

Wizualizacja danych

- wykład 4

dr Piotr Jastrzębski

Wstęp do języka Python - cd.

Struktury danych w Pythonie

- ▶ listy
- ▶ zbiory
- ▶ krotki
- ▶ słowniki

```
lista = ['a', 'b', 34, 5.6, True, 3, 'ww']  
lista.pop()
```

```
## 'ww'
```

```
print(lista)
```

```
## ['a', 'b', 34, 5.6, True, 3]
```

```
lista.pop(4)
```

```
## True
```

```
print(lista)
```

```
## ['a', 'b', 34, 5.6, 3]
```

```
lista.pop(-2)
```

```
## 5.6
```

```
print(lista)
```

```
## ['a', 'b', 34, 3]
```

```
lista.pop(0)
```

```
## 'a'
```

```
print(lista)
```

```
## ['b', 34, 3]
```

```
lista.clear()  
print(lista)
```

```
## []
```

Alternatywnie: `del lista[:]`.

```
lista2 = ['a', 'b', 5, 'A', 'a', 'b']  
print(lista2.index('a'))
```

```
## 0
```

```
print(lista2.index('a', 3))
```

```
## 4
```

```
print(lista2.index('a', 1, 4))
```

```
## ValueError: 'a' is not in list
```

```
print(lista2.index('a', 1, 5))
```

```
## 4
```

```
lista2.reverse()
```

```
print(lista2)
```

```
## ['b', 'a', 'A', 5, 'b', 'a']
```

```
lista3 = ['a', 'b', 'A', 'a', 'b']
```

```
lista3.sort()
```

```
print(lista3)
```

```
## ['A', 'a', 'a', 'b', 'b']
```



```
lista4 = lista3.copy()
print(lista4)
```

```
## ['A', 'a', 'a', 'b', 'b']
```

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 assignment
```

Zbiór - set

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

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

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']
```


Odpowiedź na pytanie - jeszcze raz:

“Mutable” - zmienne typy::

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

“Inmutable” - niezmiennie typy:

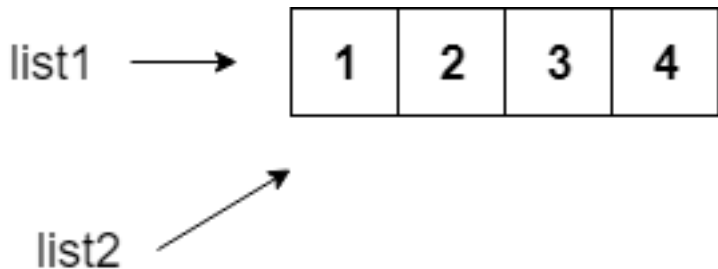
- ▶ int
- ▶ float
- ▶ decimal
- ▶ complex
- ▶ bool
- ▶ string
- ▶ tuple
- ▶ range
- ▶ frozenset
- ▶ bytes

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

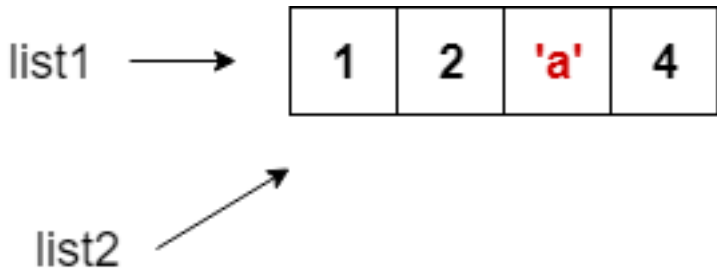
```
## [1, 2, 'a', 4]
```

```
print(list2)
```

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



Rysunek 1: Dwie pierwsze linijki.



