

Wizualizacja danych

- wykład 1

dr Piotr Jastrzębski

Sprawy organizacyjne

Sprawy organizacyjne

- ▶ Sylabus jest dostępny w systemie USOS.
- ▶ Regulamin zajęć dostępny jest na stronie prowadzącego zajęcia <http://wmii.uwm.edu.pl/~piojas/>.
- ▶ Forma zaliczenia: egzamin.
- ▶ Wykład - 15 godzin, zajęcia w ustalonych terminach.

Wymagania wstępne

- ▶ Znajomość podstawowych konstrukcji programistycznych (ze wstępu do programowania).
- ▶ Matematyka z zakresu szkoły średniej/z przedmiotu repozytorium matematyki elementarnej.

Ewentualne braki należy opanować w samodzielnym zakresie.

W razie problemów zapraszam na konsultacje.

Wstęp to języka Python

Język Python

- ▶ Poprawna wymowa: pajton.
- ▶ Język Python stworzył we wczesnych latach 90. Guido van Rossum – jako następcę języka ABC.
- ▶ Nazwa języka pochodzi od serialu komediowego emitowanego w latach siedemdziesiątych przez BBC – „Monty Python's Flying Circus” (Latający cyrk Monty Pythona). Projektant, będąc fanem serialu i poszukując nazwy krótkiej, unikalnej i nieco tajemniczej, uznał tę za świetną.

Przełomowy rok - 2008

- ▶ Utworzenie drugiej gałęzi rozwoju 3.x. Początkowe obie gałęzie były rozwijane niezależnie, lecz w roku 2020 zakończono wsparcie dla gałęzi 2.x.

Podstawowe różnice między 2.x a 3.x

- ▶ funkcja print

Python 2:

```
print 'Hello, World!'
print('Hello, World!')
print "text", ; print 'print more text on the same line'
```

Python 3

```
print('Hello, World!')
print("some text,", end='')
print(' print more text on the same line')
```


Dzielenie zmiennych typu int

Python 2:

```
print '3 / 2 =', 3 / 2
print '3 // 2 =', 3 // 2
print '3 / 2.0 =', 3 / 2.0
print '3 // 2.0 =', 3 // 2.0
```

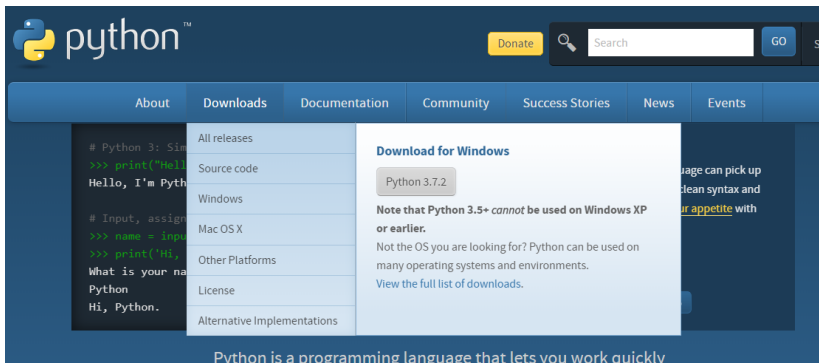
Python 3

```
print('3 / 2 =', 3 / 2)
print('3 // 2 =', 3 // 2)
print('3 / 2.0 =', 3 / 2.0)
print('3 // 2.0 =', 3 // 2.0)
```

Warto doczytać np. tutaj.

Instalacja - Windows

► <https://python.org/>



The screenshot shows the Python.org website with a dark blue header. The Python logo is on the left, and a search bar with a 'GO' button is on the right. Below the header is a navigation menu with links for 'About', 'Downloads', 'Documentation', 'Community', 'Success Stories', 'News', and 'Events'. The 'Downloads' menu is open, showing a list of options: 'All releases', 'Source code', 'Windows', 'Mac OS X', 'Other Platforms', 'License', and 'Alternative Implementations'. The 'Windows' option is selected, leading to a 'Download for Windows' section. This section features a 'Python 3.7.2' button, a note that Python 3.5+ cannot be used on Windows XP or earlier, and a link to view the full list of downloads. A code editor snippet is visible on the left side of the page, and a footer at the bottom states 'Python is a programming language that lets you work quickly'.

```
# Python 3: Simple program
>>> print("Hello, I'm Python.")
Hello, I'm Python.

# Input, assignment, and printing
>>> name = input("What is your name? ")
>>> print("Hi, " + name + ".")
Hi, Python.
```

