Laboratorium 5 – Wyrażenia regularne, interfejsy linii komend.

Cele dydaktyczne

1. Przetwarzanie wyrażeń regularnych.
2. Zapoznanie z wybranymi modułami biblioteki standardowej Pythona.
3. Tworzenie tekstowych interfejsów użytkownika.

Program można zgłosić jako zrobiony, jeśli spełnione są poniższe warunki:

1. Program jest zgodny z podaną specyfikacją.
2. Został przetestowany.
3. Student go rozumie i potrafi wyjaśnić.

UWAGA:

Wykonywanie zadań przy użyciu AI będzie traktowane jako praca niesamodzielna i będzie skutkować oceną niedostateczną.

- Zadbaj o to, aby każda funkcja w programie miała tylko jedną odpowiedzialność.
- Zadbaj o rozdzielenie funkcji przetwarzających dane od funkcji najwyższego poziomu wypisujących tekst na wyjście standardowe.
- W przypadku, gdy kilka funkcjonalności wymaga skorzystania z samych funkcji, umieść je w osobnym module, który będzie ponownie użyty.
- Przed potencjalnie źle sformatowanymi danymi zabezpiecz się wykorzystując mechanizm wyjątków.

- **z wykorzystaniem *mechanizmu obsługi wyjątków* zadbaj o poprawność działania funkcji**
- **należy zabezpieczyć funkcje przed błędnymi danymi wejściowymi i wyjątkami. (np. dzieleniem przez zero).**
- **do zadań przygotować 3 – 5 testów sprawdzających poprawność działania**
- **do prowadzącego wysłać pliki z kodem źródłowym oraz rzuty ekranu z przygotowany testami**

Wprowadzenie

W ramach niniejszego laboratorium przedmiotem analizy będą pomiary ze stacji pomiarowych zanieczyszczeń powietrza zbierane i udostępniane przez bank danych pomiarowych Głównego Inspektoratu Ochrony Środowiska.

Dane umieszczono pod tym linkiem, uwzględniając:

- plik stacje.csv – zawierający opis metadanych stanowisk pomiarowych.
- pliki z pomiarami: measurements/*.csv zawierające pomiary, nazwy pliku zgodnie z szablonem <rok>_<mierzona wielkość>_<częstotliwość>.csv

Pliki zawierają różne formaty: teksty, daty, liczby (w formacie polskim z przecinkiem).

Zadania

1. Z wykorzystaniem modułu csv, napisz funkcję parsujące dane w plikach z pomiarem oraz z pliku zawierającego metadane. Jako wynik, wykorzystaj dowolną, samodzielnie wybraną strukturę.
2. Z wykorzystaniem wyrażeń regularnych, napisz funkcję o nazwie `group_measurement_files_by_key`, która:
    a. Przyjmuje jeden argument:
        i. path – obiekt typu Path wskazujący katalog z plikami.
    b. Zwraca słownik, w którym:
        i. kluczem jest trójka: (rok, wielkość, częstotliwość) – każda z tych wartości to napis (łańcuch znaków),
        ii. wartością jest ścieżka do pliku (Path), którego nazwa pasuje do danej grupy.
    c. Funkcja powinna przeszukać wszystkie pliki w podanym katalogu (nie zagląda do podkatalogów), i w przypadku, gdy jego nazwa jest zgodna z podanym wzorcem, uwzględnić go w wartości zwracanej.
3. Z wykorzystaniem wyrażeń regularnych, napisz funkcję o nazwie get_addresses, która:
    a. Przyjmuje dwa argumenty:
        i. path – obiekt pliku Path wskazujący na plik *.csv z metadanymi
        ii. city – ciąg znaków reprezentujący miejscowość
    b. Zwraca listę czwórek postaci (województwo, miasto, ulica, opcjonalnie: numer), zawierającą adresy stacji w podanej miejscowości.
4. Z wykorzystaniem wyrażeń regularnych rozwiąż następujące zadania na pliku stations.csv:
    a. Wyodrębnij wszystkie daty w formacie RRRR-MM-DD (Z danych w kolumnach "Data uruchomienia" i "Data zamknięcia").

