

Języki i techniki programowania

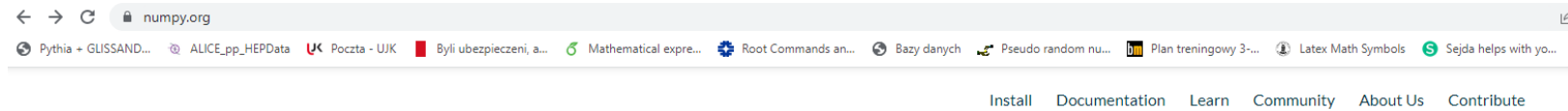
Wykład 4

Maciej Rybczyński

NumPy

NumPy (Numerical Python for scientific computing)

<https://numpy.org>



The fundamental package for scientific computing with Python

GET STARTED

NumPy 1.23.0 released

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

INTEROPERABLE

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

PERFORMANT

The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

EASY TO USE

NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

OPEN SOURCE

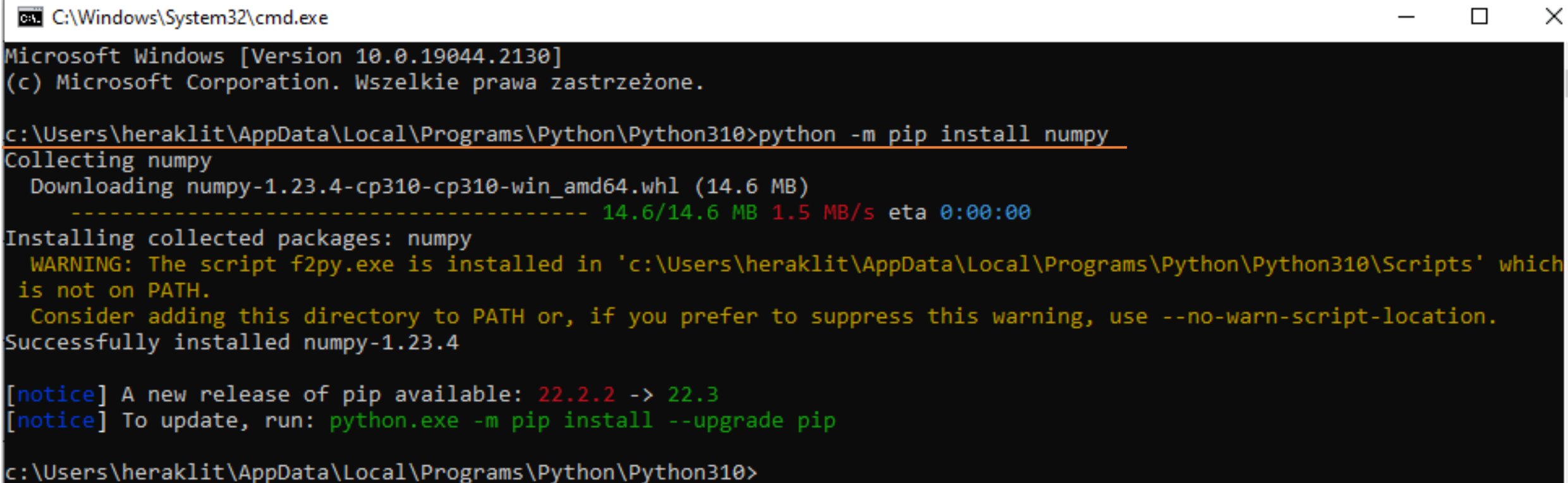
Distributed under a liberal [BSD license](#), NumPy is developed and maintained [publicly on GitHub](#) by a vibrant, responsive, and diverse [community](#).

Try NumPy

Use the interactive shell to try NumPy in the browser

Otwórz wiersz poleceń,
przejdź do katalogu gdzie jest zainstalowany python

Łatwa instalacja!



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

c:\Users\heraklit\AppData\Local\Programs\Python\Python310>python -m pip install numpy
Collecting numpy
  Downloading numpy-1.23.4-cp310-cp310-win_amd64.whl (14.6 MB)
    ----- 14.6/14.6 MB 1.5 MB/s eta 0:00:00
Installing collected packages: numpy
  WARNING: The script f2py.exe is installed in 'c:\Users\heraklit\AppData\Local\Programs\Python\Python310\Scripts' which
is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed numpy-1.23.4

[notice] A new release of pip available: 22.2.2 -> 22.3
[notice] To update, run: python.exe -m pip install --upgrade pip

c:\Users\heraklit\AppData\Local\Programs\Python\Python310>
```

wpisz polecenie:
python -m pip install numpy

Wymaganie połączenie z internetem

NumPy - array

 *lec_4a.py - E:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wykklad_4/lec_4a.py (3.10.7)*

File Edit Format Run Options Window Help

