

# Scraping Music Insights From YouTube with Selenium


A Real-Time Data Approach for Artist and Analysts



Murendeni Nethengwe



# Agenda

1. Introduction to Web Scraping
  2. Why YouTube as a Data Source?
  3. Data Extraction Goals
  4. Overview of the scraping Script
  5. Handling JavaScript Rendering
  6. Challenged and Solutions
  7. Utilization of Extracted Data
  8. Lessons Learned and Future Extensions
- 

# Web Scraping YouTube: Techniques and Insights

An in-depth look into the methodology and applications

## **Defining Web Scraping**

This is the process of automatically extracting large amounts of data from websites.

## **Key purpose and applications**

This is commonly used for data analysis, research, and business intelligence. We can use this for accessing web pages and extracting desired data from the pages

## **Presentation Overview**

A structured breakdown of key topics related to YouTube scraping, showing different techniques, challenges and futuristic applications

# Why YouTube as a Data Source?

Exploring the reason behind why I selected YouTube for scraping



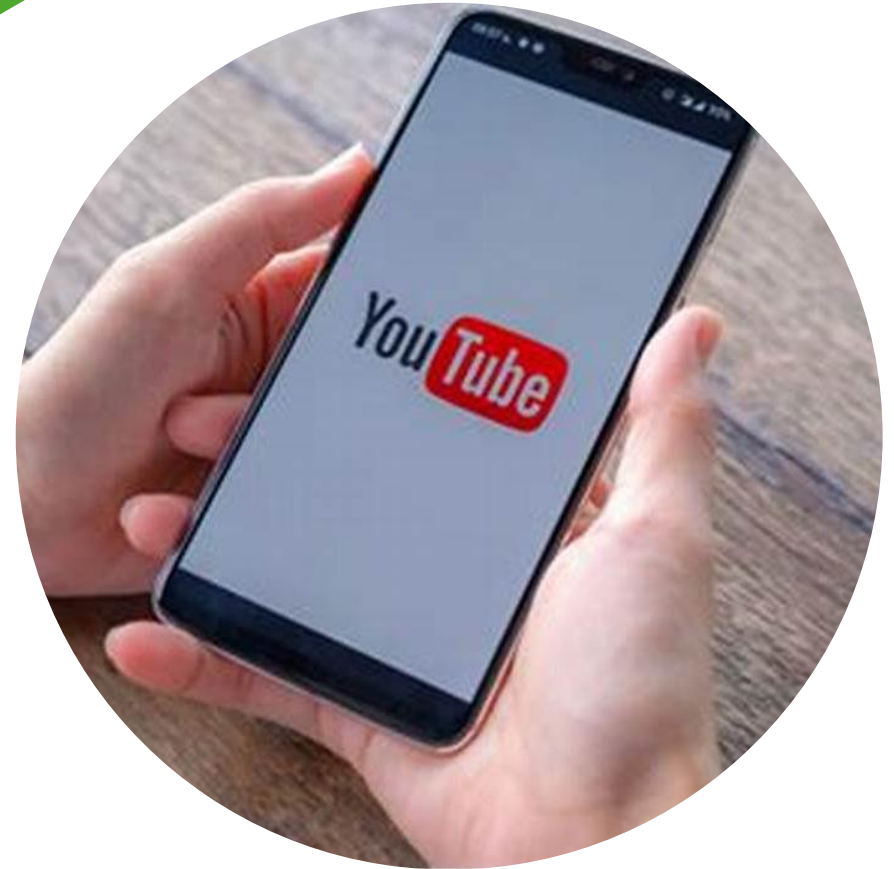
**Rich data source:** YouTube houses a wealth of content comprising videos, comments, statistics, and metadata, making it a treasure chest for data extraction and scraping



**Popularity and relevance:** As one of the largest platforms with billions of users, YouTube's trends and user's interactions offer critical insights into global social and cultural dynamics



**Consistent HTML structure:** Despite its dynamic nature, YouTube maintains a relatively consistent HTML structure across its pages. This consistency simplifies the process of locating and extracting specific element and metadata



