Laboratorium 2 – Podstawy Pythona. Praca z funkcjami. Wejście/wyjście standardowe

Języki skryptowe

Cele dydaktyczne

1. Zapoznanie z podstawowymi metodami budowania abstrakcji w Python
2. Zapoznanie z przetwarzaniem danych z wejścia standardowego

Zadania

1. Przygotowanie danych

Pobierz kilka plików w formacie TXT z treścią książek ze strony WolneLektury.pl.
Przykładowo:

- Charles Dickens – Opowieść Wigilijna
- Hans Christian Andersen – Calineczka
- Hans Christian Andersen – Brzydkie Kaczątko
- Maria Konopnicka – O Krasnoludkach i Sierotce Marysi

Przeanalizuj zawartość plików. Lektury są w format składającym się z trzech części:

- **Preambuła**
  - Zawiera tytuł, autora, tłumacza, numer ISBN (opcjonalnie).
  - Kończy się co najmniej dwoma pustymi wierszami.
  - Jeśli w pierwszych 10 wierszach nie występują dwa kolejne puste wiersze, należy założyć, że preambuły nie ma.
- **Treść**
  - Składa się z **akapitów** rozdzielonych pustymi wierszami.
  - Każdy akapit zawiera **wiele zdań**, zakończonych **kropką, wykrzyknikiem lub znakiem zapytania**.

    - Koniec wiersza NIE oznacza końca zdania.
    - **Koniec akapitu (pusty wiersz) wymusza koniec zdania**, nawet jeśli nie ma znaku interpunkcyjnego.
- **Informacja o wydaniu**
    - Zaczyna się po wierszu zawierającym pięć dywizów (`-----`).
    - Wszystkie dalsze linie należy **ignorować**.

2. Czytanie danych z wejścia standardowego

Wejście standardowe (**stdin**) to domyślny strumień, z którego program odczytuje dane wprowadzane przez użytkownika lub przekazywane przez potok.

- Napisz program, który **wypisuje tylko treść książki**, ignorując preambułę i informację o wydaniu.
- Program musi działać potokowo, usuwając **zbędne spacje wewnątrz linii** i **oczyszczając białe znaki na początku i końcu** każdej linii. Po odczytaniu EOF, niech zakończy działanie.
- Struktura akapitów (puste linie) **musi zostać zachowana**.

Przykład wywołania:

```
python lab_2.py < calineczka.txt
```

3. Funkcje

Rozszerz program o poniższe funkcjonalności, przestrzegając następujących zasad:

- Każda funkcjonalność musi być **zaimplementowana jako osobny moduł** (plik). Moduł powinien oczekiwać danych na wejściu standardowym (stdin), a wynik jego działania powinien być komunikowany na wyjście standardowe (stdout).
- Każda funkcja powinna mieć **jedną odpowiedzialność**.
- Funkcje przetwarzające dane **powinny być oddzielone** od funkcji wypisujących tekst na wyjście standardowe.
- Wspólne funkcje powinny być umieszczone w **osobnym module**, by można było je **ponownie używać**.
- **Obsłuż potencjalne błędy** (np. brak wymaganych znaków w pliku) za pomocą

mechanizmu wyjątków.
- Funkcje powinny iterować znak po znaku lub przetwarzać dane w locie. Na tej liście wyjątkowo **nie wolno używać nieskalarnych typów danych** (listy, słowniki, zbiory – **dozwolony jest tylko `str`**).

Funkcje redukujące:

a. Funkcja zliczająca akapity w tekście (akapit jest oddzielony pustą linią).
b. Funkcja zliczająca wszystkie znaki w tekście, z pominięciem białych znaków.
c. Funkcja licząca procent zdań, które zawierają przynajmniej jedną nazwę własną (niech nazwą własną będzie każdy wyraz napisany wielką literą, nie będący pierwszym wyrazem w zdaniu).

Funkcje wyszukujące

d. Funkcja wypisująca najdłuższe zdanie w książce (kryterium – liczba znaków).
e. Funkcja wyszukująca najdłuższe zdanie, w którym żadne dwa sąsiadujące słowa nie zaczynają się na tę samą literę
f. Funkcja wyszukująca pierwsze zdanie, które ma więcej niż jedno zdanie podrzędne (na podstawie przecinków).

Funkcje filtrujące:

g. Funkcja wypisująca tylko zdania zawierające co najwyżej 4 wyrazy.
h. Funkcja, która wypisuje na wyjściu tylko zdania, które są pytaniami lub wykrzyknieniami.
i. Funkcja wypisująca pierwszych 20 zdań.
j. Funkcja wypisująca tylko zdania, które zawierają co najmniej dwa wyrazy z następujących: „i”, „oraz”, „ale”, „że”, „lub”.
k. Funkcja, która wypisuje na wyjściu tylko zdania w czwartym kwartylu pod względem długości zdania (kryterium – liczba znaków).
l. Funkcja wypisująca tylko te zdania, których wyrazy są w porządku leksykograficznym.

4. `__main__`

Zapoznaj się z rozdziałem dokumentacji https://docs.python.org/3/library/__main__.html

Zmodyfikuj programy w taki sposób, aby każde wywołanie skryptu z terminalu odbywało się przy użyciu konstrukcji `if __name__ == '__main__':`.

5. Przesyłanie potokowe

Pokaż, że opracowane przez Ciebie funkcje umożliwiają na potokowe przesyłanie danych pomiędzy sobą. Na przykład, w celu wypisania ile akapitów rozciąga się na pierwsze 20 zdań tekstu, wywołanie może wyglądać następująco:

```
cat calineczka.txt | python first_20_sentences.py | python count_paragraphs.py
```

(Uwaga, polecenie powyżej nie będzie działać w CMD na windows ze względu na brak polecenia `cat` w tej powłoce. W przypadku korzystania z CMD należy użyć np. `type`.

Materiały dodatkowe

1. https://www.digitalocean.com/community/tutorials/read-stdin-python
2. 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/]
3. Ryan's tutorial, Piping & Redirection
   [URL: https://ryanstutorials.net/linuxtutorial/piping.php]