

Technische handleiding

IPFIT6

Yassir Laaouissi | INF3A | 18-10-2020

Inhoud

[Inleiding 2](#_Toc54610306)

[Main() 3](#_Toc54610307)

[Instagram() 4](#_Toc54610308)

[Marktplaats() 5](#_Toc54610309)

[Twitter() 8](#_Toc54610310)

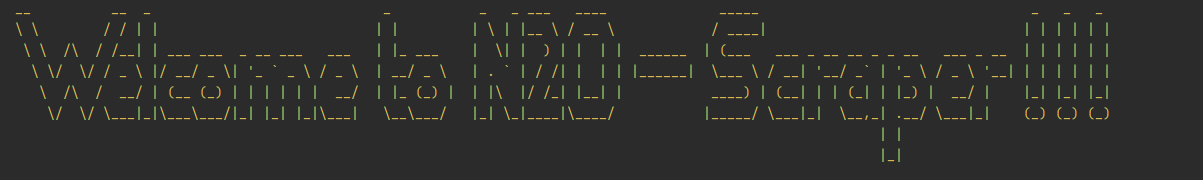
[Google\_Maps() 9](#_Toc54610311)

[Dockerfile en dockercompose 10](#_Toc54610312)

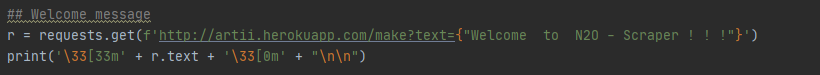
# Inleiding

Dit document omvat de beschrijving van alle functionaliteiten die de tool ontwikkeld in dit project bevat. De tool die in dit project is gerealiseerd is gemaakt ten behoeve van het automatiseren van een gedeelte van het OSINT-onderzoek dat gedaan zal worden. In dit document zijn de hoofdstukken ingedeeld op de functies die zijn gemaakt in het python project van de tool.

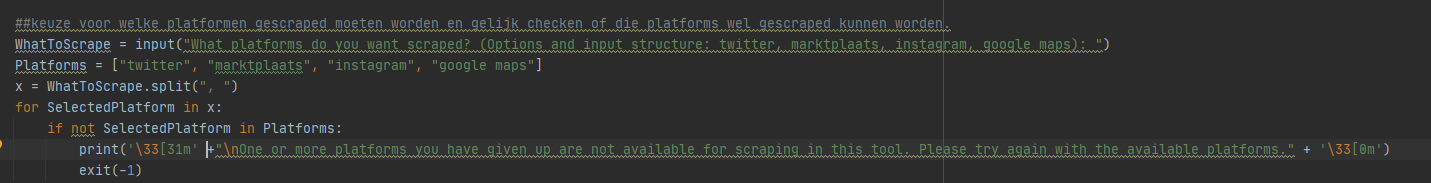
# Main()

Main() is de functie die als eerste wordt aangeroepen. Deze functie kent meerdere verantwoordelijkheden. De eerste is het tonen van de welkomstbanner (afbeelding 1). Deze wordt gerealiseerd middels een http-get request naar een website (afbeelding 2).

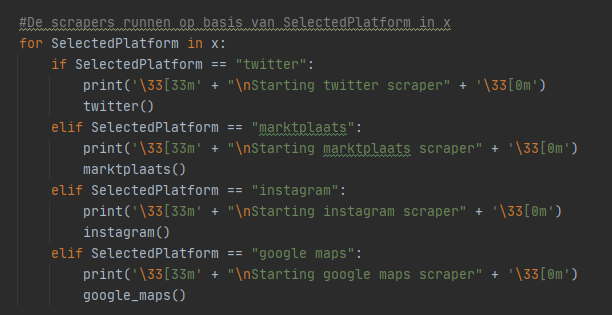
*Afbeelding 1: Welkomstbanner.*



*Afbeelding 2: Code voor de welkomstbanner.*

Nadat de welkomstbanner is getoond op het scherm is het tijd voor een keuzemenu. Het keuzemenu moet duidelijk maken welke platformen gewenst worden gescraped. Daarnaast plaatst het een check om te kijken of deze platformen wel beschikbaar zijn binnen deze tool (afbeelding 3).

