Listy

Listę tworzymy przy użyciu nawiasów kwadratowych, rozdzielając jej elementy przecinkami.

Definiowanie list

- lista trzech liczb typu int:
  lista1 = [1, 2, 3]
- lista czterech napisów:
  lista2 = ["Analiza", "Algebra", "Python", "R"]
- lista trzech elementów różnych typów:
  lista3 = [1.56, 2, "trzy"]
- lista zawierająca dwie inne listy:
  lista4 = [["dst","db","bdb"], [9,8,7,6,5]]
- lista pusta:
  lista5 = []
- lista jednoelementowa
  lista6 = [100]

Lista jest uporządkowanym zbiorem elementów.
Każdy element listy ma swój indeks.
Indeksy **n**-elementowej listy zaczynają się od **0** a kończą na **n-1**.

Dostęp do elementów listy

lista = [5, 2, 6, 1, 3]
print(lista) - wypisanie całej listy
print(lista[0]) - wypisanie pierwszego elementu listy: **5**
print(lista[4]) - wypisanie ostatniego elementu listy: **3**
print(lista[-3]) - wypisanie elementu trzeciego od końca: **6**
print(lista[1:3]) - wypisanie elementów o indeksach 1 i 2: **[2,6]**
print(lista[2:]) - wypisanie elementów o indeksach od 2 do końca: **[6,1,3]**
print(lista[0::3]) - wypisanie co trzeciego elementu listy: **[5,1]**

Zmiana podlisty

lista = [5, 4, 3, 2, 1]

lista[1] = 0

wynik: [5, 0, 3, 2, 1]

lista[2] = lista[2] + 100

wynik: [5, 0, 103, 2, 1]

lista[0:3] = [7, 7, 7]

wynik: [7, 7, 7, 2, 1]

lista[2:3] = [100]

zmiana elementu o indeksie 2: [7, 7, 100, 2, 1]

lista[1:1] = [5]

dodanie elementu na pozycję 1: [7, 5, 7, 100, 2, 1]

lista[3:5] = [6]

zamiana dwóch elementów na jeden: [7, 5, 7, 6, 1]

lista[::2] = [8, 9, 10]

zmiana co drugiego: [8, 5, 9, 6, 10]

Listy

Dodawanie i usuwanie elementów

lista = [5, 2, 6, 1, 3]

lista = lista*2

powielenie listy: [5, 2, 6, 1, 3, 5, 2, 6, 1, 3]

lista = lista[:3]

skrócenie listy: [5, 2, 6]

lista = lista + [0, 1, 1]

wydłużenie listy: [5, 2, 6, 0, 1, 1]

lista[6:] = [100, 200]

wydłużenie listy: [5, 2, 6, 0, 1, 1, 100, 200]

lista = lista[:2] + lista [3:]

usunięcie elementu o indeksie 2: [5, 2, 0, 1, 1, 100, 200]

lista[2:5] = []

usunięcie elementów o indeksach 2,3,4: [5, 2, 100, 200]

Porównywanie list

- Jeżeli elementy obu list są sobie równe, listy są równe.
- Jeżeli listy różnią się choć jednym elementem, to nie są równe.
- Jeżeli pierwszy element pierwszej listy jest większy od pierwszego elementu drugiej listy, to pierwsza lista jest większa od drugiej.
- Jeżeli pierwszy element pierwszej listy jest taki sam jak pierwszy element drugiej listy, decyduje porównanie drugich elementów, itd.
- Element nieistniejący jest zawsze mniejszy od każdego innego elementu.

Przykłady

lista = [1, 2, 3, 4]
print(lista == [1, 2, 4, 4]) - False
print(lista != [1, 2, 4, 4]) - True
print(lista > [1, 2, 2, 5]) - True
print(lista < [1, 2, 3, 4, 5]) - True

Pętla for i lista

Przetestuj program

```
lista = [1, 2, 3, 4, 5]
for i in lista:
    print(i)
```

Zmienna **i** w danym powtórzeniu pętli przyjmuje wartość kolejnych elementów listy.
Do wypisania wszystkich elementów w jednej linii użyj: **print(i, end="")**
Aby elementy listy wypisać w postaci: *indeks elementu*, *wartość elementu*, korzystamy z funkcji **enumerate**.