Download for Windows

Python 3.7.2

Note that Python 3.5+ cannot be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments. [View the full list of downloads.](#)

Python is a programming language that lets you work quickly

Rysunek 1: Strona www

Linux

Sprawdzenie wersji na Ubuntu 18.04:

```
piotrekwd@piotrekwd-VirtualBox:~$ python3
Python 3.6.5 (default, Apr 1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

Ręczna instalacja:

```
sudo apt install python3
```

Wybór IDE do Pythona

- ▶ IDLE (domyślny)
- ▶ PyCharm <https://www.jetbrains.com/pycharm/> (na ćw. i wykład)
- ▶ Spyder IDE <https://www.spyder-ide.org/>
- ▶ Visual Studio
<https://visualstudio.microsoft.com/pl/vs/features/python/>
- ▶ Visual Studio Code + odpowiednie rozszerzenia
<https://code.visualstudio.com/>
- ▶ Atom + ide-python <https://atom.io/packages/ide-python>
- ▶ i wiele innych...

Styl PEP8

- ▶ wymowa: pi-i-pi-ejt
- ▶ standaryzacja kodu używana m.in. przy rozwijaniu nowych funkcjonalności
- ▶ używanie daje lepszą organizację i czytelność kod
- ▶ pełna wersja <https://www.python.org/dev/peps/pep-0008/>

Znaki odstępu:

- ▶ we wcięciach stosujemy spacje (a nie tabulatory)
- ▶ każdy poziom wcięcia powinien składać się z 4 spacji
- ▶ wiersz powinien składać się z maksymalnie 79 znaków

Dobrze:

```
foo = long_function_name(var_one, var_two,  
                          var_three, var_four)
```

```
def long_function_name(  
    var_one, var_two, var_three,  
    var_four):  
    print(var_one)
```

```
foo = long_function_name(  
    var_one, var_two,  
    var_three, var_four)
```

Žle:

```
foo = long_function_name(var_one, var_two,  
    var_three, var_four)
```

```
def long_function_name(  
    var_one, var_two, var_three,  
    var_four):  
    print(var_one)
```


Instrukcje warunkowe:

```
if (this_is_one_thing and
    that_is_another_thing):
    do_something()
```

```
if (this_is_one_thing and
    that_is_another_thing):
    # dodatkowy komentarz
    do_something()
```

```
if (this_is_one_thing
    and that_is_another_thing):
    do_something()
```

Listy:

```
my_list = [  
    1, 2, 3,  
    4, 5, 6,  
]  
result = some_function_that_takes_arguments(  
    'a', 'b', 'c',  
    'd', 'e', 'f',  
)
```

Listy - druga wersja:

```
my_list = [  
    1, 2, 3,  
    4, 5, 6,  
]  
result = some_function_that_takes_arguments(  
    'a', 'b', 'c',  
    'd', 'e', 'f',  
)
```

Operatory arytmetyczne a przenoszenie:

Źle:

```
income = (gross_wages +  
          taxable_interest +  
          (dividends - qualified_dividends) -  
          ira_deduction -  
          student_loan_interest)
```

Dobrze:

```
income = (gross_wages
          + taxable_interest
          + (dividends - qualified_dividends)
          - ira_deduction
          - student_loan_interest)
```

Puste linie:

- ▶ dwie linie między funkcjami najwyższego poziomu i między klasami.
- ▶ pojedyncza linia między funkcjami w klasie

Kodowanie:

- ▶ dla Pythona 3 sugerowane i domyślne to UTF-8.

Importowanie bibliotek

Dobrze:

```
import os  
import sys
```

Źle:

```
import sys, os
```

Ale dobrze też:

```
from subprocess import Popen, PIPE
```

Kolejność:

1. Biblioteki systemowe.
2. Biblioteki zewnętrzne tzw. third-party imports.
3. Biblioteki lokalne.

Stringi:

- ▶ można używać pojedynczych apostrofów jak i podwójnych cudzysłówów
- ▶ ważne, aby stosować wybraną notację konsekwentnie
- ▶ jedyny wyjątek to gdy wewnątrz stringu chcemy użyć cudzysłów np.