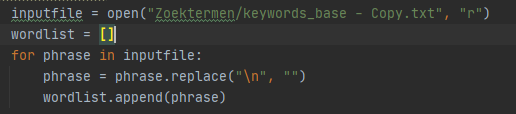
*Afbeelding 3: Code voor het keuzemenu.*

Tot slot loopt de main() door de gekozen platformen en start per gekozen menu de daartoe bestemde functie (afbeelding 4). De functies per platform worden in de volgende hoofdstukken verder behandelt.

*Afbeelding 4: Code voor het loopen door de gekozen platformen.*

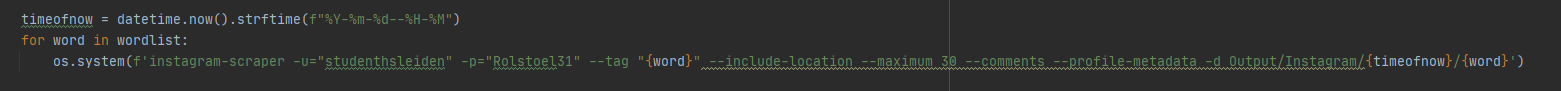
# Instagram()

Dit gedeelte van het script is verantwoordelijk voor het scrapen van Instagram aan de hand van een aantal zoektermen. Deze scraper is gebouwd aan de hand van een bestaand script die te vinden is via: <https://github.com/arc298/instagram-scraper>

De Instagram functie leest eerst alle zoektermen uit en voegt ze toe aan een list, zodat deze in een latere stap makkelijker meegenomen kunnen worden (afbeelding 5).

*Afbeelding 5: Code inlezen zoektermen.*

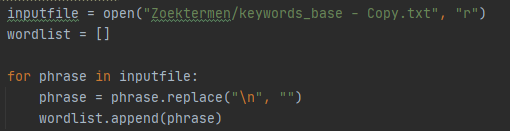
Tot slot wordt een command line commando uitgevoerd met daarin de zoektermen en een variabele genaamd “timeofnow” die worden gebruikt voor het aanmaken van de juiste opslaglocatie (afbeelding 6).

In het onderstaande commando wordt er ingelogd op een instagram account middels -u en -p. Verder wordt gezocht op alle posts die een hashtag bevatten die overeenkomt met de zoektermen. Daarnaast worden locaties, comments en metadata van het profiel van de “poster” meegenomen in de Output. En tot slot zit er ook nog een maximum van laatste 30 posts op als downloadcap.

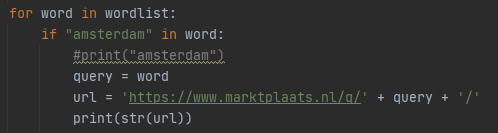
*Afbeelding 6: Code voor het uitvoeren van de instagram scrape.*

# Marktplaats()

Deze functie is verantwoordelijk voor het scrapen van alle marktplaats advertenties die worden gevonden op basis van de zoektermen die opgegeven zijn.

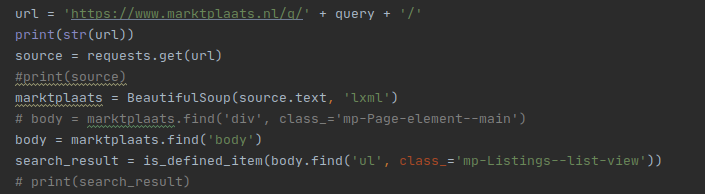
Het eerste wat deze scraper doet is het inlezen en omvormen van de zoektermen naar een gangbaar formaat. Daarnaast zet het deze gangbare zoektermen om in een list (afbeelding 7).