Przetestuj program

```
lista = [1, 2, 3, 4, 5]
for nr, wartosc in enumerate(lista):
    print(nr, wartosc)
```

Przetestuj program

```
lista = [1, 2, 3, 4, 5, 6, 7]
for i in lista[3:5]:
    print(i, i**2)
```

Przetestuj program

```
lista = [1, -2, -3, 4, 5]
for i in lista:
    if i>0:
        print(i, i**0.5)
```

Przetestuj program

```
import random
n = int(input("Podaj długość listy"))
lista = []
for i in range(n):
    lista = lista + [random.randint(1, 5)]
print(lista)
```

W każdym zadaniu należy utworzyć listę 10-elementową i wypełnić ją losowymi liczbami całkowitymi z przedziału [1,5]. Nie stosujemy gotowych metod list.

Zad. 38. Napisz program, który wypisze elementy większe od 3.
Zad. 39. Napisz program, który zmodyfikuje listę, podnosząc jej elementy do kwadratu.
Zad. 40. Napisz program, który obliczy iloczyn elementów listy.
Zad. 41. Napisz program, który obliczy średnią arytmetyczną elementów listy.
Zad. 42. Napisz program, który zliczy elementy nieparzyste oraz obliczy ich sumę.
Zad. 43. Napisz program, który sprawdzi, czy wskazany element występuje w liście.
Zad. 44. Napisz program, który dla danego elementu poda liczbę jego wystąpień w liście.

W każdym zadaniu należy utworzyć listę 10-elementową i wypełnić ją losowymi liczbami całkowitymi z przedziału [1,5]. Nie stosujemy gotowych metod list.

Zad. 45. Napisz program, który znajdzie element maksymalny listy.
Zad. 46. Napisz program, który znajdzie parę sąsiednich elementów o największej sumie.
Zad. 47. (dodatkowe) Napisz program, który znajdzie element najczęściej występujący w liście.
Zad. 48. (dodatkowe) Napisz program, który znajdzie drugi co do wielkości element listy.

Metody listy

- L.**append(x)**
  - dodaje element **x** na koniec listy **L**
  - odpowiednik **L[len(L):] = [x]**
- L.**extend(S)**
  - dodaje wszystkie elementy sekwencji **S** na koniec listy **L**
  - odpowiednik **L[len(L):] = S**
- L.**insert(i,x)**
  - wstawia element **x** na pozycji **i**
  - odpowiednikiem **L.insert(len(L),x)** jest **L.append(x)**
- L.**remove(x)** - usuwa z listy **L** pierwszy element, którego wartością jest **x** i zwraca go. Jeśli nie ma takiej wartości, zgłaszany jest błąd.
- L.**pop(i)**
  - usuwa z listy **L** element na pozycji **i** i zwraca go.
  - polecenie **L.pop()** usuwa i zwraca ostatni element z listy **L**
- L.**index(x)** - zwraca pozycję elementu **x** w liście **L**
- L.**count(x)** - zlicza wystąpienia elementu **x** w liście **L**
- L.**sort()** - sortuje listę **L** w porządku rosnącym
- L.**reverse()** - odwraca kolejność elementów listy **L**

Metody listy - przykłady

Append

```
L = [1, 2, 3]
L.append(5)
print(L)
#wynik: [1, 2, 3, 5]
```

Zamiast append...

```
L = [1, 2, 3]
L[len(L):] = [x]
print(L)
#wynik: [1, 2, 3, 5]
```

Funkcja **len(L)** zwraca długość listy **L**.
Znak # oznacza komentarz liniowy.