b. Wyciągnij listę szerokości i długości geograficznej z rekordów (np. 50.943245, 14.913327) – liczba dziesiętna z 6 cyframi po kropce.
c. Znajdź stacje o nazwach składających się z dwóch części (zawierających myślnik) np. Bogatynia - Chopina, Bielawa - ul. Grota.
d. Zamień w nazwach stacji
    i. spacje na symbol podłogi (“ “ → “_”)
    ii. polskie znaki diakrytyczne na ich odpowiedniki będące literami alfabetu łacińskiego (‘ą” → ‘a’, ‘ć’ → ‘c’, itd.)
e. Zweryfikuj, czy wszystkie stacje, których kod kończy się na „MOB” mają rodzaj stacji określony jako ‘mobilna’.
f. Wyodrębnij lokalizacje złożone z 3 członów rozdzielonych myślnikiem.
g. Znajdź lokalizacje zawierające przecinek i nazwę ulicy (ul.) lub alei (al.).

5. Korzystając z modułu argparse, zaprojektuj i skonstruuj przyjazny użytkownikowi CLI (ang. *command-line interface* – interfejs linii komend), który:
    a. z wykorzystaniem argumentów:
        i. pozwoli przekazać użytkownikowi, mierzoną wielkość (PM2.5, PM10, NO, …), częstotliwość (1g/24g), przedział czasowy (początek i koniec w formacie rrrr-mm-dd)
    b. z wykorzystaniem podkomend, pozwoli na uruchomienie następujących funkcjonalności:
        i. wypisanie nazwy i adresu losowej stacji, która w zadanym przedziale czasowym mierzy tę wielkość.
        ii. obliczenie średniej i odchylenia standardowego danej wielkości w zadanym przedziale czasowym dla danej stacji jako argument podkomendy.
6. Do programu z CLI (zad 5.) dołącz moduł logging. Skonfiguruj moduł logging, tak, aby:
    a. Po przeczytaniu każdego z danymi wiersza logować na poziomie DEBUG liczbę przeczytanych bajtów.
    b. Po otwarciu i zamknięciu każdego pliku logować odpowiednią informację w trybie INFO.
    c. Zaloguj w trybie WARNING sytuacje, które nie przerywają działania programu, ale mogą świadczyć o nieprawidłowych danych wejściowych, np.:
        i. Użytkownik podał mierzoną wielkość, która nie występuje na żadnej stacji.
        ii. Brak dostępnych pomiarów dla zadanych parametrów (np. filtr zwrócił pustą listę).
        iii. Częstotliwość lub wielkość nie jest wspierana przez daną stację.
    d. Zaloguj w trybie ERROR błędy krytyczne, które uniemożliwiają dalsze działanie programu, np. próba do odwołanai do pliku, który nie istnieje.
    e. Logi na poziomach DEBUG, INFO, WARNING muszą być wypisywane na stdout, natomiast ERROR i CRITICAL na stderr (wyjście błędu).

    Punkty: 8

Zadania rozszerzające (na max pkt. – dla ambitnych)

1. Przejrzyj narzędzia takie jak typer, click, docopt. Wybierz jedno z nich. Skonstruuj kolejny interfejs linii komend z wykorzystaniem wybranego narzędzia, według tego samego projektu, co CLI z zadania 5. Porównaj i przeanalizuj oba interfejsy.
2. Napisz funkcję analizującą dane pomiarowe w celu wykrycia **anomalii**, które mogą świadczyć o błędach pomiarowych, awarii czujnika albo zjawiskach nietypowych. Twoja funkcja powinna:
    a. przyjąć listę pomiarów (czas, wartość, stacja, wielkość),
    b. zaproponować sensowne reguły stanowiące anomalie, np. wykrywać podejrzane serie pomiarów — np.:
        i. zbyt częste skoki wartości (delta między kolejnymi przekracza próg),
        ii. zbyt wiele wartości zerowych / None / ujemnych (czujnik może być uszkodzony),
        iii. nagłe skoki powyżej progów alarmowych (np. PM10 > 500).

    Zapewij podkomendę w CLI umożliwiającą uruchomienie tej funkcji.