*Afbeelding 7: Code die de zoektermen inleest en omzet tot list.*

Daarna loopt de functie door alle zoektermen heen om te kijken of er amsterdam in voor komt. En vormt aan de hand daarvan een custom query URL (afbeelding 8 en 9). Als er geen amsterdam in de zoekterm voorkomt dan worden variabelen zoals postcode en radius toegevoegd aan de query, zodat er alleen advertenties van 20 km om het centrum van Amsterdam heen worden verzameld. Zit er wel amsterdam in de zoekterm dan zijn deze variabelen niet nodig en wordt er alleen gezocht op de zoekterm.

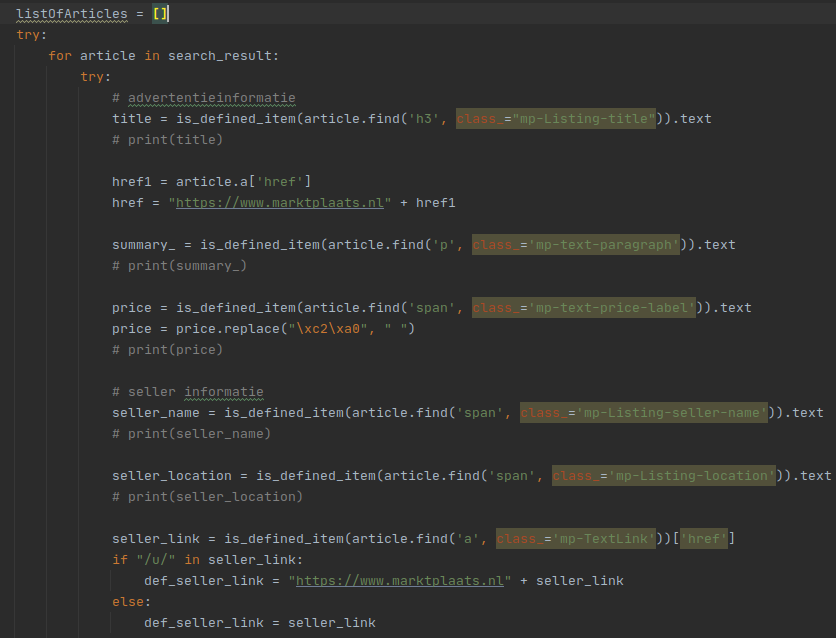
*Afbeelding 8 en 9: De check voor amsterdam in zoekterm en custom URL*

Daarna voer de code voor zowel zoektermen met Amsterdam als zonder een get request uit op de query URL, en zoekt naar het HTML-element waarin de advertenties staan opgeslagen (afbeelding 10).

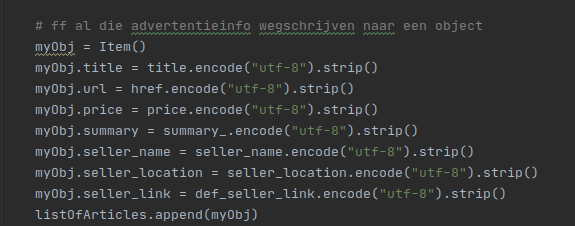


*Afbeelding 10: Get request en inzoomen op mp-listings*

Daarna wordt per advertentie in de mp-listings de link naar de advertentie, de beschrijving, de prijs, de naam van de verkoper, de locatie van de verkoper en de link van het profiel van de verkoper gescraped (zie afbeelding 11). Deze informatie wordt weggeschreven in een object genaamd myobj (zie afbeelding 12).

