Laboratorium 3 — Ćwiczenia w programowaniu proceduralnym.

Funkcje, parametry opcjonalne, argumenty nazwane i lambdy

Podobnie jak w przypadku $C++$ argumenty funkcji mogą posiadać domyślne wartości (możemy ominąć je wtedy przy wywoływaniu funkcji). To co odróżnia Pythona to parametry nazwane, co pozwala na podawanie argumentów w dowolnej kolejności, o ile znamy ich nazwę. Możliwe jest to ponieważ lista argumentów w Pythonie jest sprowadzana do postaci słownika.

Uwaga: argumenty opcjonalne muszą zawsze znajdować się na końcu.

Uwaga: jako parametry opcjonalne zaleca się stosowanie prostych typów danych lub niezmiennych/stałych (immutable) obiektów. W języku Python argumenty opcjonalne inicjalizowane są jednokrotnie, jakiekolwiek zmiany w obiekcie zostaną zapamiętane między wywołaniami.

```
def nazwa(par1, par2 = 5, par3 = 10):
    a = par1 * par2
    b = par2 + par3
    return (a, b)
# mozliwe wywolania:
nazwa(5)
nazwa(5, 6)
nazwa(5, par3 = 6)
nazwa(par3 = 6, par2 = 6, par1 = 2)
```

Python pozwala także na wykorzystanie funkcji lambda.

```
f = lambda a : a + 10
print(f(5))
```

Zadanie 1 Napisz funkcję, która przyjmuje na wejście wartość w sekundach, a zwraca czas w postaci "x godzin, y minut, z sekund".

Zadanie 2 Napisz funkcję wyznaczają wartość poniższego wzoru, posiadającą dwa opcjonalne parametry: stopień logarytmu — domyślnie 10, wartość k — domyślnie 2.

$$f(x, n, k) = log_n(x^2 + 5) \cdot (k + 1) \cdot x.$$

Wywołaj funkcję dla x = 2, k = 7 stosując argumenty nazwane.

Zadanie 3 Wykorzystaj funkcję lambda do zaimplementowania następującego wzoru:

$$f(x) = sin(x + 1) + cos(x^4).$$

Oblicz wartość dla liczb całkowitych z przedziału $< -5, 2 >$, wynik zapisz w liście.

Funkcje o nieograniczonej liczbie parametrów

W funkcjach o nieograniczonej liczbie parametrów, argumenty poprzedza się jedną lub dwiema gwiazdkami. Najczęściej stosowanymi dla tego typu zmiennych są nazwy **args** oraz **kwargs**.

```
def test(*args, **kwargs):
    print(args, " - " ,kwargs)
```

Wyrażenie z jedną gwiazdką, tj. **args** (od słowa **argument**), czyli argumenty, to zazwyczaj zmienna, która zawiera tuple argumentów pozycyjnych. Używamy go, gdy do funkcji chcemy przekazać dowolną liczbę argumentów pozycyjnych. Parametr ***args** umieszczamy na końcu listy parametrów w definicji funkcji.

```
def suma(a, *args):
    print(a, " - ", args)
    sum = a
    for i in args:
        sum += i
    print(sum)
suma(2, 5, 6, 8, 1)

>>> 2
>>> (5, 6, 8, 1)
>>> 22
```

Wyrażenie z dwoma gwiazdkami, tj. **kwargs** (od słowa **keyword arguments**), czyli argumenty nazwane, to zmienna, która zawiera pary argumentów nazwa-wartość. Przekazane w ten sposób argumenty są dostępne w funkcji w postaci słownika (tj. par klucz-wartość).

```
def utworz_figure(typ, **kwargs):
    print(typ)
    print(kwargs)
    if typ == 'trojkat':
        print(kwargs['a']*kwargs['h']*0.5)
    elif typ == 'prostokat':
        print(kwargs['a']*kwargs['b'])

utworz_figure('prostokat', a = 4, b = 6)

>>> prostokat
>>> {'a': 4, 'b': 6}
>>> 24
```

Nic nie stoi na przeszkodzie, aby jednocześnie używać **args** oraz **kwargs**, musimy jedynie pamiętać o ich kolejności. Należy zacząć od podania zdefiniowanych argumentów, następnie wyrażenie z jedną gwiazdką i na samym końcu z dwoma.

Rozpakowywanie argumentów funkcji

Możliwe jest także zastosowanie gwiazdki w momencie wywołania funkcji, a nie w jej definicji. W takim przypadku. W takim przypadku jej działanie jest odwrotne, tj. pozwala na przypisanie argumentów z listy/słownika (podanej jako argument funkcji) do kolejnych argumentów z jej definicji. Proces taki nazywamy rozpakowywaniem.

```
def rozpakuj(a, b, c):
    print(a)
    print(b)
    print(c)

l = [5,6,2]
rozpakuj(*l)

l2 = {'c' : 5, 'b' : 6, 'a' : 2}
rozpakuj(**l2)

>>> 5
>>> 6
>>> 2
>>> 2
>>> 6
>>> 5
```

Zadanie 4 Napisz funkcję, która przyjmie dowolną liczbę argumentów typu integer i wyznaczy oraz zwróci ich sumę.

Zadanie 5 Napisz funkcję, która oblicza objętości brył: kuli, prostopadłościanu, walca, stożka wykorzystując wyrażenie **kwargs**. Przetestuj napisaną funkcję dla każdej z brył.

Zadanie 6 Napisz **jedną linijkę kodu**, która pozwoli rozpakować cztery wartości z listy do osobnych zmiennych na dwa sposoby: bez zastosowania funkcji oraz z zastosowaniem funkcji.

Zadanie 7 Napisz funkcję obliczającą silnię na dwa sposoby: z wykorzystaniem rekurencji oraz wersję tradycyjną.

Zadanie 8 Zaimplementuj funkcję obliczająca w sposób rekurencyjny n-ty wyraz ciągu „Tribonacciego".

$$F_n = \begin{cases} 0 & \text{dla } n = 0 \\ 0 & \text{dla } n = 1 \\ 1 & \text{dla } n = 2 \\ F_{n-1} + F_{n-2} + F_{n-3} & \text{dla } n > 2 \end{cases}$$