Funkcje w Pythonie to podstawowe bloki kodu, które pozwalają na grupowanie instrukcji wykonujących określone zadanie. Funkcja może przyjmować argumenty i zwracać wynik, co czyni je niezwykle użytecznymi do organizacji i wielokrotnego użycia kodu.

Korzyści z używania funkcji:

- modularność - funkcje pozwalają dzielić kod na mniejsze, niezależne moduły,
- wielokrotne użycie kodu - kod w funkcji można używać w różnych częściach programu,
- łatwiejsze testowanie - funkcje można testować i sprawdzać ich poprawność w izolacji.

Struktura funkcji

Parametry i argumenty	Funkcja może mieć dowolną liczbę parametrów, a także może ich nie mieć wcale. Parametry mogą mieć wartości domyślne, np. def powitanie(imie="Świecie").
Wartość zwracana	Funkcja może zwracać wartość przy użyciu return. Jeśli return nie zostanie użyty, funkcja zwraca None.
Zmienne lokalne	Zmienne zdefiniowane wewnątrz funkcji mają zasięg lokalny (są dostępne tylko wewnątrz tej funkcji).

Funkcję definiuje się za pomocą słowa kluczowego ***def***, nazwy funkcji i nawiasów, w których mogą być umieszczone parametry:

```
def nazwa_funkcji(argument1, argument2):
    # Ciało funkcji - kod, który wykonuje funkcja
    wynik = argument1 + argument2
    return wynik
#wywołanie funkcji
wynik = nazwa_funkcji(5, 10)
print(wynik)
```

W powyższym przykładzie „nazwa_funkcji” jest nazwą tworzonej funkcji, a argument1 i argument2 są parametrami wejściowymi. Słowo kluczowe return pozwala na zwrócenie wyniku z funkcji.
Użycie funkcji polega na jej wywołaniu, poprzez podanie jej nazwy oraz wartości dla jej argumentów.

Deklaracja funkcji w funkcji. Python pozwala na definiowanie funkcji w obrębie innej funkcji – jest ona widoczna tylko wewnątrz zewnętrznej funkcji.

Parametry i argumenty funkcji. Argumenty od parametrów różnią się tym, ze pierwsze są tylko wartościami a drugie, to całe zmienne. Funkcja zawsze musi być wywoływana z określoną liczbą argumentów.

Zasięg zmiennych w funkcjach - zmienne mogą mieć różne zasięgi, zależnie od miejsca ich definicji:

- zasięg lokalny – zmienne zadeklarowane wewnątrz funkcji są widoczne tylko tam.
- zasięg globalny – zmienne zadeklarowane poza funkcją są widoczne w całym kodzie.

Typy funkcji w Pythonie

Funkcje bez parametrów - funkcje, które nie wymagają żadnych danych wejściowych.	def przywitanie(): print("Hello")
Funkcje z parametrami - funkcje, które przyjmują argumenty i używają ich w swoim ciele.	def dodaj(a, b): return a + b
Funkcje z wartością domyślną - parametry mogą mieć wartości domyślne, które są używane, jeśli argument nie zostanie przekazany.	def powitanie(imie="World"): return "Hello, " + imie
Funkcje anonimowe (lambda) - funkcje lambda to funkcje jednowierszowe, które są użyteczne przy krótkich obliczeniach.	dodaj = lambda x, y: x + y wynik = dodaj(2, 3)

Kolejność parametrów w funkcji

(param)	**Parametry pozycyjne**
(param = wartość)	Parametry z domyślnymi wartościami
(*paramTupl)	Parametr zbierający nadmiarowe (pozostałe) argumenty pozycyjne w krotkę
(paramDict)**	Parametr zbierający nadmiarowe argumenty nazwane w słownik

Python wykonuje następujące kroki mające na celu dopasowanie argumentów przed przypisaniem:

1. Przypisuje argumenty niebędące słowami kluczowymi zgodnie z pozycją
2. Przypisuje argumenty będące słowami kluczowymi poprzez dopasowanie nazw
3. Przypisuje dodatkowe argumenty niebędące słowami kluczowymi do krotki *paramTupl
4. Przypisuje pozostałe argumenty będące słowami kluczowymi do słownika **paramDict
5. Przypisuje do argumentów nieprzypisanych wartości domyślne z nagłówka

Przekazywanie obiektów niemutowalnych

Jeśli do funkcji przekazany zostaje obiekt niemutowalny (np. liczbę, napis lub krotkę), jakakolwiek próba zmiany jego wartości wewnątrz funkcji **nie wpłynie na oryginalny obiekt poza funkcją**. Wynika to z faktu, że w momencie próby zmiany wartości, Python tworzy **nowy obiekt**, a nie modyfikuje istniejący.

```
def zwieksz_liczbe(x):
    x = x + 1
    print("Wewnątrz funkcji:", x)
liczba = 10
zwieksz_liczbe(liczba)
print("Poza funkcją:", liczba)
```

Przekazywanie obiektów mutowalnych

Jeśli do funkcji przekaże się obiekt mutowalny (np. listę lub słownik), to zmiany dokonane na tym obiekcie wewnątrz funkcji **wpłyną na oryginalny obiekt poza funkcją**. W tym przypadku parametr w funkcji jest referencją do tego samego obiektu, więc każda modyfikacja obiektu jest widoczna także poza funkcją

```
def dodaj_element(lista):
    lista.append(4)
    print("Wewnątrz funkcji:", lista)

liczby = [1, 2, 3]
dodaj_element(liczby)
print("Poza funkcją:", liczby)
```

Rekurencja

Rekurencja polega na tym, że funkcja wywołuje samą siebie. Ta technika pozwalająca na rozwiązywanie problemów poprzez podział na mniejsze, łatwiejsze do rozwiązania podproblemy. Jednak przy używaniu rekurencji trzeba zachować ostrożność, aby funkcja miała **warunek stopu** – dzięki temu uniknie się nieskończonego wywoływania samej siebie, co mogłoby spowodować przepełnienie stosu wywołań (ang. *stack overflow*).

Przykład:

Silnia to funkcja matematyczna oznaczana symbolem n!, która dla liczby całkowitej n oblicza iloczyn wszystkich liczb całkowitych od 1 do n. Silnię można obliczyć rekurencyjnie, ponieważ:

- 0! wynosi 1 (to jest warunek stopu).
- n! można zdefiniować jako n * (n - 1)! dla n > 0.

```
def silnia(n):
    if n == 0:          # Warunek stopu
        return 1
    else:
        return n * silnia(n - 1)  # Wywołanie
rekurencyjne
```

Warunek stopu: if n == 0: return 1.
Zapobiega nieskończonej rekurencji, ponieważ gdy n wynosi 0, funkcja zwraca 1 i kończy rekurencję.
Wywołanie rekurencyjne: return n * silnia(n - 1). Każde wywołanie redukuje wartość n o 1, aż osiągnie warunek stopu.

Zad. 1
Napisz funkcję, która pozwoli obliczyć kwadrat liczby podanej przez użytkownika.
Zaprezentuj wywołanie funkcji.

Zad. 2
Napisz funkcję odwracającą string.

Zad. 3
Napisz funkcję o nazwie powitanie, która przyjmuje dwa argumenty:
imie – reprezentuje imię osoby, którą chcemy przywitać (domyślna wartość to "Użytkowniku").
jezyk – reprezentuje język, w którym funkcja ma wyświetlić powitanie (domyślna wartość to "polski").
Funkcja powinna działać w następujący sposób:

Jeśli jezyk to "polski", funkcja powinna wyświetlić powitanie w języku polskim, np. Cześć, Anna.
Jeśli jezyk to "angielski", funkcja powinna wyświetlić powitanie w języku angielskim, np. Hello, John.
Jeśli jezyk to "niemiecki", funkcja powinna wyświetlić powitanie w języku niemieckim, np. Guten Morgen, Hans.
Jeśli jezyk to inna wartość, funkcja powinna wyświetlić domyślne powitanie Witaj, <imie>.
Zaprezentuj wywołanie funkcji z różnymi parametrami.