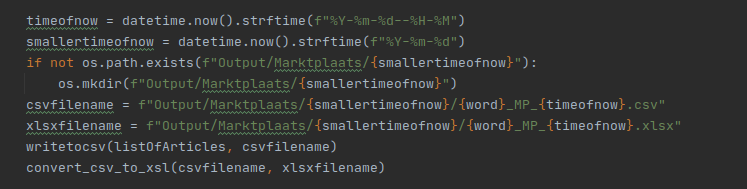


*Afbeelding 11: Verschillende elementen van een advertentie worden gextraheerd.*

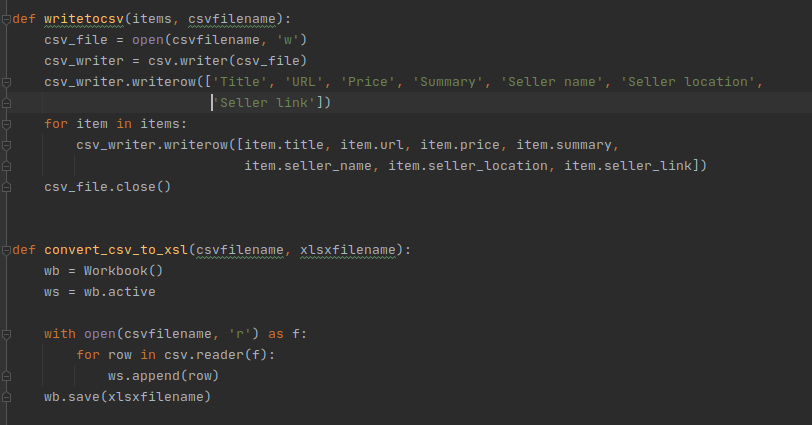


*Afbeelding 12: MyObj()*

Tot slot wordt het object in een CSV en XLSX-bestand opgeslagen (zie afbeelding 13 en 14). Maar daarvoor wordt eerst de juiste mappenstructuur aangemaakt.



*Afbeelding 13: Opslag van MyObj in CSV en XLSX*



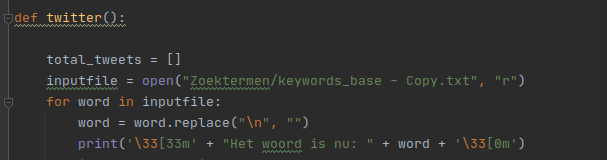
*Afbeelding 14: writetocsv() en convert\_csv\_to\_xls()*

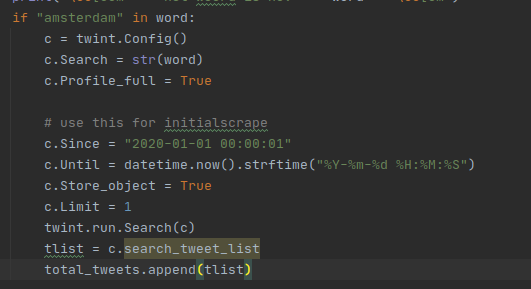
# Twitter()

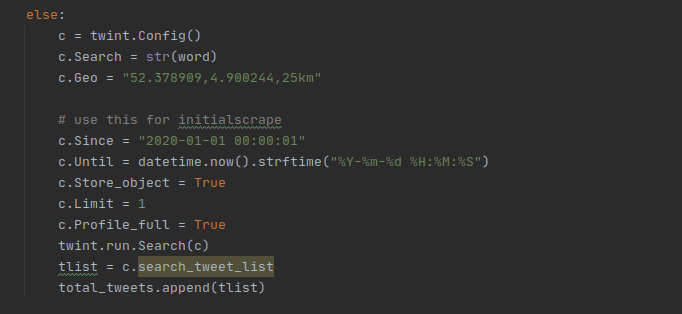
Deze functie staat ervoor garant dat twitter wordt gescraped. Deze functie heeft een gehele tijd goed gefunctioneerd, echter is er vermoedelijk iets aan de kant van twitter verandert waardoor deze functie niet meer naar toebehoren werkt. Echter zal dit gedeelte van de scraper toch beschreven worden, gezien dit wellicht handig is als deze toch werkend gemaakt wordt. Iets wat in de tijd van dit project niet meer zal gebeuren.