Rysunek 2: `list1[2] = 'a'`

Funkcje

```
def functionname( parameters ):  
    "function_docstring"  
    function_suite  
    return [expression]
```

```
def printme(str):  
    """Funkcja wyświetlająca string"""  
    print(str)  
    return
```

```
printme("abc")
```

```
## abc
```

```
print(printme.__doc__)
```

```
## Funkcja wyświetlająca string
```

Przekazywanie przez referencję

```
def changeme(lista):  
    print("Przed zmianą: ", lista)  
    lista[2] = 50  
    print("Po zmianie: ", lista)  
    return
```

```
mylist = [10, 20, 30]  
changeme(mylist)
```

```
## Przed zmianą: [10, 20, 30]  
## Po zmianie: [10, 20, 50]
```

```
print("Poza funkcją: ", mylist)
```

```
## Poza funkcją: [10, 20, 50]
```



```
def changeme(lista):  
    lista = [2, 3, 4]  
    print("Wewnątrz funkcji: ", lista)  
    return
```

```
lista = [10, 20, 30]  
changeme(lista)
```

```
## Wewnątrz funkcji: [2, 3, 4]
```

```
print("Poza funkcją: ", lista)
```

```
## Poza funkcją: [10, 20, 30]
```

```
def changeme():  
    global lista  
    lista = [2, 3, 4]  
    print("Wewnątrz funkcji: ", lista)  
    return
```

```
changeme()
```

```
## Wewnątrz funkcji: [2, 3, 4]
```

```
print("Poza funkcją: ", lista)
```

```
## Poza funkcją: [2, 3, 4]
```

Obowiązkowy argument

```
def printme(str):  
    print(str)  
    return
```

```
printme()
```

```
## TypeError: printme() missing 1 required  
positional argument: 'str'
```

Keyword argument

```
def kwadrat(a):  
    return a*a
```

```
print(kwadrat(a=4))
```

```
## 16
```

Domyślny argument

```
def sumsub(a, b, c=0, d=0):  
    return a - b + c - d
```

```
print(sumsub(12, 4))
```

```
## 8
```

```
print(sumsub(3, 4, 5, 7))
```

```
## -3
```

```
def srednia(first, *values):  
    return (first + sum(values)) / (1 + len(values))
```

```
print(srednia(2, 3, 4, 6))
```

```
## 3.75
```

```
print(srednia(45))
```

```
## 45.0
```

```
def f(**kwargs):  
    print(kwargs)
```

```
f()
```

```
## {}
```

```
f(pl="Polish", en="English")
```

```
## {'pl': 'Polish', 'en': 'English'}
```

Funkcje matematyczne

Link do dokumentacji <https://docs.python.org/3/library/math.html>

```
import math
```

```
a=0
```

```
b=math.sin(2*math.pi)
```

```
print(b)
```

```
## -2.4492935982947064e-16
```

```
print(math.isclose(a,b, rel_tol=1e-09, abs_tol=1e-09))
```

```
## True
```


Programowanie obiektowe w Pythonie



Rysunek 3: Lego jako model programowanie obiektowego

```
class Employee:
    """Common base class for all employees"""
    empCount = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print("Total Employee %d" % Employee.empCount)

    def displayEmployee(self):
        print("Name : ", self.name, ", Salary: ",
              self.salary)
```

```
emp1 = Employee("John", 2000)
emp2 = Employee("Anna", 5000)
emp1.displayEmployee()
```

```
## Name : John , Salary: 2000
```

```
emp2.displayEmployee()
```

```
## Name : Anna , Salary: 5000
```

Bibliografia

- ▶ <https://pl.wikipedia.org/wiki/Python>, dostęp online 12.02.2019.
- ▶ <https://bulldogjob.pl/news/264-java-php-ruby-jak-wlasciwie-wymawiac-nazwy-technologie>. dostęp online 12.02.2019.
- ▶ https://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html, dostęp online 14.02.2019.
- ▶ K. Ropiak, Wprowadzenie do języka Python, <http://wmii.uwm.edu.pl/~kropiak/wd/Wprowadzenie%20do%20j%C4%99zyka%20Python.pdf>, dostęp online 14.02.2019.
- ▶ B. Slatkin, Efektywny Python. 59 sposobów na lepszy kod, Helion 2015.

Bibliografia - cd2

- ▶ <https://docs.python.org/3/tutorial/datastructures.html>, dostęp online 1.03.2019.
- ▶ https://www.python-course.eu/python3_functions.php, dostęp online 2.03.2019.
- ▶ https://www.tutorialspoint.com/python3/python_functions.htm, dostęp online 2.03.2019.
- ▶ https://www.tutorialspoint.com/python3/python_classes_objects.htm, dostęp online 3.03.2019.
- ▶ <https://pl.wikipedia.org/wiki/Wizualizacja>