```
print('Oglądam film "Player One"')
```


Spacje w wyrażeniach:

- ▶ należy unikać ich nadużywania

Dobrze:

```
spam(ham[1], {eggs: 2})
```

Źle:

```
spam( ham[ 1 ], { eggs: 2 } )
```

Dobrze:

```
foo = (0,)
```

Źle:

```
bar = (0, )
```

Dobrze:

```
spam(1)
```

Źle:

```
spam (1)
```

Dobrze:

```
dct['key'] = lst[index]
```

Źle:

```
dct ['key'] = lst [index]
```

```
x = 1  
y = 2  
long_variable = 3
```

Žle:

```
x           = 1  
y           = 2  
long_variable = 3
```

Cechy języka Python

- ▶ Python wspiera różne paradygmaty programowania: obiektowy, imperatywny oraz funkcyjny.
- ▶ Posiada w pełni dynamiczny system typów i automatyczne zarządzanie pamięcią (garbage collector).
- ▶ Często używany jako język skryptowy. Interpretery Pythona są dostępne na wiele systemów operacyjnych. Różne implementacje Pythona: CPython (język C), IronPython (platforma .NET), Jython (Java), PyPy (Python).
- ▶ Prosta i czytelna składnia ułatwiająca utrzymywanie, używanie i rozumienie kodu.

Zen

```
import this
```

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

...

PL - [https:](https://pl.python.org/)

[//pl.python.org/forum/index.php?topic=392.msg1844#msg1844](https://pl.python.org/forum/index.php?topic=392.msg1844#msg1844)

Powstawowe typy danych w Pythonie

- ▶ liczby (int, float, complex): 22, 5.2, 2j + 9
- ▶ łańcuchy znaków (str): 'tekst1', "tekst2"
- ▶ lista (list): [3, 22, 'tekst', False]
- ▶ krotka (tuple): (6, 17, 'tekst', False)
- ▶ słownik (dict): {'klucz': 'wartość', 23: 33, 'status': False}
- ▶ typ logiczny (bool): True, False

Liczby

```
print(type(5))
```

```
## <class 'int'>
```

```
print(type(4.5))
```

```
## <class 'float'>
```

```
print(type(55+3j))
```

```
## <class 'complex'>
```

```
print(type(4e+4))
```

```
## <class 'float'>
```

```
print(type(40000))
```

```
## <class 'int'>
```


Łańcuchy znaków - stringi

```
str = 'Hello World!'
```

```
print(str)
```

```
## Hello World!
```

```
print(str[0])
```

```
## H
```

```
print(str[2:5])
```

```
## llo
```

```
print(str[2:])
```

```
## llo World!
```

```
print(str * 2)
```

```
## Hello World!Hello World!
```

```
print(str + 'WMII')
```

```
## Hello World!WMII
```

Podstawowa instrukcja wyjścia - funkcja print

Składnia wg dokumentacji:

```
print(*objects, sep=' ', end='\n', file=sys.stdout,  
      flush=False)
```

- ▶ `objects` - to co ma być wyświetlone
- ▶ `sep` - separator, domyślnie znak spacji
- ▶ `end` - co co ma być wyświetlone na końcu, domyślnie znak końca linii
- ▶ `file` - określa gdzie mają być `objects` wyświetlone, domyślnie `sys.stdout` (domyślny ekran)
- ▶ `flush` - określa czy “wyjście” ma być buforowane przed przekazaniem do `file`, domyślne `False`

```
print(1, 2, 3, 4)
```

```
## 1 2 3 4
```

```
print(1, 2, 3, 4, sep='*')
```

```
## 1*2*3*4
```

```
print(1, 2, 3, 4 ,sep='#', end='&')
```

```
## 1#2#3#4&
```

```
print('x', 'y', 'z', sep='', end='')  
print('a', 'b', 'c', sep='', end='')
```