Metody listy - przykłady

Extend

```
L = [1, 2, 3]
L.extend([5,5,5])
print(L)
#wynik: [1, 2, 3, 5, 5, 5]
```

Zamiast extend...

```
L = [1, 2, 3]
L[len(L):] = [5,5,5]
print(L)
#wynik: [1, 2, 3, 5, 5, 5]
```

Metody listy - przykłady

Insert

```
L = [1, 2, 3]
L.insert(1,5)
print(L)
#wynik: [1, 5, 2, 3]
```

Insert i append

```
L = [1, 2, 3]
L.insert(len(L),4)    #lub  L.append(4)
print(L)
#wynik: [1, 2, 3]
```

Metody listy - przykłady

Remove

```
L = [1, 2, 3, 5, 5]
L.remove(5)
print(L)
#wynik: [1, 2, 3, 5]
```

Pop

```
L = [1, 2, 3, 4, 5]
L.pop(1)
print(L)
#wynik: [1, 3, 4, 5]
L.pop()
print(L)
#wynik: [1, 3, 4]
```

Metody listy - przykłady

Index

```
L = [1, 2, 3, 5, 5]
n = L.index(5)
print(n)
#wynik: 3
```

Index

```
L = [1, 2, 3, 5, 5]
print(L.index(5))
#wynik: 3
```

Index

```
L = [1, 2, 3, 5, 5]
print(L.index(4))
#wynik: ...ValueError: 4 is not in list
```

Count

```
L = [1, 2, 3, 5, 5]
print(L.count(5))
#wynik: 2
print(L.count(4))
#wynik: 0
```

Sort

```
L = [4, 1, 8, 5, 1, 3, 6, 0]
L.sort()
print(L)
#wynik: [0, 1, 1, 3, 4, 5, 6, 8]
```

Reverse

```
L = [1, 2, 3, 4, 5, 6]
L.reverse()
print(L)
#wynik: [6, 5, 4, 3, 2, 1]
```

Zad. 49. Napisz program, który wprowadzi do listy n początkowych wyrazów ciągu Fibonacciego, a następnie wypisze jej elementy. Ciąg Fibonacciego dany jest wzorem:

$$F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2}.$$

Zad. 50. Napisz program, który utworzy i wypisze listę składającą się z n losowych liczb całkowitych z przedziału $[1, 50]$, następnie usunie z listy elementy o wartościach z przedziału $[11, 21]$ oraz wyświetli listę zmodyfikowaną.

Zad. 51. Napisz program, który utworzy i wypisze listę składającą się z n losowych liczb całkowitych z przedziału $[1, 50]$, a następnie po każdej liczbie podzielnej przez 3 wstawi nowy element o wartości 0; na koniec wyświetli listę zmodyfikowaną.

Przykład:
Lista pierwotna: [28, 15, 4, 22, 18, 33, 47, 35]
Lista zmodyfikowana: [28, 15, 0, 4, 22, 18, 0, 33, 0, 47, 35]

Macierze (tablice wielowymiarowe)

Jednym z łatwiejszych sposobów reprezentowania w Pythonie macierzy są listy z zagnieżdżonymi podlistami.

Tablica 3x3 oparta na listach

```
M = [[1,2,3],
     [4,5,6],
     [7,8,9]]
print(M[1])
#wynik: [4,5,6]
print(M[1][1])
#wynik: 5
print(M[2][0])
#wynik: 7
```

Jeden indeks pozwala pobrać cały wiersz (a tak naprawdę zagnieżdżoną podlistę), natomiast dwa indeksy - pojedynczy element tego wiersza.

