1. **Co to jest? Do czego służy?**  
     
   Skrypt napisany w Pythonie, służy do pobrania **danych rejestrowych** z APIwywiadowni gospodarczej *EMIS* dla wprowadzonej listy identyfikatorów firm. Dozwolone identyfikatory to: NIP, REGON, KRS. Jako wartość domyślną skrypt wykorzystuje NIP. Po pobraniu dane są zapisywane w formacie JSON a następnie parsowane do postaci wymaganej do wykorzystania danych do narzędzia FAIT.
2. **Jak działa?**  
     
   Skrypt pobiera JSONy dla wprowadzonej przez użytkownika listy podmiotów. Pobrane dane są eksportowane do plików JSON, a następnie przetwarzane, zapisywane jak dataframe i następnie eksportowane do formatu xlsx w oczyszczonej postaci. Następnie dane są wykorzystywane przez narzędzie FAIT.

Wybór użytkownika polega na:

1. Zapisania jako zmienną ID\_CLASS typu identyfikatorów wprowadzonych podmiotów, dla których mają zostać pobrane dane finansowe. Dozwolone wartości to: 'NIP', 'REGON', 'KRS'.
2. Zapisania jako zmienną *INPUT\_FILE* ścieżki do pliku w formacie .xlsx (łącznie z nazwą pliku), w którym znajduje się lista NIPów podmiotów, dla których pobrane mają zostać dane podmiotów,
3. Zapisania jako zmienną *JSON\_OUTPUT\_PATH* ścieżki do katalogu, do którego zwrócone mają zostać pobrane pliki w formacie JSON.
4. Zapisania jako zmienną *XLSX\_OUTPUT\_PATH* ścieżki do pliku (łącznie z nazwą pliku),,do którego zwrócone mają zostać sparsowane pliki w formacie .xlsx.

W pierwszej kolejności skrypt pobiera klucz sesji (za pomocą funkcji **get\_session\_id)**, który składa request o podanie unikalnego klucza sesji. Skrypt jest pobierany i następnie wykorzystywany do autentykacji użytkownika podczas pobierania danych z API.

W dalszej kolejności skrypt odczytuje wprowadzoną przez użytkownika listę NIP sprawdzanych podmiotów (za pomocą funkcji **get\_nip\_name**) i zapisuje je jako listę. Lista wprowadzonych NIP jest walidowana oraz deduplikowana. Dodatkowo tworzony jest log, przechowujący informacje na temat poprawnych i niepoprawnych NIP.

Następnie skrypt za pomocą funkcji **get\_isic\_list** pobiera dla każdego z poprawnych NIP zwróconych przez funkcję get\_nip\_nape do listy *nip\_list* kod ISIC. Kod ISIC (EMIS\_ID) to unikalny identyfikator podmiotu w bazie danych EMIS. W przypadku niepowodzenia (firmy nie ma w bazie EMIS) NIP podmiotu, dla którego nie udało się pobrać kodu ISIC jest zapisywany w oddzielnym słowniku wraz z odpowiednią adnotacją, po czym funkcja kontynuuje pobieranie kodów. Funkcja zwraca 2 słowniki – *nips\_isics* oraz *nips\_errors* – przechowujące odpowiednio informacje na temat udanych i nieudanych pobrań kodu ISIC.

Następnie skrypt za pomocą funkcji **get\_company\_info** pobiera JSONy dla każdego kodu ISIC. Funkcja pomocą funkcji **request\_data**, która tworzy odpowiedni request dla zadanego ISIC zmapowanego z NIP. Funkcja wykorzystuje pobrane w poprzednim kroku klucze ISIC i w pętli pobiera dla nich dane udostępnione w bazie EMIS, odczytuje, rozkodowuje i zapisuje je w formacie JSON oraz eksportuje do wskazanego folderu. Eksportowane pliki zawierają odpowiedni NIP firmy.

W przypadku powodzenia pobrania dane są zapisywane we wskazanym folderze outputowym z odpowiednim guidem (nazwą pliku), który zawiera NIP podmiotu, oraz guid *EMIS\_REGISTER*. Nie każde zapytanie musi zakończyć się sukcesem, dlatego w skrypcie umieszczona została klauzula try/except. W przypadku niepowodzenia skrypt zapisuje dany kod ISIC i zwracany kod błędu. W przypadku niepowodzenia plik zostaje zapisany i wyeksportowany do folderu outputowego z odpowiednim guidem (dodana fraza *INVALID)*, który wskazuje na to, że pobranie danych jest nieudane.

Funkcja zwraca 2 słowniki przechowujące odpowiednio informacje na temat udanych i nieudanych zapytań: *nips\_jsons* oraz *nips errors*, który jest updatowany o nieudane pobrania danych.

Po zakończeniu pobierania danych zostają one przekazane jako input do funkcji **parse\_company\_info**. Funkcja odczytuje pobrane dane, i parsuje z nich wszystkie pobrane dane rejestrowe. Prawidłowo pobrane i rozparsowane dane rejestrowe są oczyszczane ze zbędnych kolumn, następuje zmiana nazw kolumn na bardziej czytelne. Zbędne dane są usuwane. Pola zawierające informacje na temat reprezentacji podmiotów (Executives), udziałowców (Shareholders), kodów fiskalnych i rejestrowych (Eternal IDs) oraz klasyfikacji działalności (NAICS) są przekształcane z niestałej struktury na pojedyncze pola. Rekordy, których nie udało się rozparsować są odkładane i zapisywane z odpowiednim opisem błędu.

Następnie za pomocą funkcji **create\_register\_output**, tworzone są tabele końcowe.Prawidłowo sparsowane rekordy są łączone w jedną tabelę i następnie zapisywane w formacie DataFrame. Nieprawidłowo pobrane/sparsowane dane także są łączone w tabelę DataFrame, która jest łączona z poprawnie pobranymi danymi, przez co tworzony jest ujednolicony plik outputowy.

Następnie za pomocą funkcji **export\_output** dane są w formacie .xslx do wskazanego folderu outputowego. Tworzone są zakładki zawierające odpowiednio:

1. FullOutput – całkowity output, który udało się poprawnie pobrać i sparsować oraz listę nipów (i EMIS\_ID jeśli są dla niego dostępne informacje w EMIS),
2. ValidOutput – tylko pobranie pobrane i sparsowane rekordy
3. InvalidOutput - tylko niepoprawnie pobrane i sparsowane rekordy, w celu łatwiejszej identyfikacji braku / niepoprawnych danych.
4. EMIS\_input - oprócz eksportu całości sparsowanych danych tworzona jest tabela zawierająca jedynie dane na temat NIP oraz kodów ISIC podmiotów, która jest wykorzystywana jako jedno ze źródeł danych podczas realizacji FAITa.

Z pobrania tworzony jest log, który w konsoli wyświetla informacje na temat pobierania danych.

1. **Uwagi. Możliwości poprawy/rozwoju.**

- dodanie funkcjonalności eksportowania przetworzonych danych do bazy danych,

- poprawa parsowania pól Executives, Shareholders, Externald IDs i NAICS. Na ten moment część parsująca te pola jest napisana w bardzo prosty i niewydajny sposób, każde pole jest parsowane oddzielną funkcją – warto rozważyć przetwarzanie danych np. za pomocą funkcji json\_normalize.