```
## xyzabc
```

```
print('a', 'b', '\n', 'c')
```

```
## a b
```

```
## c
```

\t - przesunięcie do następnego "tab"=8 spacji

```
print('sdf', 3456, -2, sep='\t')
```

```
## sdf 3456 -2
```

\r - przesunięcie do lewej strony po każdym wyświetleniu

```
print(345, 'y', 'abc', sep='\r')
```

```
## abc
```

```
a = 16
b = 2.25
z = 45
print('{:5d} {:6.3f} {:10d}'.format(a, b, z))
```

```
##      16  2.250          45
```

```
print('{2:5d} {1:6.3f} {0:10d}'.format(a, b, z))
```

```
##      45  2.250          16
```

Input - podstawowe wejście

```
name= input('Podaj imię \n')  
print('typ:', type(name))
```

Podaj imię

Jan

typ: <class 'str'>


```
number= int(input('Podaj liczbę \n'))  
print('typ:', type(number))
```

Podaj liczbę

32

typ: <class 'int'>

Operacje arytmetyczne

```
print(5+3)
```

```
## 8
```

```
print(4*5.2)
```

```
## 20.8
```

```
print(9-7)
```

```
## 2
```

```
print(25%7)
```

```
## 4
```

```
print(4/5)
```

```
## 0.8
```

```
print(4//5)
```

```
## 0
```

```
print(4/5.0)
```

```
## 0.8
```

```
print(4//5.0)
```

```
## 0.0
```

```
print(3**0)
```

```
## 1
```

```
print(0**0)
```

```
## 1
```

```
print(4/0)
```

```
ZeroDivisionError: division by zero
```

Operacje na stringach

```
print('raz'+ ' '+ 'dwa')
```

```
## raz dwa
```

```
print('tekst'*3)
```

```
## tekstteksttekst
```

Operatory przypisania

▶ = standardowy

▶ +=, -=, *=, /=, %=, **=, //=

```
a = 5  
a += 1  
print(a)
```

6

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

36

Operatory porównania

Znak	Znaczenie	Przykład
>	większe niż	$x > y$
<	mniejsze niż	$x < y$
==	równa się	$x == y$
!=	nie równa się	$x != y$
>=	większe lub równe	$x >= y$
<=	mniejsze lub równe	$x <= y$

Operatory logiczne i typ logiczny

Operator	Znaczenie	Przykład
and	i	x and y
or	lub	x or y
not	negacja	not x

```
a = True  
b = False  
print(a or b)
```

```
## True
```

```
print(type(a))
```

```
## <class 'bool'>
```


Kolejność operatorów

Od ostatniego:

- ▶ lambda
- ▶ if - else
- ▶ or
- ▶ and
- ▶ not x
- ▶ in, not in, is, is not, <, <=, >, >=, !=, ==
- ▶ |
- ▶ ^
- ▶ &
- ▶ <<, >>
- ▶ +, -
- ▶ *, @, /, //, %
- ▶ +x, -x, ~x

- ▶ `**`
- ▶ `await x`
- ▶ `x[index]`, `x[index:index]`, `x(arguments...)`,
`x.attribute`
- ▶ `(expressions...)`, `[expressions...]`, `{key:
value...}`, `{expressions...}`

Źródło:

<https://docs.python.org/3/reference/expressions.html#operator-precedence>.

Listy

Listy w Pythonie mogą przechowywać elementy różnych typów.

```
list1 = ['raz', 'dwa', 5, 5];  
list2 = [1, 2, 3, 4, 5 ];  
list3 = ["a", "b", "c", "d"];  
print(list3)
```

```
## ['a', 'b', 'c', 'd']
```

```
list4 = ['s', 'ww', True, 5]
print(list4[3])
```

```
## 5
```

```
list4[1] = True
print(list4[1])
```

```
## True
```

```
print(list4[-1])
```

```
## 5
```

```
print(list4[2:])
```

```
## [True, 5]
```

```
print(len([2, 3, 4]))
```

```
## 3
```

```
print([1, 2, 3] + [4, 5, 6])
```

```
## [1, 2, 3, 4, 5, 6]
```

```
print(['Hi!'] * 4)
```

```
## ['Hi!', 'Hi!', 'Hi!', 'Hi!']
```

```
print(3 in [1, 2, 3])
```

```
## True
```

Pytanie do przemyślenia

Co oznacza w Pythonie, że wartości przekazywane są przez referencję?