Macierze (tablice wielowymiarowe)

Utworzenie i wypisanie macierzy wymiaru n x m

```
import random
n = int(input("Podaj liczbę wierszy: "))
m = int(input("Podaj liczbę kolumn: "))

#najpierw macierz zerowa:
M = [[0]*m for i in range(n)]
for i in range(n):
    print(M[i])

#wypełnienie macierzy zerowej liczbami losowymi 0, 1, . . . , 9
for i in range(n):
    for j in range(m):
        M[i][j] = random.randint(0,9)
```

Macierze (tablice wielowymiarowe)

Utworzenie i wypisanie macierzy wymiaru n x m (c.d.)

```
#wypisanie macierzy
for i in range(n):
    print(M[i])

#przykładowy wynik dla n = 3 i m = 5:
[0, 4, 0, 6, 8]
[4, 6, 8, 2, 6]
[8, 0, 1, 2, 9]
```

Macierze (tablice wielowymiarowe)

Dodawanie macierzy

```
import random
n = int(input("Podaj wymiar macierzy kwadratowej: "))

#utworzenie dwóch macierzy zerowych A i B
A = [[0]*n for i in range(n)]
B = [[0]*n for i in range(n)]

#wypełnienie macierzy A i B liczbami losowymi
for i in range(n):
    for j in range(n):
        A[i][j] = random.randint(0,9)
        B[i][j] = random.randint(0,9)
```

Macierze (tablice wielowymiarowe)

Dodawanie macierzy (c.d.)

```
#wypisanie macierzy A i B
print("Macierz A:")
for i in range(n):
    print(A[i])
print("Macierz B:")
for i in range(n):
    print(B[i])

#macierz C powstała w wyniku dodania macierzy A i B
C = [[0]*n for i in range(n)]
for i in range(n):
    for j in range(n):
        C[i][j] = A[i][j] + B[i][j]

#wypisanie macierzy wynikowej C
for i in range(n):
    print(C[i])
```

Zad. 52. Napisz program, który utworzy macierz wymiaru n x m składającą się z losowych liczb całkowitych z przedziału [0,9] i doda do siebie wszystkie jej elementy. Parametry n i m należy pobrać od użytkownika.

Zad. 53. Napisz program, który utworzy macierz wymiaru n x m składającą się z losowych liczb całkowitych z przedziału [0,9] i znajdzie element maksymalny oraz jego położenie (indeks wierszowy i kolumnowy). Parametry n i m należy pobrać od użytkownika.

Zad. 54. Napisz program, który utworzy macierz A wymiaru n x m składającą się z losowych liczb całkowitych z przedziału [0,9], a następnie utworzy macierz transponowaną A^T. Parametry n i m należy pobrać od użytkownika.

Zad. 55. Napisz program, który utworzy macierze A_{4x5} i B_{5x3} składające się z losowych liczb całkowitych z przedziału [0,9] oraz macierz C_{4x3} będącą iloczynem macierzy A i B.

Krotki

- Krotka (ang. tuple) jest w przybliżeniu listą, której nie można zmodyfikować.
- Krotki są kodowane w nawiasach okrągłych, a nie w nawiasach kwadratowych.
- Krotki obsługują dowolne typy danych, zagnieżdżanie i zwykłe operacje na sekwencjach.
- Krotki mają dwie metody wywoływalne (index i count) specyficzne dla tego typu — nie jest ich zatem tak wiele jak w przypadku list.

Po co zatem typ podobny do listy, który obsługuje mniejszą liczbę operacji?
W praktyce krotki nie są używane tak często jak listy, jednak ich niezmienność jest ich zaletą. Kiedy w programie przekazujemy zbiór obiektów w postaci listy, obiekty te mogą się w dowolnym momencie zmienić. W przypadku użycia krotki zmienić się nie mogą. Krotki nakładają pewne ograniczenia w zakresie integralności, które przydają się w programach większych od pisanych w tej prezentacji.

Krotki

Przetestuj program

```
T = (1,2,3,4)
print(T)
print(len(T))
print(T[1])
print(T.index(4))
print(T.count(4))
R = T + (5,6)
print(R)
S = (5,6) + T
print(S)
```

Sprawdź, że poniższy program zakończy się błędami

```
T = (1,2,3,4)
T[1] = 6          #Krotki są niezmienne.
T.append(5)       #Krotki nie powiększają/nie zmniejszają się.
```