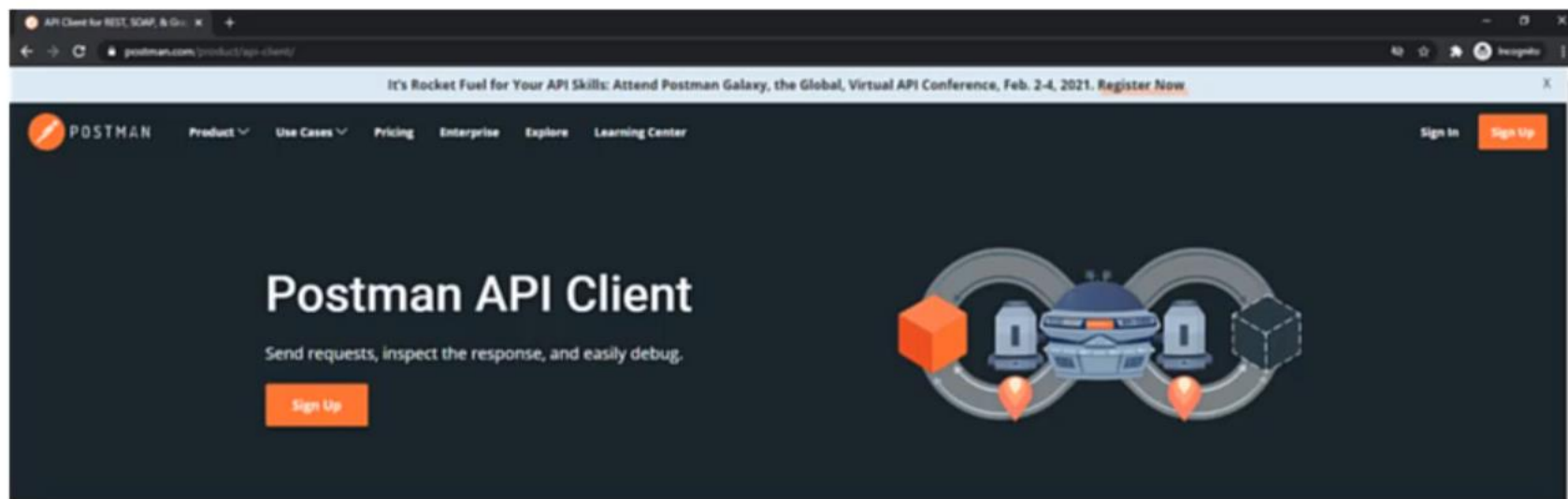
---

Investigate

Take Action

GET vs HEAD

# https://postman.com/downloads/



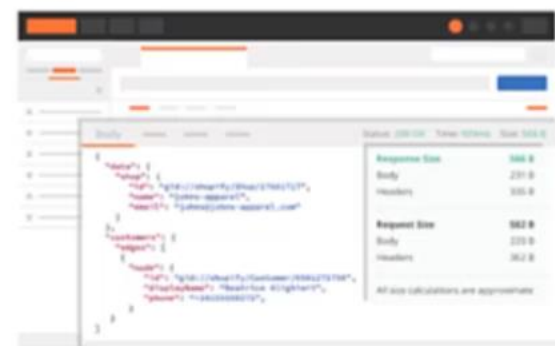
## Send Requests and View Responses

Create and execute any REST, SOAP, and GraphQL queries from within Postman.

- **Define complex requests**

Send any type of request in Postman. Create and save custom methods and send requests with the following body types:

- **URL-encoded**—The default content type for sending simple text data
- **Multipart/form-data**—For sending large quantities of binary data or text containing non-ASCII characters
- **Raw body editing**—For sending data without any encoding
- **Binary data**—For sending image, audio, video, or text files



GET

▼

Enter test URL

Params

Send

Save

Body

Pre-request Script

Tests

No Auth

Hit the Send button to get a response.

Do more with requests

Share

Mock

Monitor

Document

Body

Cookies

Headers (10)

Test Results

**accept-ranges** → bytes

**connection** → Keep-Alive

**content-length** → 50424

**content-type** → image/jpeg

**date** → Mon, 28 Dec 2020 17:38:11 GMT

**etag** → "c4f8-588101e5fc380"

**keep-alive** → timeout=5, max=100

**last-modified** → Sat, 04 May 2019 13:48:14 GMT

**server** → Apache

**x-server** → 3239



## Postman

<https://postman.com>

## Charles Proxy

<https://www.charlesproxy.com>

## Fiddler

<https://www.telerik.com/fiddler>



# Downloading Different Media Types

---

# Downloading Different Media Types

## Different types of media



Image - relative path



Image - absolute path

Link to [Pluralsight website](https://www.pluralsight.com)

[Powerpoint media slides](#)

[ZIP file containing media](#)

[Text file example](#)

[Audio file](#)

[Video file](#)

<http://www.howtowsrape.com/examples/media4.html>

# Demo



Download web page

Parse for links

Add to memory list

Download link target

Save to folder

# Code Example

Image Links

```
<h1>Different types of media </h1>

 Image - relative path </br>
<br/><br/><br/>
 Image - absolute path </br>

<h3>Link to <a href="https://www.pluralsight.com/">Pluralsight website</a> </h3>
<br/><br/><br/>
<strong>
<a href="http://howtowsrape.com/examples/media/images/SampleSlides.pptx">Powerpoint media slides</a></br>
</strong>
</br>
<h2>
<a href="http://howtowsrape.com/examples/media/images/SampleZip.zip">ZIP file containing media</a></h2></br>
</br>
<a href="http://howtowsrape.com/examples/media/images/SampleText.txt">Text file example</a></br>
<br/><br/>
<a href="http://howtowsrape.com/examples/media/images/Clapping.mp3">Audio file</a></br>
<br/><br/>
<a href="http://howtowsrape.com/examples/media/images/BigRabbit.mp4">Video file</a></br>
```

Downloadable HREF

<http://www.howtowsrape.com/examples/media4.html>

# Streaming Large Files

---

# Downloading Media Files

## Small Files

Simple GET  
Fast

## Larger Files

Problematic  
Streamed

```
def downloadFile(AFileName):  
  
    filename = AFileName.split("/")[-1]  
  
    rawImage = requests.get(AFileName,  
                             stream=True)  
  
    with open(filename, 'wb') as fd:  
        for chunk in  
            rawImage.iter_content  
                (chunk_size=1024):  
            fd.write(chunk)  
  
    return
```


- ◀ Define function
- ◀ Extract filename
- ◀ Requests using 'stream = True' parameter
- ◀ Open file for writing
- ◀ Receive data chunks
- ◀ Write streamed data chunk to file

# Downloading YouTube Video

---



# PAFY Library



[Help](#) [Sponsor](#) [Log in](#) [Register](#)

## pafy 0.5.5

`pip install pafy`

 Latest version

Released: Nov 22, 2019

Retrieve YouTube content and metadata

### Navigation

 Project description

 Release history

 Download files

---

### Project links

 Homepage

 Download

---

### Statistics

View statistics for this project via [Libraries.io](#), or by using [our public dataset on Google BigQuery](#)

### Project description

`pypi` `v0.5.5`

`downloads` `25k/month`

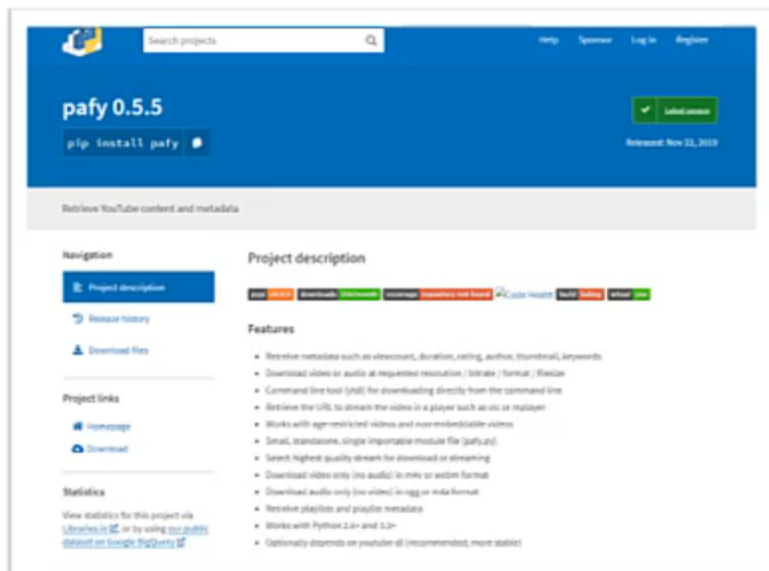
`coverage` `repository not found`

 `build` `failing`

`wheel` `yes`

### Features

- Retrieve metadata such as viewcount, duration, rating, author, thumbnail, keywords
- Download video or audio at requested resolution / bitrate / format / filesize
- Command line tool (ytld) for downloading directly from the command line
- Retrieve the URL to stream the video in a player such as vlc or mplayer
- Works with age-restricted videos and non-embeddable videos
- Small, standalone, single importable module file (pafy.py)
- Select highest quality stream for download or streaming
- Download video only (no audio) in m4v or webm format
- Download audio only (no video) in ogg or m4a format
- Retrieve playlists and playlist metadata
- Works with Python 2.6+ and 3.3+
- Optionally depends on youtube-dl (recommended; more stable)



## PAFY:

- Open Source
- Pip install pafy
- Always seek permission!

# Optimising Media Downloads

---



**Maximize compute resources**

**Reduce server impact**

**Obey rules**

**Ensure you scrape:**

- Responsibly
- Legally
- Ethically

# Accept-Ranges

The Accept-Ranges response HTTP header is a marker used by the server to advertise its support of partial requests.

Authorization

Headers

Body

Pre-request Script

Tests

Type

No Auth

Body

Cookies

Headers (10)

Test Results

accept-ranges → bytes

connection → Keep-Alive

content-length → 26316800

content-type → text/plain

etag → "3200000-5b7c4ef63e469"

Part-1

Part-2

Part-3

Merged\_File





Auto-resume

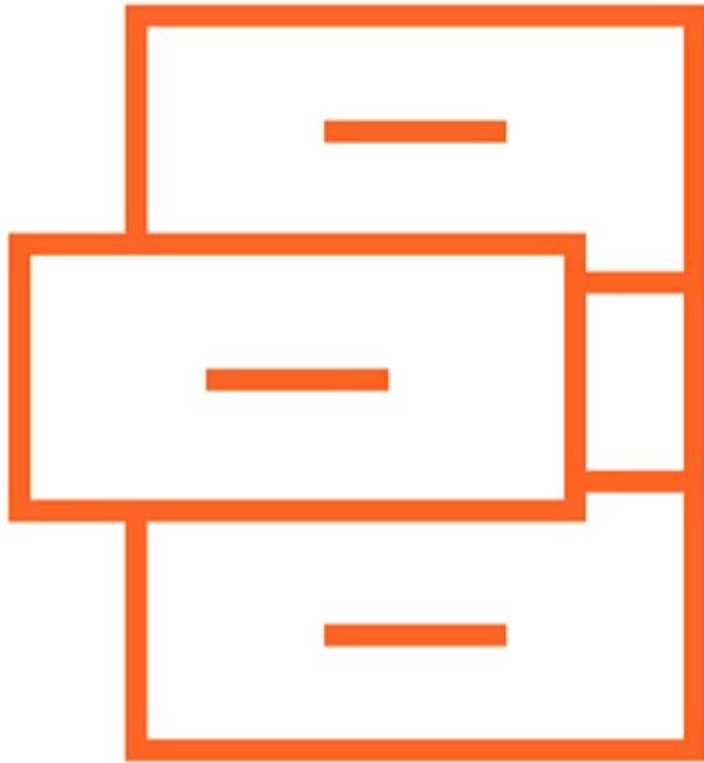
Network failure

Content length – failure point

# Persisting Media

---





## **Persistence options**

- General file storage
- Structured database

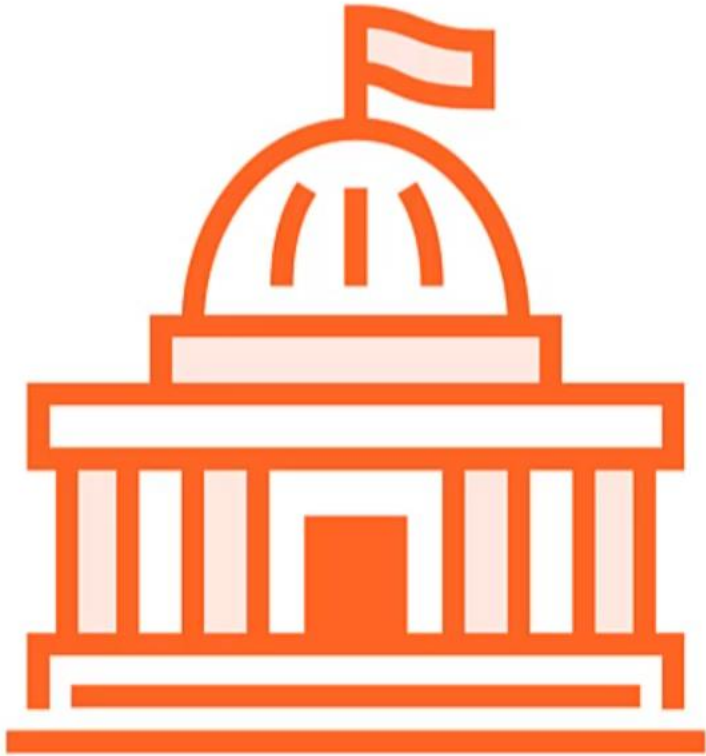
## **Specific use case**

- Data privacy
- Legal requirements



## Cost of storage

- Data grows quickly
- Cost at rest, and in motion
- Consider 'cold storage'



## Considerations

- Access requirements
- Audit track
- Data governance