Wizualizacja danych - wyklad 4

Odpowiedź na pytanie:

"Mutable" - zmienne typy::

- list
- dictionary
- set
- bytearray
- user defined classes

“Inmutable” - niezmienne typy:

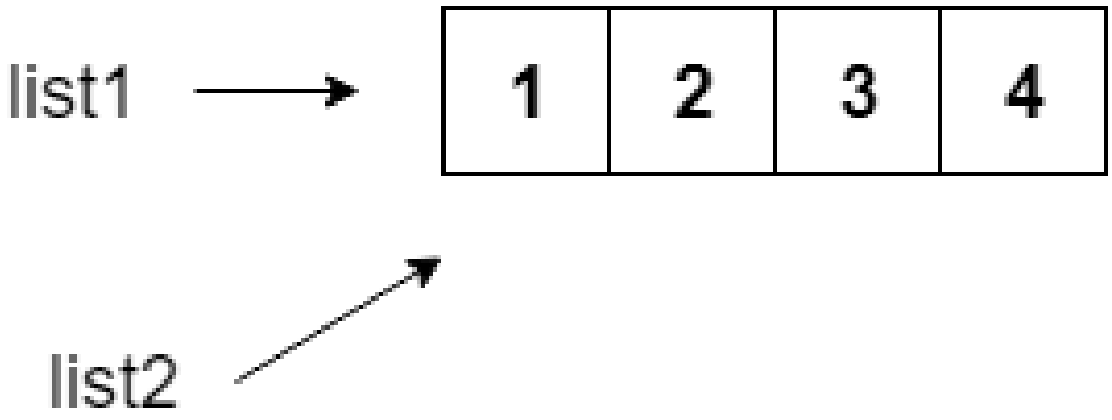
- int
- float
- decimal
- complex
- bool
- string
- tuple
- range
- frozenset
- bytes

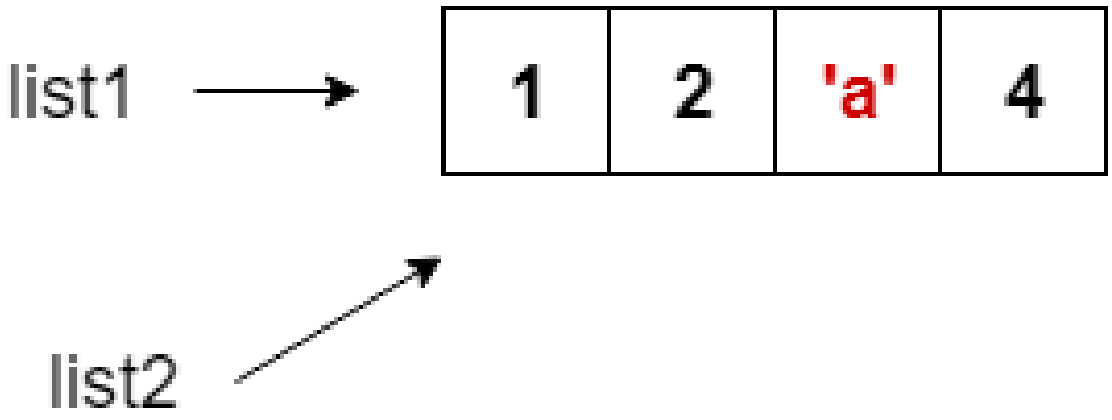
```
list1 = [1, 2, 3, 4]
list2 = list1
list1[2] = 'a'
print(list1)
```

```
## [1, 2, 'a', 4]
```

```
print(list2)
```

```
## [1, 2, 'a', 4]
```


list1
1
2
3
4
list2


list1
1
2
'a'
4
list2

Listy

Lista w Pythonie to tzw. typ sekwencyjny umożliwia przechowywanie elementów różnych typów.

Cechy:

- zmienny (`mutable`) - umożliwia przypisanie wartości pojedynczym elementom
- do zapisu używamy nawiasów kwadratowych
- poszczególne elementy rozdzielamy przecinkami
- każdy element listy ma przyporządkowany indeks
- elementy listy są numerowane od zera
- listy są uporządkowane
- listy są dynamiczne (mogą mieć różną długość)
- listy mogą być zagnieżdżone

Uwaga!

Listy w języku Python są specyficzną strukturą danych nie zawsze dostępną w innych językach programowania. Pojęcie listy w całej informatyce "szersze". Wyróżnia się np. listy jednokierunkowe, które nie muszą mieć indeksu. Nie będziemy takich przypadków analizować.

```
nazwa = [element1, element2, ..., elementN]
```

Pusta lista:

```
a = []
b = list()
```

Lista z liczbami:

```
a = [2, 3, 4.5, 5, -3.2]
```

Lista mieszana:

```
b = ['abcd', 25+3j, True, 1]
```

Kolejność ma znaczenie

```
a = [1, 2, 3, 4]
b = [4, 3, 2, 1]
print(a == b)
```

```
## False
```

Elementy na liście nie muszą być unikalne

```
a = [1, 2, 3, 4, 2]
b = [1, 2, 3, 4]
print(a)
```

```
## [1, 2, 3, 4, 2]
```

```
print(a == b)
```

```
## False
```

Indeks - dostęp do elementów listy (od zera)

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[1])
```

```
## 3
```

```
print(a[4])
```

```
## -2.3
```

```
print(a[0])
#print(a[7])
```

```
## 1
```

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[-1])
```

```
## 9.3
```

```
print(a[-5])
```

```
## abc
```

```
print(a[-7])
```

```
## 1
```

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[1:4])
```

```
## [3, 'abc', False]
```

```
print(a[-5:-2])
```

```
## ['abc', False, -2.3]
```

```
print(a[:4])
```

```
## [1, 3, 'abc', False]
```

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[2:])
```

```
## ['abc', False, -2.3, 'XYZ', 9.3]
```

```
print(a[0:6:2])
```

```
## [1, 'abc', -2.3]
```

```
print(a[1:6:2])
```

```
## [3, False, 'XYZ']
```

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[6:0:-2])
```

```
## [9.3, -2.3, 'abc']
```

```
print(a[::-1])
```

```
## [9.3, 'XYZ', -2.3, False, 'abc', 3, 1]
```

```
print(a[:])
```

```
## [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
```

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(a[::2])
```

```
## [1, 'abc', -2.3, 9.3]
```

```
print(a[::-2])
```

```
## [9.3, -2.3, 'abc', 1]
```

Specjalne funkcje

Długość (rozmiar listy)

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(len(a))
```

```
## 7
```

Implementacja samodzielna długości

```
def dlugosc(lista):
    x = 0
    for i in lista:
        x += 1

    return x


a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
print(dlugosc(a))
```

```
## 7
```

Maksimum i minimum?

Działa wtedy gdy mamy porządek

- liczby <=
- napisy - porządek leksykograficzny (omówimy przy napisach)

```
a = [4,-5,3.4,-11.2]
print(min(a))
```

```
## -11.2
```

```
print(max(a))
```

```
## 4
```

```
b = ['abc', 'ABCd', 'krt', 'abcd']
print(min(b))
```

```
## ABCd
```

```
print(max(b))
```

```
## krt
```

Modyfikacja i wstawianie

```
a = [4,-5,3.4,-11.2]
a[2] = 'a'
print(a)
```

```
## [4, -5, 'a', -11.2]
```

```
a = [4,-5,3.4,-11.2]
a[2] = ['a','b']
print(a)
```

```
## [4, -5, ['a', 'b'], -11.2]
```

```
a = [4,-5,3.4,-11.2]
a[1:2] = ['a','b']
print(a)
```

```
## [4, 'a', 'b', 3.4, -11.2]
```

```
a = [4,-5,3.4,-11.2]
a[1:3] = ['a','b']
print(a)
```

```
## [4, 'a', 'b', -11.2]
```

Dodawanie list

```
a = [4,-5,3.4,-11.2]
b = ['a','b','c']
print(a+b)
```

```
## [4, -5, 3.4, -11.2, 'a', 'b', 'c']
```

Mnożenie listy przez liczbę całkowitą (int)

```
a = [4,-5,3.4,-11.2]
print(a*2)
```

```
## [4, -5, 3.4, -11.2, 4, -5, 3.4, -11.2]
```

```
print(2*a)
```

```
## [4, -5, 3.4, -11.2, 4, -5, 3.4, -11.2]
```

Usuwanie elementów z listy

```
a = [1, 3, 'abc', False, -2.3, 'XYZ', 9.3]
del a[2]
print(a)
```

```
## [1, 3, False, -2.3, 'XYZ', 9.3]
```

```
del a[:]
print(a)
```

```
## []
```

Do poczytania

- https://docs.python.org/3.9/tutorial/introduction.html#lists
- https://docs.python.org/3.9/tutorial/datastructures.html#more-on-lists

`list.append(x)` - dodaje element na końcu listy. Równoważnie `a[len(a):] = [x]`

```
a = [1, 3, 'abc', False]
a.append(5.3)
print(a)
```

```
## [1, 3, 'abc', False, 5.3]
```

`list.extend(iterable)` - dodaje elementy z argumenty na koniec listy. Równoważnie: `a[len(a):] = iterable`

```
a = [1, 3, 'abc', False]
b = [3, -2]
a.extend(b)
print(a)
```

```
## [1, 3, 'abc', False, 3, -2]
```

Różnice?

```
a = [1, 3, 'abc', False]
b = [3, -2]
a.append(b)
print(a)
```

```
## [1, 3, 'abc', False, [3, -2]]
```

`list.insert(i, x)` - wstawia element `x` na pozycji `i`

```
a = [1, 3, 'abc', False]
a.insert(0,'w')
print(a)
```

```
## ['w', 1, 3, 'abc', False]
```

```
a.insert(4,9.0)
print(a)
```

```
## ['w', 1, 3, 'abc', 9.0, False]
```

`list.remove(x)` - usuwa element z listy (pierwszy od początku)

```
a = [1, 3, 'abc', False]
a.remove(False)
print(a)
```

```
## [1, 3, 'abc']
```

```
b = [3, 4, 5, 3]
b.remove(3)
print(b)
```

```
## [4, 5, 3]
```

`list.pop()` - usuwa i zwraca ostatni element

`list.pop(i)` - usuwa i zwraca element na pozycji `i`