```
import numpy as np
```

```
L = [1.0, 2.2, 3.1, 5.5]
```

```
a = np.array(L)
```

```
print(a)
```

```
print(a.shape)           # rozmiar tablicy (w kazdym wymiarze)
```

```
print(a.ndim)           # liczba osi (wymiarów) tablicy
```

```
print(a.size)           # liczba wszystkich elementów tablicy
```

```
print(a.dtype.name)     # typ elementów tablicy
```

```
= RESTART: E:/CERNBox/zajecia/Inzynieria_danych
```

```
[1.  2.2 3.1 5.5]
```


```
(4,)
```

```
1
```

```
4
```

```
float64
```

NumPy - array

 lec_4b.py - E:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wykklad_4/lec_4b.py (3.10.7)

File Edit Format Run Options Window Help

```
import numpy as np
```

```
L = [[1,2,3], [4,5,6]]
```

```
a = np.array(L)          # dwu-wymiarowa tablica
```

```
print(a)
```

```
print(a.shape)          # rozmiar tablicy (w kazdym wymiarze)
```

```
print(a.ndim)           # liczba osi (wymiarów) tablicy
```

```
print(a.size)           # liczba wszystkich elementów tablicy
```

```
print(a.dtype.name)     # typ elementów tablicy
```

```
= RESTART: E:/CERNBox/zaj
```

```
[[1 2 3]
```

```
 [4 5 6]]
```

```
(2, 3)
```

```
2
```

```
6
```

```
int32
```

Możemy określić typ elementów tablicy

 lec_4c.py - E:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wykklad_4/lec.

File Edit Format Run Options Window Help

```
import numpy as np
```

```
print(np.array([1,2,3], dtype=float))
```

```
print(np.array([1.0,2.0,3.0], dtype=int))
```

```
print(np.array([[1,2,3], [4,5,6]], dtype=complex))
```

```
===== RESTART: E:/CERNBox/zaj  
[1.  2.  3.]  
[1  2  3]  
[[1.+0.j  2.+0.j  3.+0.j]  
 [4.+0.j  5.+0.j  6.+0.j]]  
>>>
```

Dlaczego NumPy jest użyteczny?

```
import numpy as np
```

```
a = np.array([1,2,3])
```

```
print(a*10)
```

```
print(a/2)
```

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

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

```
print(a + 10.5)
```

```
print(np.cos(a))    # np.cos(a) - używa funkcji z biblioteki NumPy
```

```
print(a > 0)
```

operatory arytmetyczne zastosowane „na tablicach”
działają **w odniesieniu do elementów!**

```
[10 20 30]
```

```
[0.5 1.  1.5]
```

```
[1 4 9]
```

```
[1 4 9]
```

```
[11.5 12.5 13.5]
```

```
[ 0.54030231 -0.41614684 -0.9899925 ]
```

```
[ True  True  True]
```


Dlaczego NumPy jest użyteczny?

```
import numpy as np

a = np.array([1, 2, 3])

b = np.array([5.0, 4.0, 3.0])

print(a + b)
print(a * b)
print(b / a)
print(a > b)
print((a - b) == 0)
```

```
[6.  6.  6.]
[5.  8.  9.]
[5.  2.  1.]
[False False False]
[False False  True]
```

Szybkie obliczenia

```
import numpy as np
import time
```

```
L = range(1,10**7)
L = np.array(L)
print(L)
```

```
start = time.time()
G1 = []
for i in L:
    G1.append(i + 10)
print(time.time() - start)
```

```
start = time.time()
G2 = L + 10
print(time.time() - start)
```

```
===== RESTART: E:/CERNBox/zajecia/Inzynieria_danych/Py
[          1          2          3 ... 99999997 99999998 99999999]
1.1828365325927734
0.007982969284057617
```

~150 razy szybciej!

Dlaczego NumPy jest użyteczny?

```
import numpy as np
```

```
def fun(x, y):  
    print("let's calculate")  
    return np.sqrt(x*y)
```

```
a = np.array([4,2,3])
```

```
b = np.array([4,6,7.0])
```

```
print(fun(a,b))
```

```
let's calculate
```

```
[4.          3.46410162  4.58257569]
```

zeros, ones

```
import numpy as np
```

```
print(np.zeros(10))
```

```
print(np.zeros(10, dtype=int))
```

```
print(np.ones(10))
```

```
print(np.ones(10, dtype=int))
```

```
[0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

```
[0 0 0 0 0 0 0 0 0 0]
```

```
[1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]
```

```
[1 1 1 1 1 1 1 1 1 1]
```

