Systembeskrivning: Videoskrapa

Innehåll

[Prisskrapa 1](#_Toc137542436)

[Tjänster som ingår i skrapningen 1](#_Toc137542437)

[Struktur över filer 1](#_Toc137542438)

[Beskrivning av filer: 2](#_Toc137542439)

[Scraper.py 2](#_Toc137542440)

[Selenium 2](#_Toc137542441)

[Beautiful soup 2](#_Toc137542442)

[Uppbyggnad av tjänster genom pythonfiler, exempel: **AmazonPrime\_SE.py** 2](#_Toc137542443)

[Uppbyggnad av tjänster, exempel YoutubePremium\_SE.py 4](#_Toc137542444)

[*Pythonfiler*: **main.py** 5](#_Toc137542445)

[*Excelfiler:* video\_DK\_2023.xlsx, video\_FI\_2023.xlsx, video\_SE\_2023.xlsx, video\_NO\_2023.xlsx 7](#_Toc137542446)

# Prisskrapa

* Skrapar paket för olika streamingtjänster. Skrapningen inkluderar namn på paketet, pris, eventuella kampanjer och information om paketet

## Tjänster som ingår i skrapningen

**Sverige:** Netflix, Viaplay, Cmore, Disney+, HBO Max, Discovery+, Amazon Prime Video, Apple TV+, Tele2 Play+, Youtube Premium, Skyshowtime, Hayu, Eurosport

**Finland:** Netflix, Viaplay, Cmore, Disney+, HBO Max, Discovery+, Amazon Prime Video, Apple TV+, Youtube Premium, Ruutu+, Skyshowtime, Hayu, Eurosport

**Norge:** Netflix, Viaplay, Disney+, HBO Max, Discovery+, Amazon Prime Video, Apple TV+, Youtube Premium, TV2 Play, Stim NO, Skyshowtime, Hayu, Eurosport

**Danmark:** Netflix, Viaplay, Cmore, Disney+, HBO Max, Discovery+, Amazon Prime Video, Apple TV+, Youtube Premium, TV2 Play, Skyshowtime, Hayu, Eurosport

# Struktur över filer

**En bild som visar text, skärmbild, diagram, linje

Automatiskt genererad beskrivning**

*Figuren ovan visar hur de olika filerna är sammankopplade. En mer detaljerad beskrivning av respektive fil ses nedan.*

# Beskrivning av filer:

## Scraper.py

* 'Startar upp' skraporna som används för skrapningen dvs. Beutiful Soup och Selenium. Denna fil måste importeras till alla filer som ska skrapa information. Importeras alltid med namnet scrape.
* En hemsida som skrapas med selenium använder en "driver" som hämtas i funktionen **scrape.selenium\_site(URL).**
* En hemsida som använder Beutiful Soup använder en "soup" som hämtas i funktionen **scrape.b24\_scrape(URL)**

### Selenium

* + En bild som visar text, skärmbild, Teckensnitt

    Automatiskt genererad beskrivning
  + Chromedriver är en fil som är definierad som ’chromedriver.exe’ och krävs för att köra selenium
  + Denna filen kan behövas uppdateras i samband med uppdateringar av Google chrome, ladda ner en uppdaterad version av chromedriver då från följande länk: <https://chromedriver.chromium.org/>!
  + Raden: *options.headless = False* gör så att fönstrena öppnas i samband med skrapning. Om denna sätts till *True* kommer fönstret inte att öppnas, detta är ej rekommenderat då det blir svårare med felhantering då.
  + **FELHANTERING:** Om chromedriver inte uppdateras 🡪 FELMEDDELANDE!

### Beautiful soup

* + Används på ett fåtal tjänster där det inte behövs att man ’klickar’ på hemsidan.

## Uppbyggnad av tjänster genom pythonfiler, exempel: **AmazonPrime\_SE.py**

* Använder sig av både Beautiful Soup och Selenium.

**En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning**

* I metoden **price()** läses <URL:n> och startas med raden:
  + soup = scrape.bs4\_scrape(self.URL)
* Sedan hämtas alla taggar som ”p”-taggar i HTML-koden alltså <p> </p>, vilka bland annat innehåller priset. För att få ut priset används en regex för att endast ta ut siffrorna.
* En lista med priserna returneras i metodens slut!