```
a = [1, 3, 'abc', False]
print(a.pop())
```

```
## False
```

```
print(a)
```

```
## [1, 3, 'abc']
```

```
b = [3, -4, 6.2, 7]
print(b.pop(3))
```

```
## 7
```

```
print(b)
```

```
## [3, -4, 6.2]
```

`list.clear()` - usuwa wszystkie elementy z listy. Równoważnie: `del a[:]`

```
a = [1, 3, 'abc', False]
a.clear()
print(a)
```

```
## []
```

`list.index(x)` - zwraca indeks elementu x (o ile istnieje, inaczej błąd), w przypadku duplikatów pierwszy z lewej
`list.index(x,start)` - zwraca indeks elementu x (o ile istnieje, inaczej błąd) zaczynając od pozycji `start`, w przypadku duplikatów pierwszy z lewej `list.index(x,start,end)` - zwraca indeks elementu x (o ile istnieje, inaczej błąd) zaczynając od pozycji `start` a kończąc na `end-1`, w przypadku duplikatów pierwszy z lewej

```
a = [1, 3, 1, 4, 5, 2, 3]
print(a.index(3))
```

```
## 1
```

```
print(a.index(3, 5))
```

```
## 6
```

```
print(a.index(3, 1, 4))
```

```
## 1
```

```
a = ['abc','xyz','abc','efg']
print(a.index('abc'))
```

```
## 0
```

```
print(a.index('abc', 2))
```

```
## 2
```

```
print(a.index('abc', 1, 4))
```

```
## 2
```

`list.count(x)` - zwraca ile razy występuję element `x` na liście

```
a = ['abc','xyz','abc','efg']
print(a.count('abc'))
```

```
## 2
```

```
print(a.count(4))
```

```
## 0
```

`list.sort()` - sortuje listę (o ile elementy można posortować)

```
a = ['abc','xyz','abc','efg']
a.sort()
print(a)
```

```
## ['abc', 'abc', 'efg', 'xyz']
```

`list.reverse()` - odwraca kolejność elementów na liście (nie ma nic związku z sortowaniem!)

```
a = [4, 5, -2, 7.3, 9, -22, 23]
a.reverse()
print(a)
```

```
## [23, -22, 9, 7.3, -2, 5, 4]
```

`list.copy()` - tworzy kopię listy

Spójrzmy na przykład jak działa operator przypisania dla list.

```
a = [4, 5, -2, 7.3, 9, -22, 23]
b = a
b[2] = 100
print(b)
```

```
## [4, 5, 100, 7.3, 9, -22, 23]
```

```
print(a)
```

```
## [4, 5, 100, 7.3, 9, -22, 23]
```

Różnica z użyciem copy.

```
a = [4, 5, -2, 7.3, 9, -22, 23]
b = a.copy()
b[2] = 100
print(b)
```

```
## [4, 5, 100, 7.3, 9, -22, 23]
```

```
print(a)
```

```
## [4, 5, -2, 7.3, 9, -22, 23]
```

Lista jako stos

```
stack = [3, 4, 5, 8, 9]
stack.append(6)
stack.append(7)
print(stack)
```

```
## [3, 4, 5, 8, 9, 6, 7]
```

```
print(stack.pop())
```

```
## 7
```

```
print(stack)
```

```
## [3, 4, 5, 8, 9, 6]
```

Lista jako kolejka

```
from collections import deque

queue = deque(["aw", "tg", "kj"])
queue.append("gg")
print(queue)
```

```
## deque(['aw', 'tg', 'kj', 'gg'])
```

```
print(queue.popleft())
```

```
## aw
```

```
print(queue)
```

```
## deque(['tg', 'kj', 'gg'])
```

List Comprehensions

```
squares = []
for x in range(5):
    squares.append(x ** 2)

print(squares)
```

```
## [0, 1, 4, 9, 16]
```

```
squares = [x**2 for x in range(5)]
print(squares)
```

```
## [0, 1, 4, 9, 16]
```

Krotka - tuple

```
krotka = 123, 'abc', True
print(krotka[2])
```

```
## True
```

```
krotka[0] = 1
```

```
## TypeError: 'tuple' object does not support item assignme
```

Zbiór - set

```
cyfry = {'raz', 'dwa', 'raz', 'trzy', 'raz', 'osiem'}
print(cyfry)
```

```
## {'dwa', 'osiem', 'raz', 'trzy'}
```

Słownik

```
tel = {'jack': 4098, 'sape': 4139}
tel['guido'] = 4127
print(tel)
```

```
## {'jack': 4098, 'sape': 4139, 'guido': 4127}
```

```
tel['jack']
```

```
## 4098
```

```
del tel['sape']
tel['irv'] = 4127
print(tel)
```

```
## {'jack': 4098, 'guido': 4127, 'irv': 4127}
```

```
print(list(tel))
```

```
## ['jack', 'guido', 'irv']
```

```
print(sorted(tel))
```

```
## ['guido', 'irv', 'jack']
```