Zad. 4
Napisz funkcję, obliczającą średnią z listy liczb. Zaprezentuj wywołanie funkcji.

Zad. 5

Napisz funkcję, która na podstawie podanej wagi i wzrostu oblicza i zwraca wskaźnik BMI, a następnie informuje użytkownika w jakim jest zakresie.

**Wskaźnika Body Mass Index (BMI) oblicza się na podstawie dwóch danych: wzrostu i aktualna masa ciała.*
Należy podzielić wagę wyrażoną w kilogramach przez wzrost podniesiony do kwadratu, wyrażony w metrach, czyli w skrócie BMI = kg / m2.
Uzyskany wynik należy porównać z zakresami wartości BMI. Klasyfikacja BMI dla osób dorosłych wygląda następująco:

- *niedowaga – BMI poniżej 18,5. Tutaj rozróżnia się trzy podkategorie,*
- *pożądana masa ciała – BMI od 18,5 do 24,9,*
- *nadwaga – BMI od 25 do 29,9,*
- *otyłość – BMI powyżej 30. Obecnie wyróżnia się otyłość I stopnia – od 30 do 34,9, II stopnia – od 35 do 39,9, oraz III stopnia – powyżej 40.*

Zad. 6

Napisz funkcję, która oblicza pole trójkąta o boku a, b ,c. Pamiętaj o zabezpieczeniu funkcji przed błędnymi danymi i wyjątkami. Użyj **wzoru Herona.**

***** *Do obliczenia pola trójkąta o bokach a, b, c można użyć **wzoru Herona**:*

$$\text{Pole} = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}$$

*gdzie **s** to połowa obwodu trójkąta (tzw. Półobwód):*

$$s = \frac{a+b+c}{2}$$

Do sprawdzenia poprawności danych wprowadzanych przez użytkownika zastosuj strukturę Try & Except

```
try:
# Kod, który potencjalnie może wywołać wyjątek
except ExceptionType:
# Kod do wykonania, jeśli wystąpi wyjątek typu ExceptionType
```

Zad 7.

Napisz funkcję rekurencyjną obliczania potęgę n'tego stopnia liczby a, wartości dla argumentów funkcji podaje użytkownik.

Zad 8.

A.

Napisz funkcję o zmiennej liczbie parametrów, która wyświetla wartości parametrów na ekranie.
B.
Zmodyfikuj funkcję tak, aby znajdowała i zwracała wartość maksymalną.

**Użyj parametru *args, który łączy wszystkie dodatkowe (nadmiarowe) argumenty pozycyjne, niebędące słowami kluczowymi podczas wywoływania funkcji, w krotkę.*
W Pythonie `*args` pozwala funkcji przyjmować zmienną liczbę argumentów pozycyjnych, które są przekazywane jako krotka (`tuple`). Dzięki temu można przekazać dowolną liczbę dodatkowych argumentów do funkcji bez konieczności definiowania ich z góry.

Zad 9.

Napisz funkcję rekurencyjną, która poda n-ty wyraz ciągu Fibonacciego.

*Ciąg Fibonacciego to sekwencja liczb, w której każdy wyraz (od trzeciego wzwyż) jest sumą dwóch poprzednich wyrazów. Formuła rekurencyjna dla n-tego wyrazu ciągu Fibonacciego jest następująca.

$$F(n) = F(n-1) + F(n-2)$$

gdzie:

- $F(0) = 0$
- $F(1) = 1$

Zad 10. dodatkowe

Napisz funkcję, która otrzymuje dwa obiekty iterowalne (sekwencje) i zwraca listę wspólnych dla obu obiektów wartości.

Wykorzystaj konwersję do zbiorów podanych sekwencji oraz operację przecięcia wiedząc, że operator **& dla zbiorów zwraca przecięcie, czyli elementy występujące w obu zbiorach.*