```
a = 5  
b = a  
b += 2  
print(a)
```

```
## 5
```

```
print(b)
```

```
## 7
```

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

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

```
print(list2)
```

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

Instrukcje warunkowe

```
a = 5
if a > 0:
    print('liczba dodatnia')
elif a == 0:
    print('zero')
else:
    print('liczba ujemna')
```

```
## liczba dodatnia
```


Pętle

```
words = ['kot', 'pies', 'chomik']  
for w in words:  
    print(w, len(w))
```

```
## kot 3  
## pies 4  
## chomik 6
```

```
i = None  
for i in range(2):  
    print(i)
```

```
## 0  
## 1
```

```
i = 1
j = 1
while i < 4:
    j = 1
    while j < 4:
        print(i, '*', j, '=', i * j)
        j += 1
    i += 1
```

Bibliografia

- ▶ <https://pl.wikipedia.org/wiki/Python>, dostęp online 12.02.2019.
- ▶ <https://bulldogjob.pl/news/264-java-php-ruby-jak-wlasciwie-wymawiac-nazwy-technologiei>. 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.
- ▶ <https://www.python.org/dev/peps/pep-0008/>, dostęp online 14.02.2019.

Bibliografia - cd2

- ▶ <https://www.flynerd.pl/2017/05/python-4-typy-i-zmienne.html>, dostęp online 14.02.2019.
- ▶ <http://pytolearn.csd.auth.gr/p0-py/01/print.html>, dostęp online 15.02.2019.
- ▶ https://www.tutorialspoint.com/python3/python_lists.htm, dostęp online 17.02.2019.

Wstęp do języka Python - cd.

Struktury danych w Pythonie

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

Listy

Listy w Pythonie mogą przechowywać elementy różnych typów.

```
list1 = ['raz', 'dwa', 5, 5];  
list2 = [1, 2, 3, 4, 5 ];  
list3 = ["a", "b", "c", "d"];  
print(list3)
```

```
## ['a', 'b', 'c', 'd']
```

```
list4 = ['s', 'ww', True, 5]
print(list4[3])
```

```
## 5
```

```
list4[1] = True
print(list4[1])
```

```
## True
```

```
print(list4[-1])
```

```
## 5
```

```
print(list4[2:])
```

```
## [True, 5]
```



```
print(len([2, 3, 4]))
```

```
## 3
```

```
print([1, 2, 3] + [4, 5, 6])
```

```
## [1, 2, 3, 4, 5, 6]
```

```
print(['Hi!'] * 4)
```

```
## ['Hi!', 'Hi!', 'Hi!', 'Hi!']
```

```
print(3 in [1, 2, 3])
```

```
## True
```

```
lista = ['a', 'b', 34, 5.6, True]
lista.append('5')
print(lista)
```

```
## ['a', 'b', 34, 5.6, True, '5']
```

```
lista.extend([4, 5, 6])
print(lista)
```

```
## ['a', 'b', 34, 5.6, True, '5', 4, 5, 6]
```

```
lista.insert(2, 'w')  
print(lista)
```

```
## ['a', 'b', 'w', 34, 5.6, True, '5', 4, 5, 6]
```

```
lista.remove(True)  
print(lista)
```

```
## ['a', 'b', 'w', 34, 5.6, '5', 4, 5, 6]
```

```
lista.pop()
```

```
## 6
```

```
print(lista)
```

```
## ['a', 'b', 'w', 34, 5.6, '5', 4, 5]
```

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

```
## 5.6
```

```
print(lista)
```

```
## ['a', 'b', 'w', 34, '5', 4, 5]
```

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