# Data Extraction Goals

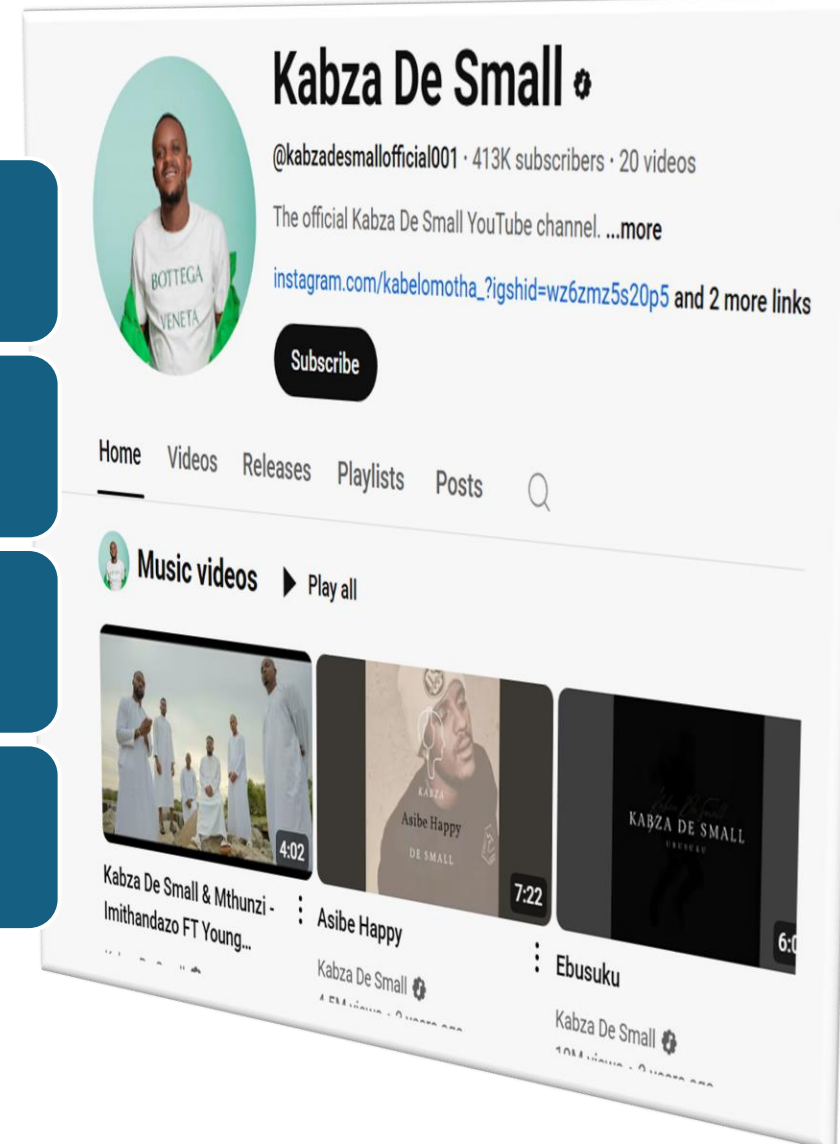
What specific data we are extracting

**Video title:** The name of each video uploaded by the artist

**Number of views:** The number of views each video has received

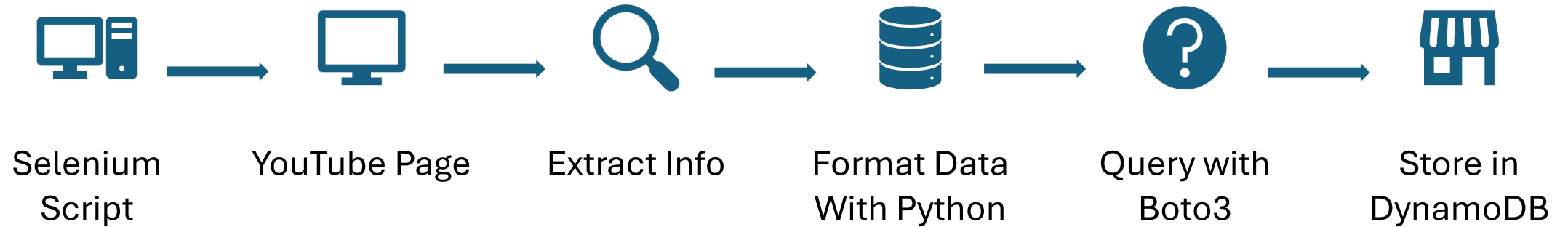
**Upload date:** How far back in days the video was uploaded

**Importance of the data:** Understanding these extracted data can lead to strategic decisions in content creation, marketing and further research initiatives. This data can provide valuable insights into the popularity engagement of artist content