Deze functie is gemaakt aan de hand van de volgende github repository: <https://github.com/twintproject/twint>

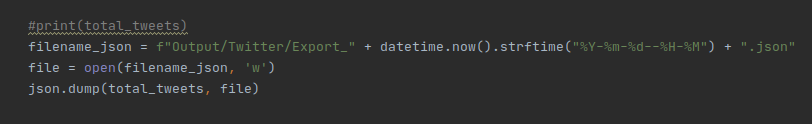
Het eerste wat deze functie doet is het inlezen van de zoektermen en omvormen tot een gangbaar formaat (afbeelding 15). Daarna checkt het of er Amsterdam in de zoekterm zit. Zit er Amsterdam in dan wordt alleen op zoekterm en tijd gefilterd. Zit er geen Amsterdam in de zoekterm dan wordt er gekeken naar tweets binnen een straal van 25km vanaf het centrum van Amsterdam (afbeelding 16 en 17). De tweets worden vervolgens weggeschreven naar een list, die dan weer opgeslagen wordt in een JSON-bestand (zie afbeelding 18).



*Afbeelding 15: zoektermen inladen en omvormen*

*Afbeelding 16: Scrapen met Amsterdam*

*Afbeelding 17: Scrapen zonder Amsterdam*

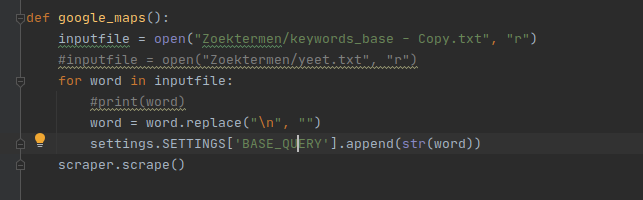


*Afbeelding 18: Opslaan tweets in JSON-formaat.*

# Google\_Maps()

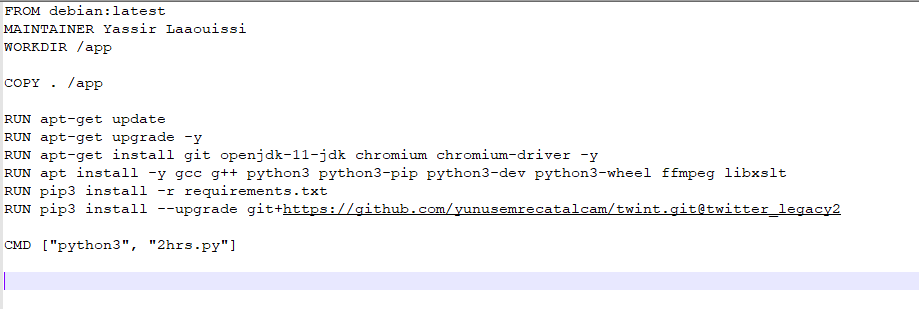
De laatste functie van deze scraper is Google\_Maps(). Zoals de naam suggereert is deze functie bedoelt voor het scrapen van Google Maps. Deze functie is gebasseerd op de github repository: <https://github.com/lyyka/google-maps-businesses-scraper>

Net zoals de andere functies leest deze functie alle zoektermen in en vormt ze om tot een gangbaar formaat. Daarna schrijft hij de zoekterm weg naar een bestand genaamd settings, en op basis daarvan wordt de scraper functie uitgevoerd met de meegegeven settings (afbeelding 19).



*Afbeelding 19: Google maps functie*

# Dockerfile en dockercompose

Docker heeft een Dockerfile nodig zodat de software weet wat er geinstalleerd moet worden in de mini virtuele machine die docker aanmaakt. Daarom is de dockerfile gevuld met de volgende gegevens:

Tot slot is er een docker compose file om ervoor te zorgen dat de docker VM de output van de scraper weg schrijft naar de host machine. De docker compose file ziet er als volgt uit:

