Laboratorium 3 – Podstawowe struktury danych Pythona.

Cele dydaktyczne

Zapoznanie z podstawowymi strukturami danych Pythona: listami, krotkami, słownikami.

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 czatu GPT będzie traktowane jako praca niesamodzielna i będzie skutkować oceną niedostateczną.

Do zadań, gdzie jest to możliwe, przygotować 3-5 testów sprawdzających poprawność działania(pamiętaj o walidacji danych)

Zadania

1. Przygotowanie i czytanie danych

Logi HTTP to pliki lub zapisy zawierające informacje o żądaniach i odpowiedziach przesyłanych między klientem a serwerem w protokole HTTP. Zawierają one dane takie jak adres IP użytkownika, metoda żądania, adres URL, kod odpowiedzi serwera oraz znacznik czasu, co pozwala na analizę ruchu sieciowego i diagnostykę problemów.

Pobierz plik z logami serwera webowego: http_first_100k.log.

https://drive.google.com/drive/folders/1tNfn31un5u4D0Ep15Ezy7fCD6bTZWx6f?usp=sharing

Zapoznaj się ze strukturą pliku, która opisana jest tutaj: https://www.secrepo.com/Datasets%20Description/Network/http.html
W szczególności, zapoznaj się z polami (pozostałe możesz ingorować):

- **ts** – Znacznik czasu żądania w formacie UNIX timestamp (sekundy od 1970-01-01).
- **uid** – Unikalny identyfikator sesji (np. dla konkretnego połączenia).
- **id.orig_h** – Adres IP hosta wysyłającego żądanie (klienta).

- **id.orig_p** – Port źródłowy hosta wysyłającego żądanie.
- **id.resp_h** – Adres IP serwera, do którego skierowano żądanie.
- **id.resp_p** – Port docelowy serwera, zwykle 80 (HTTP) lub 443 (HTTPS).
- **method** – Metoda HTTP (GET, POST, HEAD, PUT, DELETE, OPTIONS, itp.).
- **host** – Nazwa domenowa hosta serwera, do którego wysyłane jest żądanie.
- **uri** – Ścieżka URI zasobu, który jest żądany (np. /index.html).

Przygotuj skrypt pozwalający odczytywać logi HTTP z wejścia standardowego.

2. Listy i krotki

a. Utwórz funkcję `read_log`, która czyta wszystkie linie z wejścia standardowego i zwraca **listę** zawierającą poszczególne wpisy. Funkcja powinna:
   i. Podzielić każdą linię (wpis) na niezależne elementy.
   ii. Dokonać konwersji ciągów znaków na odpowiednie typy danych wbudowane oraz biblioteki standardowej (np. kod statusu – int, data i znacznik czasu żądania – datetime.datetime.
   iii. Uwzględnić i reprezentować co najmniej atrybuty opisane powyżej,
   iv. Zapisać każdy wpis (niepustą linię) jako krotkę na liście.
   v. Zwrócić listę krotek.

b. Napisz funkcję `sort_log(log,index)`, która:
   i. przyjmuje dwa parametry:
      1. log – lista krotek reprezentującą wpis oraz
      2. index – liczba określająca element krotki, według którego zostanie wykonane sortowanie.
   ii. Zaimplementuj mechanizm sortowania listy korzystając z funkcji `sorted()` lub `sort()`.
   iii. Z wykorzystaniem mechanizmu obsługi wyjątków zadbaj o poprawność działania funkcji (np. gdy index przekroczy rozmiar krotki).

c. Napisz funkcję `get_entries_by_addr`, która:
   i. przyjmuje jako parametr listę krotek reprezentującą log,
   ii. przyjmuje jako parametr ciąg znaków reprezentujący adres IP lub nazwę domenową hosta wykonującego żądanie,
   iii. waliduje podany adres ip,
   iv. zwraca listę wpisów z danym ip.

d. Napisz funkcję `get_entries_by_code`, która:
   i. przyjmuje jako parametr listę krotek reprezentującą log,
   ii. przyjmuje jako parametr kod statusu HTTP (np. 200),
   iii. waliduje podany kod statusu,
   iv. zwraca listę wpisów z danym kodem statusu.

e. Napisz funkcję `get_failed_reads`, która:
   i. przyjmuje jako parametr listę krotek reprezentującą log
   ii. tworzy oddzielne listy zawierające wpisy z kodami statusu HTTP 4xx oraz 5xx.
   iii. przyjmuje opcjonalny parametr logiczny, który określa, czy zwrócić jedną, połączoną

listę czy osobne,
   iv. opcjonalnie, łączy listy w jedną
   v. zwraca odpowiedni wynik

f. Napisz funkcję `get_entires_by_extension`, która:
   i. przyjmuje jako parametr listę krotek reprezentującą log,
   ii. przyjmuje jako parametr ciąg znaków reprezentujący rozszerzenie pliku (np. “jpg”)
   iii. zwraca wszystkie wpisy zawierające zapytania o zasoby z danym rozszerzeniem.

3. Słowniki

a. Utwórz funkcję `entry_to_dict`, która:
   i. przyjmuje na wejściu krotkę reprezentującą pojedynczy wpis
   ii. tworzy słownik, w którym kluczem jest ciąg znaków reprezentujący znaczenie kolejnych pól wpisu, np. “ip”, “status code”, “ts”, etc. Wartością w słowniku jest wartość danego pola.
   iii. zwraca słownikową reprezentację pojedynczego wpisu

b. Utwórz funkcję `log_to_dict`, która:
   i. przyjmuje na wejściu listę krotek reprezentującą cały log,
   ii. tworzy słownik, w którym kluczem jest uid reprezentujący sesję. Wartością w tym słowniku powinna być lista słowników w postaci zwracanej przez funkcję entry_to_dict.
   iii. zwraca słownik zawierający wszystkie wpisy z listy – słownikowa reprezentacja logu

c. Utwórz funkcję `print_dict_entry_dates`, która:
   i. przyjmuje na wejściu słownikową reprezentację logu
   ii. iteruje po elementach słownika
   iii. wyświetla na wyjściu standardowym w czytelny sposób:
      1. adresy ip/nazwy domenowe hostów w sesji,
      2. liczbę żądań wykonanych przez danego hosta,
      3. datę pierwszego i ostatniego żądania wykonanego przez hosta,
      4. procentowy udział żądań danej metody np. (GET – 30 %, POST – 70%)
      5. stosunek liczby żądań z kodem 2xx do liczby wszystkich żądań.

Materiały dodatkowe

1. Alex Martelli, Anna Martelli Ravenscroft, Steve Holden, Paul McGuire, Python in a Nutshell, 4th Edition, Published by O'Reilly Media, Inc., Rozdział 3 [URL: https://learning.oreilly.com/library/view/python-in-a/9781098113544/]
2. https://docs.python.org/3/tutorial/datastructures.html