```
## 4
```

```
print(lista)
```

```
## ['a', 'b', 'w', 34, '5', 5]
```

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

```
## 'a'
```

```
print(lista)
```

```
## ['b', 'w', 34, '5', 5]
```

```
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)
```

```
## {'raz', 'trzy', 'dwa', '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']
```


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 2: 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
```

Odpowiedź na pytanie

“Mutable” - zmienne typy::

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

“Inmutable” - niezmiennie typy:

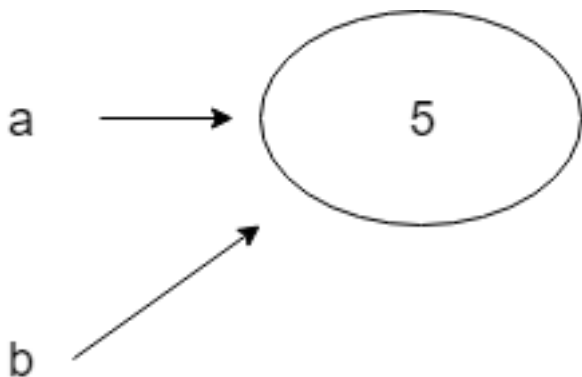
- ▶ int
- ▶ float
- ▶ decimal
- ▶ complex
- ▶ bool
- ▶ string
- ▶ tuple
- ▶ range
- ▶ frozenset
- ▶ bytes


```
a = 5
b = a
b += 2
print(a)
```

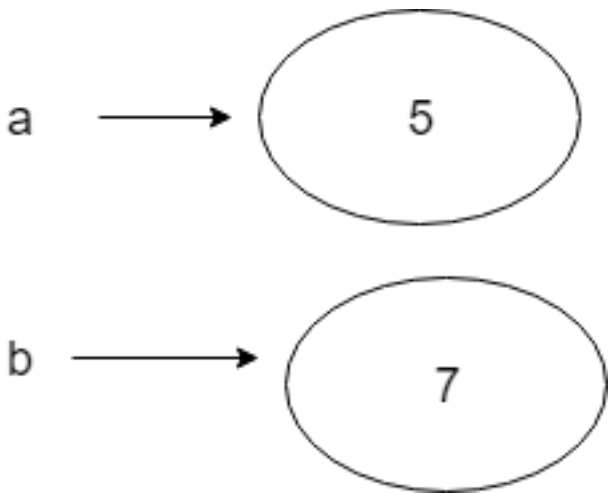
```
## 5
```

```
print(b)
```

```
## 7
```



Rysunek 3: Dwie pierwsze linijki



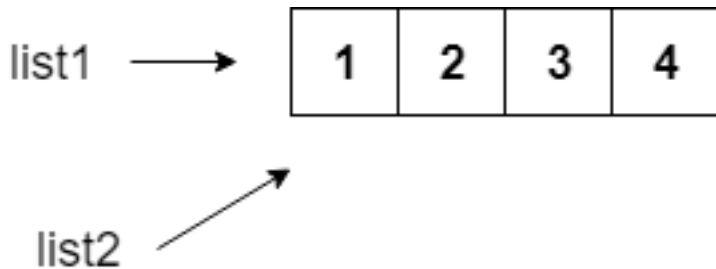
Rysunek 4: $b=+2$

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

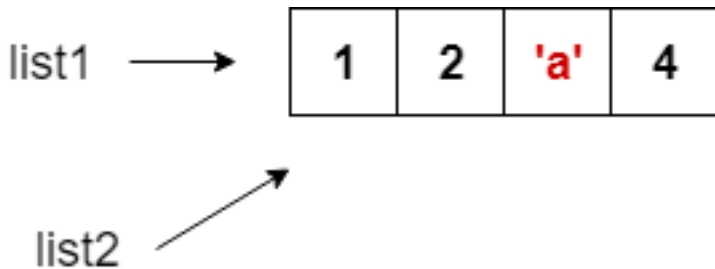
```
## [1, 2, 'a', 4]
```

```
print(list2)
```

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



Rysunek 5: Dwie pierwsze linijki.



Rysunek 6: `list1[2] = 'a'`

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>