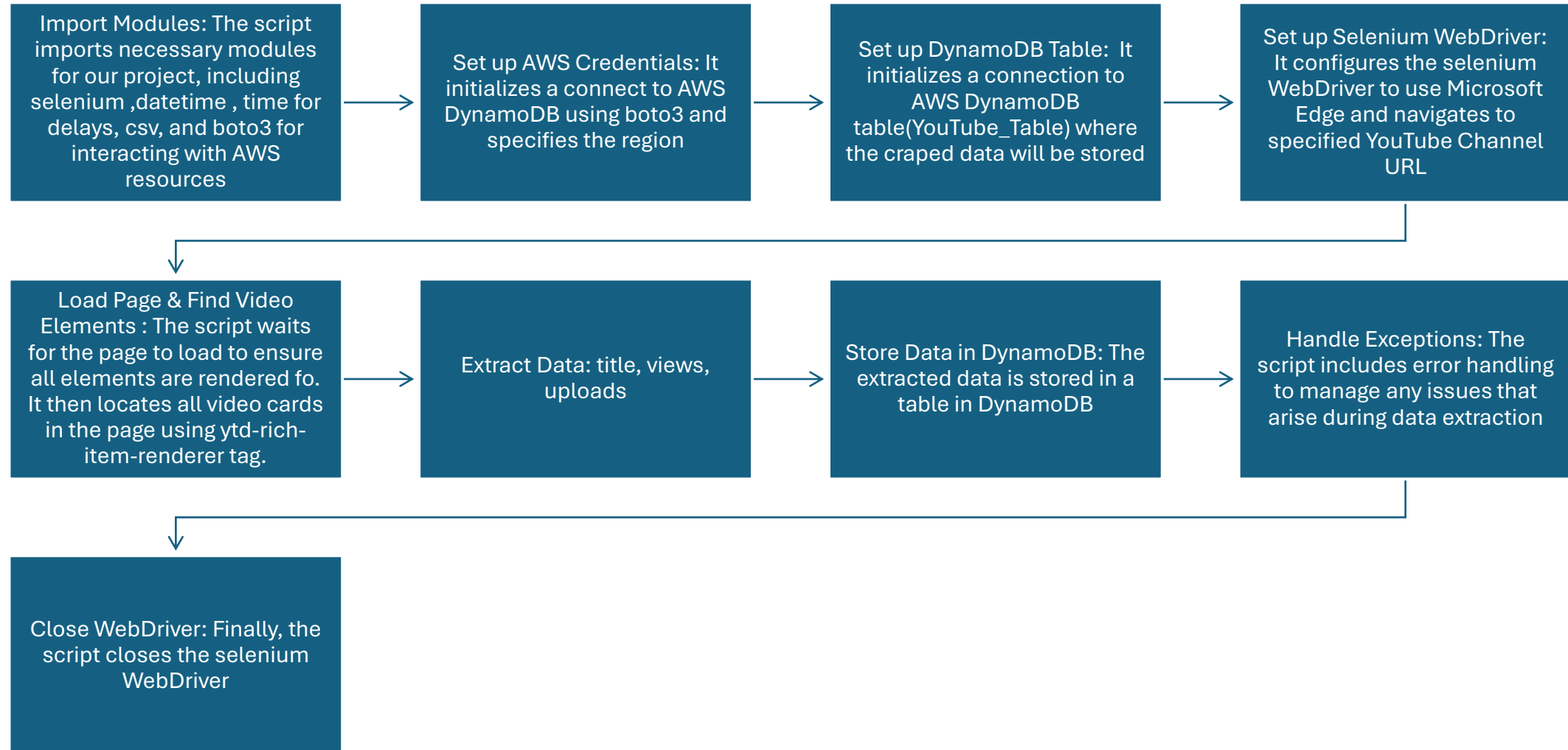
# Tech Stack

How it all connects



# Overview of the scraping Script

## Understanding the components and setup



# Handling JavaScript Rendering

Strategies for overcoming challenges



**Challenges with JavaScript :** Many pages, including YouTube, rely heavily on JavaScript for rendering content, which can complicate traditional scraping methods that fetch static HTML



**Using Selenium WebDriver:** Selenium facilitates interaction with web pages as if a user were navigating them, which is vital for accessing elements rendered by JavaScript post page-load



**Waiting for element to load:** The script includes a `time.sleep()` command to pause execution for a few seconds allowing enough time for the JavaScript on the page to render all video elements



# Challenges and Solutions in Data Scrapping

Addressing common obstacles



**JavaScript rendering issues:** Dynamic content often poses a threat to consistent data extraction; adopting headless browsing can mitigate these concerns



**Data storage & Management:** As the volume of data increases, so does the complexity of managing it; employing a structured database like DynamoDB ensures organized storage and accessibility



**Using AWS DynamoDB:** Leveraging this service provides scalable, high performance storage solutions tailored for rapid retrieval in data-intensive application

# Utilization of Extracted Data

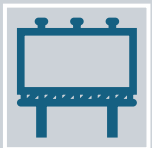
Transforming data into actionable insight



**Performance analysis** : interpreting scraped metrics allows for evaluation of content performance and audience engagement within YouTube's ecosystem



**Trending monitoring:** Continuous evaluation of viewership data can unveil patterns and shifts in user behaviour, aiding in future content strategy



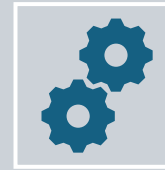
**Marketing and promotion strategies:** Insights derived from the analysis can inform targeted marketing efforts, SEO optimizations, and enhancement of user interaction

# Lessons and Future Extension

Reflecting on experiences to enhance future project



**Data extraction techniques:** Evaluating various methodologies enhances our scraping efficiency and accuracy, allowing for broader applications across different platforms



