Wstęp do programowania

dr inż. Krzysztof Dorywalski

Programowanie proceduralne

Programowanie proceduralne (funkcyjne)

Funkcja to zbiór poleceń, który służy do realizacji określonego zadania w programie, np. obliczenia sumy dwóch liczb. Podział programu na funkcje, które realizują odrębne zadania, nazywamy modularyzacją programu

- czytelny kod
- wielokrotne wykorzystanie kodu
- łatwiejsza praca zespołowa

Definiowanie i wywoływanie funkcji

```
def nazwa_funkcji():
    polecenie_1
    polecenie_2
    itd.
```

Definiowanie i wywoływanie funkcji

```
#definiowanie funkcji
def mnozenie(x, y):
    return x * y

#wywołanie funkjci z argumentami
wynik = mnozenie(3, 5)
print(wynik)
```

Zadanie 1:

Napisz program, który dla danych a i b wprowadzonych z klawiatury oblicza pole prostokąta jako prostą funkcję.

Tworzenie modułu

```python
#MODUŁ FUNKCJE

#funkcja do mnożenia
def mnozenie(x, y):
    return x * y
```

```python
import funkcje
#wywołanie funkcji z argumentami
wynik = funkcje.mnozenie(3, 5)
print(wynik)
```

Importowanie modułów

```
import funkcje
#wywołanie funkcji z argumentami
wynik = funkcje.mnozenie(3, 5)
print(wynik)
```

```
from funkcje import mnozenie
#wywołanie funkcji z argumentami
wynik = mnozenie(3, 5)
print(wynik)
```

```
from funkcje import *
#wywołanie funkcji z argumentami
wynik = mnozenie(3, 5)
print(wynik)
```

```
import funkcje as f
#wywołanie funkcji z argumentami
wynik = f.mnozenie(3, 5)
print(wynik)
```

Zadanie 2:

Napisz moduł zawierający funkcje do obliczania pół figur płaskich: prostokąt, trójkąt, koło i trapez. Napisz skrypt wywołujący funkcje dla przykładowych argumentów.

Funkcja do wyszukiwania powtórzeń

```
def wspolna(sek1, sek2):
    wynik = []
    for x in sek1:
        if x in sek2:
            wynik.append(x)
    return wynik
```

```
s1 = "Marian"
s2 = "Marianna"
porownaj = wspolna(s1,s2)
print(porownaj)
```

Zmienne lokalne a zmienne globalne

```
a = 10

def test():
    b = a + 5
    return b

print(test())
print(a)
```

```
a = 10

def test():
    a = 50
    return a

print(test())
print(a)
```

```
a = 10

def test():
    global a
    a = 50
    return a

print(test())
print(a)
```

Zadanie 3:

Napisz skrypt, do konwersji pomiędzy jednostkami używanymi do opisu fali światła. Program ma wyświetlać menu:

(1) eV -> nm
(2) nm -> eV
(3) eV -> cm-1
(4) cm-1 -> eV
(5) nm -> cm-1
(6) cm-1 -> nm
(k) koniec

Po wybraniu odpowiedniej opcji, wykonana zostanie odpowiednia operacja. Napisz funkcje dla poszczególnych operacji, które umieścisz w module o nazwie *optyka*

Funkcja do sortowania

Zadanie 4*:

Napisz skrypt, który sortuje N liczb (w zadaniu N = 10) znajdujących się na liście o nazwie liczby.

Rekurencja

Funkcję, która wywołuje samą siebie, nazywamy **funkcją rekurencyjną**

```
def wiadomosc(i):
    if i > 0:
        print('To jest funckja rekurencyjna')
        wiadomosc(i-1)

wiadomosc(5)
```

Rekurencja – obliczanie silni

```
def silnia_rek(n):
    if n > 1:
        return n * silnia_rek(n-1)
    else:
        return 1
```

Obliczanie silni – iteracja

```
def silnia_iter(n):
    silnia_temp = 1
    if n > 1:
        for i in range(2, n+1):
            silnia_temp = silnia_temp * i
        return silnia_temp
    else:
        return 1
```

Pomiar wydajności funkcji

```
import timeit

test = """
Definicja funkcji, której czas wykonania chcemy sprawdzić
Wywołanie funkcji
"""
czas = timeit.timeit(test, number=100)/100
print(czas)
```

Zadania – rekurencja

Zadanie 5:

Napisz program, który znajduje *n* pierwszych liczb ciągu Fibonacciego. Zaproponuj wersję rekurencyjną i iteracyjną. Porównaj wydajność obu rozwiązań

Fibonacci - wskazówka

W matematyce ciąg Fibonacciego to ciąg liczb naturalnych określony rekurencyjnie w następujący sposób:

```
jeśli n = 0, to wtedy fib(n) = 0
jeśli n = 1, to wtedy fib(n) = 1
jeśli n >= 2, to wtedy fib(n) = fib(n – 1) + fib(n – 2)
```

Pierwszy wyraz jest równy $F_0 = 0$, drugi jest równy $F_1 = 1$, każdy następny jest sumą dwóch poprzednich, tzn. $F_{n-1} + F_{n-2}$ dla $n > 1$. Wyrazy ciągu Fibonacciego to: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233 itd.

Zadania – rekurencja

Zadanie 6:

Napisz skrypt, który oblicza wartość *a* zgodnie ze wzorem poniżej

$$a^n = \begin{cases} 1 & dla\ n = 0 \\ a \cdot a^{n-1} & dla\ n\ nieparzystych \\ \left(a^{\frac{n}{2}}\right)^2 & dla\ n\ parzystych \end{cases}$$

Zadania – rekurencja

Zadanie 7:

Napisz program, który rekurencyjnie znajduje 10 pierwszych liczb trójkątnych i wyświetla je na ekranie

Liczby trójkątne - wskazówka

W matematyce liczba trójkątna to taka, którą można zapisać w postaci sumy kolejnych początkowych liczb naturalnych: $T_n = 1 + 2 + 3 + \ldots + (n - 1) + n$. Liczby należące do tego ciągu nazywane są trójkątnymi, gdyż można je przedstawić w formie trójkąta. Na przykład #6 = 21.

#1		1					
#2		2	1				
#3		3	2	1			
#4		4	3	2	1		
#5		5	4	3	2	1	
#6		6	5	4	3	2	1

Kolejne elementy ciągu uzyskujemy w sposób następujący:

#1 = 1
#2 = 1 + 2 = 3
#3 = 1 + 2 + 3 = 6
#4 = 1 + 2 + 3 + 4 = 10
#5 = 1 + 2 + 3 + 4 + 5 = 15
#6 = 1 + 2 + 3 + 4 + 5 + 6 = 21 itd.