LABORATORIUM 2

TREŚCI KSZTAŁCENIA:

- PROGRAMOWANIE FUNKCYJNE W PYTHONIE, PRACA ZE STRUKTURAMI DANYCH, MAPOWANIE I FILTROWANIE Z WYKORZYSTANIEM LIST SKŁADANYCH, FUNKCJA MAP(), REDUCE(), FILTER(), EVAL(), EXEC()

Spis treści

Programowanie funkcyjne

Programowanie funkcyjne to paradygmat programowania, który koncentruje się na funkcjach i ich kompozycji, traktując funkcje jako "pierwszorzędne obiekty" (mogą być przekazywane jako argumenty, zwracane z innych funkcji itp.). Kluczowe cechy programowania funkcyjnego obejmują niemutowalność (brak zmiany stanu), brak efektów ubocznych oraz stosowanie funkcji czystych.

Praca ze Strukturami Danych

W programowaniu funkcyjnym pracujemy z niezmiennymi strukturami danych (np. krotki, listy, słowniki). Użycie funkcyjnych podejść, takich jak listy składane, map(), filter(), i reduce(), pozwala na efektywne przekształcanie i filtrowanie danych.

Mapowanie i Filtrowanie z Listami Składanymi

Listy składane (list comprehensions) to uproszczony sposób tworzenia nowych list na podstawie istniejących iterowalnych obiektów w zwięzły i czytelny sposób.

```
# Przykład: podniesienie do kwadratu liczb parzystych z listy
numbers = [1, 2, 3, 4, 5, 6]
squares_of_even = [x ** 2 for x in numbers if x % 2 == 0]
print(squares_of_even)  # Wyjście: [4, 16, 36]
```

Funkcje map(), filter(), i reduce()

- map(function, iterable): Zastosowanie funkcji do każdego elementu iterowalnego obiektu

```
# Przykład: podwojenie każdej liczby
doubled = list(map(lambda x: x * 2, numbers))
print(doubled)  # Wyjście: [2, 4, 6, 8, 10, 12]
```

- filter(function, iterable): Filtrowanie elementów iterowalnego obiektu na podstawie funkcji zwracającej wartość logiczną.

```
# Przykład: wybór liczb parzystych
evens = list(filter(lambda x: x % 2 == 0, numbers))
print(evens)  # Wyjście: [2, 4, 6]
```

- reduce(function, iterable): Redukowanie iterowalnego obiektu do pojedynczej wartości przez kolejne zastosowanie funkcji (dostępne w module functools).

```
from functools import reduce

# Przykład: obliczenie sumy liczb
total = reduce(lambda x, y: x + y, numbers)
print(total)  # Wyjście: 21
```

Funkcje eval() i exec()

- eval(expression): Oblicza wyrażenie przekazane jako string i zwraca jego wynik. Stosowane głównie do prostych obliczeń, choć niesie ryzyko bezpieczeństwa.

```
expr = "3 + 5 * 2"
result = eval(expr)
print(result)  # Wyjście: 13
```

- exec(code): Wykonuje kod Python przekazany jako string. Umożliwia dynamiczne wykonywanie skryptów, ale jest bardziej niebezpieczne niż eval() i powinno być używane ostrożnie.

```
code = """
for i in range(3):
    print(i)
"""
exec(code)
# Wyjście:
# 0
# 1
# 2
```

Zadania do samodzielnego rozwiązania

Rozwiązania w postaci niezbędnych plików źródłowych należy przesłać do utworzonego zadania na platformie e-learningowej zgodnie ze zdefiniowanymi instrukcjami oraz w nieprzekraczalnym wyznaczonym terminie.

Zadanie 1. Analiza Tekstu i Transformacje Funkcyjne

Napisz program, który przyjmuje długi tekst (np. artykuł, książkę) i wykonuje kilka złożonych operacji analizy tekstu:

- Zlicza liczbę słów, zdań, i akapitów w tekście.
- Wyszukuje najczęściej występujące słowa, wykluczając tzw. stop words (np. "i", "a", "the").
- Transformuje wszystkie wyrazy rozpoczynające się na literę "a" lub "A" do ich odwrotności (np. "apple" → "elppa").

Wskazówka: Użyj map(), filter(), i list składanych, aby przeprowadzać transformacje na tekście.

Zadanie 2: Walidacja i Przekształcenia Operacji na Macierzach

Stwórz system, który przyjmuje operacje na macierzach jako stringi i wykonuje je dynamicznie, zapewniając jednocześnie walidację poprawności operacji:

- Operacje mogą obejmować dodawanie, mnożenie i transponowanie macierzy.
- System powinien sprawdzać poprawność operacji (zgodność wymiarów) i zwracać wynik w postaci macierzy.
- Wykorzystaj eval() i exec() do wykonywania operacji na macierzach, a także funkcje walidacyjne, które sprawdzają poprawność przed wykonaniem.

Wskazówka: Zaimplementuj walidację operacji i użyj funkcji funkcyjnych do przekształceń i obliczeń na macierzach.

Zadanie 3. Dynamiczne Wyznaczanie Ekstremów w Niejednorodnych Danych

Napisz funkcję, która przyjmuje listę niejednorodnych danych (np. liczby, napisy, krotki, listy, słowniki) i wykonuje dynamiczną analizę danych, aby:

- Zwrócić największą liczbę (lub wartość numeryczną) w danych.
- Zwrócić najdłuższy napis.
- Zwrócić krotkę o największej liczbie elementów.

Wskazówka: Użyj filter() do selekcji odpowiednich typów danych oraz map() do przekształceń na elementach.

Zadanie 4 Implementacja Złożonej Funkcji Matrycowej z Użyciem reduce()

Napisz program, który przyjmuje listę macierzy i łączy je w jedną za pomocą operacji zdefiniowanej przez użytkownika (np. suma macierzy, iloczyn), korzystając z reduce(). Program powinien:

- Dynamicznie wywoływać różne operacje (np. sumowanie, mnożenie) na macierzach.
- Umożliwiać definiowanie niestandardowych operacji przez użytkownika.

Wskazówka: Wykorzystaj reduce() do łączenia macierzy oraz eval() do dynamicznej definicji operacji.

Zadanie 5. System Dynamicznego Generowania Kodów Python (Metaprogramowanie)

Napisz narzędzie, które generuje dynamicznie kod w Pythonie na podstawie szablonów i danych wejściowych, a następnie uruchamia ten kod. Narzędzie powinno:

- Przyjmować szablon kodu jako string (np. def funkcja(x): return x + 2).
- Uzupełniać szablon o brakujące fragmenty kodu (np. zmienne, funkcje) w czasie działania.
- Weryfikować poprawność generowanego kodu przed uruchomieniem.

Wskazówka: Wykorzystaj eval() i exec() w połączeniu z walidacją wejściową i generowaniem kodu z szablonów.