**Error handling:** Diligent error management ensures robust scraping scripts, enabling smoother operations in the face of unpredictable web changes



**Data validation and cleaning:** Rectifying inconsistencies post-extraction ensures integrity of our analyses, paving the way for reliable conclusion



**Potential for API integration and machine learning:** Future developments may explore utilizing APIs and machine learning to streamline and automate data extraction processes

Live Demo

# Live Demo

The screenshot displays the Visual Studio Code interface with a Python script named `selenium_scraper.py` open in the editor. The Explorer sidebar on the left shows the project structure, including a `.venv` directory, a `WebDriver` folder, and the `project2.py` file. The script in the editor is a Selenium-based scraper that interacts with a DynamoDB database and a YouTube channel.

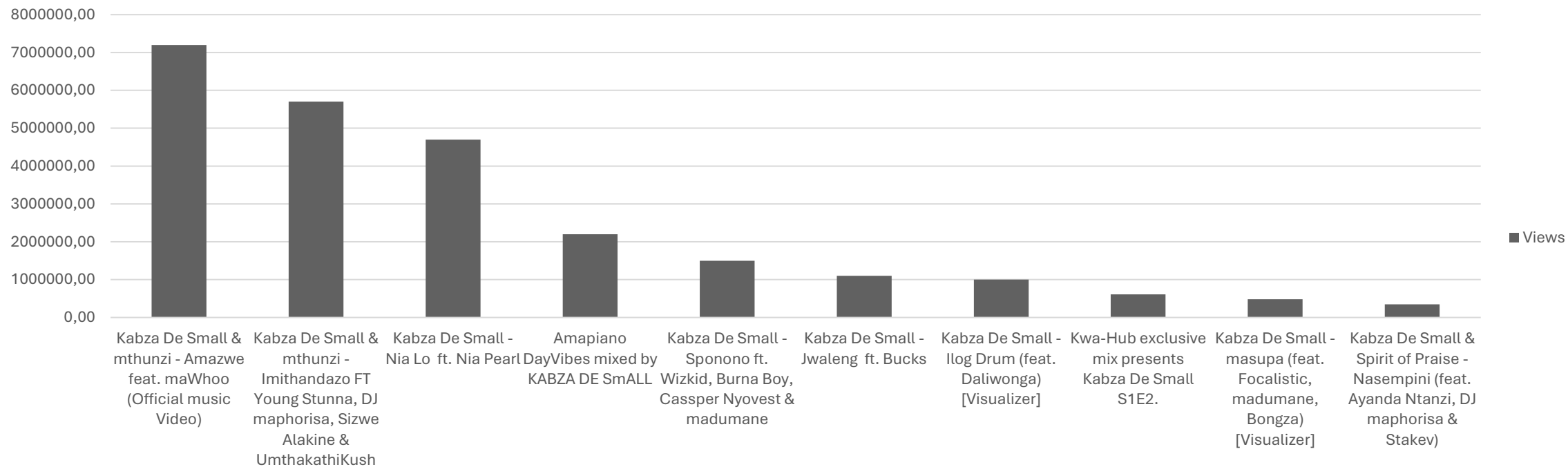
```
1 # Import modules
2 from selenium import webdriver
3 from selenium.webdriver.common.by import By
4 from datetime import datetime
5 import time
6 import csv
7 import boto3
8
9 # Set up AWS credentials
10 dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
11
12 # Set up DynamoDB table
13 table = dynamodb.Table('YouTube_Table')
14
15 # Set up Selenium WebDriver
16 url = "https://www.youtube.com/@kabzadesmallofficial001/videos?view=0&sort=p&flow=grid" # URL of the Y
17 edge_driver_path = "C:/Users/Kone/Documents/proj/WebDriver/msedgedriver.exe" # Path to the Edge driver
18
19 # Set up Edge WebDriver
20 options = webdriver.EdgeOptions()
21 driver = webdriver.Edge(options=options)
22
23 # Navigate to the YouTube channel
24 driver.get(url)
25
26 # Wait for videos to load
27 time.sleep(5)
28
29 # Find all video cards
30 videos = driver.find_elements(By.TAG_NAME, "ytd-rich-item-renderer")
31
32 # Loop through videos and extract data
33 for video in videos:
34     try: # Try to extract data from the video card
```

An "Activate Windows" watermark is visible in the bottom right corner of the editor area.

The status bar at the bottom shows the current file is `master*` and includes various utility buttons like `BLACKBOX Chat`, `Add Logs`, `CyberCoder`, `Improve Code`, `Share Code Link`, `Open Website`, `Generate Commit Message`, `Select Interpreter`, and `BLACKBOXAI: Open Chat`.

# Analysis of Scraped Data

Top 10 Most Viewed





# Thank You

