Laboratorium 1

Analiza Danych Pomiarowych
Ćwiczenia Laboratoryjne
Inżynieria Biomedyczna 2019/2020

Zagadnienia

Zagadnienia do samodzielnego opracowania:

- Język Python: podstawy.
- Definiowanie i wywoływanie funkcji.
- Struktury danych: listy, krotki, słowniki, zbiory.
- Instrukcje warunkowe: if, elif, else, ternary operator.
- Instrukcje sterujące i pętle: for, while, break, continue.
- Wyrażenia lambda.
- List/dictionary comprehension.

Zadania

Zadanie 1.
Zdefiniuj w dowolny sposób listę i w momencie deklaracji dodaj do niej wartości liczbowe: 1, 5, 8. Następnie dodaj do niej element o wartości 17. Następnie przemóż wszystkie elementy listy przez 5. Na koniec usuń z listy najmniejszą wartość i wypisz listę.
Przewidywany rezultat:

```
>>> [25, 40, 85]
```

Zadanie 2.
Zdefiniuj funkcje **multiply_lists** przyjmującej dwa parametry (domyślnie traktowane jako listy). Zwróć, bez modyfikowania oryginalnych list, trzecią listę zawierającą wzajemnie przemnożone przez siebie korespondujące elementy. Załóż, że listy są tej samej długości.
Przykład i przewidywany rezultat:

```
list_1 = [i for i in range(0, 10, 2)]
list_2 = [i for i in range(1, 11, 2)]
list_3 = multiply_lists(list_1, list_2)
print(list_3)
>>> [0, 6, 20, 42, 72]
```

Zadanie 3.
Zrealizuj to samo co w Zadaniu 2. zastępując funkcje **multiply_lists** wyrażeniem lambda o długości maksymalnie jednej linii kodu.

Zadanie 4.
Zdefiniuj funkcje **add_dictionaries** przyjmującej dwa parametry (domyślnie traktowane jako słowniki). Zwróć bez modyfikowania oryginalnych słowników, trzeci słownik zawierający wszystkie pary klucz -> wartość ale w przypadku gdy klucz powtarza się w obu słownikach, dodaj do siebie odpowiadające wartości. Załóż, że jest możliwa operacja dodawania wartości.
Przykład i przewidywany rezultat:

```
dict_1 = {'a': 5, 'b': 7, 'c': 13}
dict_2 = {'c': 15, 'd': 3, 'e': -5}
dict_3 = add_dictionaries(dict_1, dict_2)
print(dict_3)
>>> {('b', 7), ('d', 3), ('e', -5), ('c', 28), ('a', 5)}
```

Zadanie 5.
Zdefiniuj funkcje **extend_lists** przyjmującej jako argument dowolną liczbę argumentów. Załóż, że argumenty wejściowe będą zawsze listami. Funkcja powinna zwrócić, bez modyfikacji list oryginalnych, listę zawierającą wszystkie elementy wszystkich list. Dodatkowo, funkcja powinna usuwać elementy powtarzające się.
Przykład i przewidywany rezultat:

```
list_1 = [1, 2, 3, 4, 15]
list_2 = [3, 4, 5, 6]
list_3 = [6, 7, 8]
list_4 = [8, 9, 10]
print(extend_lists(list_1, list_2, list_3))
print(extend_lists(list_1, list_2, list_3, list_4))
>>> [1, 2, 3, 4, 5, 6, 7, 8, 15]
>>> [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15]
```

Zadanie 6.
Zdefiniuj funkcję **zad_7_func** o dowolnej liczbie argumentów. W przypadku parzystej liczby argumentów funkcja powinna zwrócić słownik o parach: numer argumentu -> argument, a w przypadku nieparzystej liczby argumentów sprawdzić czy liczba argumentów jest liczbą pierwszą.
Przykład i przewidywany rezultat:

```
print(zad_8_func(5, 7, 4, "Temperatura", 5, "Natezenie"))
print(zad_8_func(2, 3, 5))
print(zad_8_func(2, 9, 11, 5, 7, 2, 1, 4, 6))
>>> {(4, 5), (3, 'Temperatura'), (0, 5), (1, 7), (5, 'Natezenie'), (2, 4)}
>>> True
>>> False
```

Zadanie 7.
Zdefiniuj funkcję **zad_7_func**, która jako argument przyjmuje krotkę. Funkcja powinna zwrócić słownik, który do każdego elementu wejściowej krotki przyporządkowuje krotkę zawierającą wszystkie elementy bez danego elementu.
Przykład i przewidywany rezultat:

```
input_tuple = (1, 2, 3, 4, 5)
print(zad_10_func(input_tuple))
>>> {(4, (1, 2, 3, 5)), (5, (1, 2, 3, 4)), (1, (2, 3, 4, 5))
, (3, (1, 2, 4, 5)), (2, (1, 3, 4, 5))}
```