domyślnie float64

zeros, ones

```
import numpy as np
```

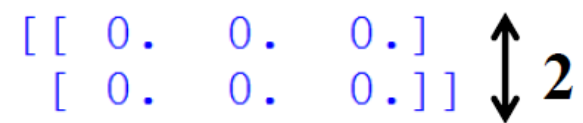
```
print(np.zeros( (2, 3) ))
```

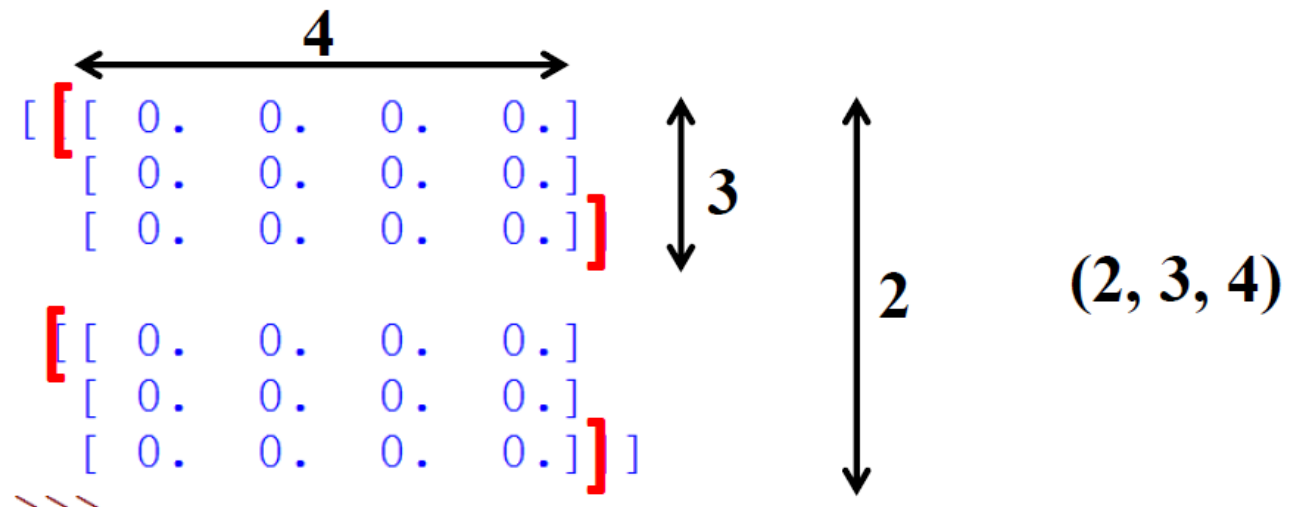
```
print('\n')
```

```
print(np.zeros( (2, 3, 4) ))
```

```
>>> ===== RESTART =====
```

```
>>> 
```

```

```

```

```

arange

```
import numpy as np
```

```
a = np.arange(2, 10, 2)
```

```
print(a)
```

```
b = np.arange(0, 1, 0.1)
```

```
print(b)
```

jak **range()** w pythonie, ale lepsze

```
>>> | [2 4 6 8]
    | [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]
```

b jest równoważne: **[i/10 for i in range(0,10)]**

reshape

```
import numpy as np
```

```
a = np.arange(0, 10, 1)
```

```
print(a)
```

```
print('\n')
```

```
b = a.reshape(5, 2)
```

5 wierszy, 2 kolumny

```
print(b)
```

```
>>>
```

```
[0 1 2 3 4 5 6 7 8 9]
```

```
[[0 1]
 [2 3]
 [4 5]
 [6 7]
 [8 9]]
```

5

2

sum, min, max

```
import numpy as np
```

```
a = np.array([1, 2, 3, 4, 5])
```

```
print(np.sum(a))
```

```
print(np.min(a))
```

```
print(np.max(a))
```

```
>>> 15  
      1  
      5
```

inny sposób:

a.sum(), a.min(), a.max()

NumPy – liczby losowe

```
import numpy as np


print(np.random.random(3))          # 3 liczby rzeczywiste z przedziału [0,1)
print(np.random.uniform(0, 10, 5))  # 5 l. rzecz. z [0,10)
print(np.random.randint(0, 5, (2,21))) # tablica (2,21) liczb całkowitych z przedziału [0, 5)

[0.19638011  0.95885895  0.94762488]
[8.19532256  9.97263402  2.10617271  6.36093885  1.3557365 ]
[[4 1 3 1 3 0 1 2 1 0 0 4 1 1 4 0 0 3 4 2 0]
 [2 2 3 1 4 0 3 0 2 3 3 2 3 1 0 1 0 0 3 1 0]]
>>>
```

https://numpy.org/doc/stable/

numpy.org/doc/stable/

SSAND... ALICE_pp_HEPData Poczta - UJK Byli ubezpieczeni, a... Mathematical expe... Root Commands an... Bazy danych Pseudo random nu... Plan treningowy 3-... Latex Math Symbols Se

User Guide API reference Development Release notes Learn

1.23 (stable)

Search the docs ...


NumPy documentation

Version: 1.23

Download documentation: [PDF Version](#) | [Historical versions of documentation](#)

Useful links: [Installation](#) | [Source Repository](#) | [Issue Tracker](#) | [Q&A Support](#) | [Mailing List](#)


NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.



Getting Started

New to NumPy? Check out the Absolute Beginner's Guide. It contains an introduction to NumPy's main concepts and links to additional tutorials.

[To the absolute beginner's guide](#)



User Guide

The user guide provides in-depth information on the key concepts of NumPy with useful background information and explanation.

[To the user guide](#)