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

* Metoden **information(driver)** använder selenium och en ”driver” som läser in hemsidan tas in som ett argument.
* Information om paketen hittas via classnamn på HTML-koden på hemsidan. I detta fall ligger informationen i ’dv-copy-body’
* En lista med informationen returneras i slutet av metoden

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

* Metoden **create\_object()** kallar på de andra metoderna och lägger in all information i en dictionary self.SE, som i sin tur har fyra olika listor [”Package”], [”Price”], [”Campaign”] och [”Information”]
* Avslutningsvis finns det lite felhantering som kollar så att det finns information i alla listor och att alla är lika långa.
* **FELHANTERING:**
  + Om listorna är tomma, ett tecken på att tex hemsidan har uppdaterat sina taggar/klasser. Då måste detta uppdateras i koden!
  + Om listorna är olika långa, prova att bara köra t ex Amazon och printa för att se vilken lista som saknar information t ex. self.SE[”Price”] och undersök om det skett t ex förändringar på hemsidan

## Uppbyggnad av tjänster, exempel YoutubePremium\_SE.py

* Youtube är uppbyggd med hjälp av en metod *scrape\_all()* och använder selenium då klick krävs på hemsidan
* Ibland kan man bli blockad på tjänsten, därför används en try:/except:-sats för att kunna öppna tjänsten på nytt om hemsidan skulle låsa sig.
* För att hitta information används find\_element\_by\_xpath(). Detta uttryck hittas genom att kopiera ’full xpath’ i inspektor mode på hemsidan. Se bilder nedan.

*En bild som visar skärmbild, text

Automatiskt genererad beskrivning*

*Bilden ovan visar hur det ser ut när man kopierat ett xpath-uttryck till koden*

En bild som visar text, Teckensnitt, skärmbild

Automatiskt genererad beskrivning

**FELHANTERING:** *Bilden ovan visar hur det ser ut i inspektionsmode. Genom att klicka på det element som önskas kopieras och sedan ’kopiera hela xpath-uttrycket’. Kan xpath-uttryck uppdateras om hemsidan skulle förändras eller uppdateras.*

## *Pythonfiler*: **main.py**

* En bild som visar text, skärmbild

  Automatiskt genererad beskrivningI filen importeras samtliga filer
* För respektive land skapas det upp object och genom metoden create\_object() för respektive tjänst som kör skrapningen för den tjänsten. Se exempel i bilden nedan.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

* Sedan används merge\_output-klassen för att merga för respektive land där instansvariablerna SE, FI, NO och DK står för de olika länderna.
  + Klassen *MergeOutput* används för att kombinera datan från de olika skrapningarna till en lång dictionary per land. Detta för att kunna skriva ut all output till en excel för respektive land.

En bild som visar text, skärmbild, programvara, Multimedieprogram

Automatiskt genererad beskrivning

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

* Funktionerna *read\_data(file,object)*: Läser in excelfilerna och undersöker om dagens datum redan är skrapat, om inte så kallar funktionen på *create\_dataframe(file, object, df)* som används för att skapa en dataframe av all kombinerad skrapad data och sedan skriva till excelfiler.
* Endast ett land i taget körs, genom att köra main-filen och kommentera ut övriga länder. I bilden nedan körs t ex. Sverige.

En bild som visar text, skärmbild, programvara, Multimedieprogram

Automatiskt genererad beskrivning

**FELHANTERING:**

* Koden är uppbygg så att alla listor måste vara av rätt längd för att informationen skall kunna exporteras till excel. Om en tjänst t ex. har en tom prislista så det saknas rader kommer ingen information från någon tjänst exporteras till excelen vid en körning och ett felmeddelande uppstår.

## *Excelfiler:* video\_DK\_2023.xlsx, video\_FI\_2023.xlsx, video\_SE\_2023.xlsx, video\_NO\_2023.xlsx

* Excefilerna innehåller all data från skrapningarna. Det finns en fil per land och alla filer är uppbyggda på samma sätt för respektive land.
* Kolumner:
  + Date: Datum för skrapningen (OBS! Om dagens datum redan finns med läggs inte någon ny data in i excelarket).
  + Package: Namnet på paketet
  + Price: Pris för paketet
  + Campaign: Om det finns någon eventuell kampanj i samband med att skrapningen görs.
  + Information: Information om vad respektive paket innehåller