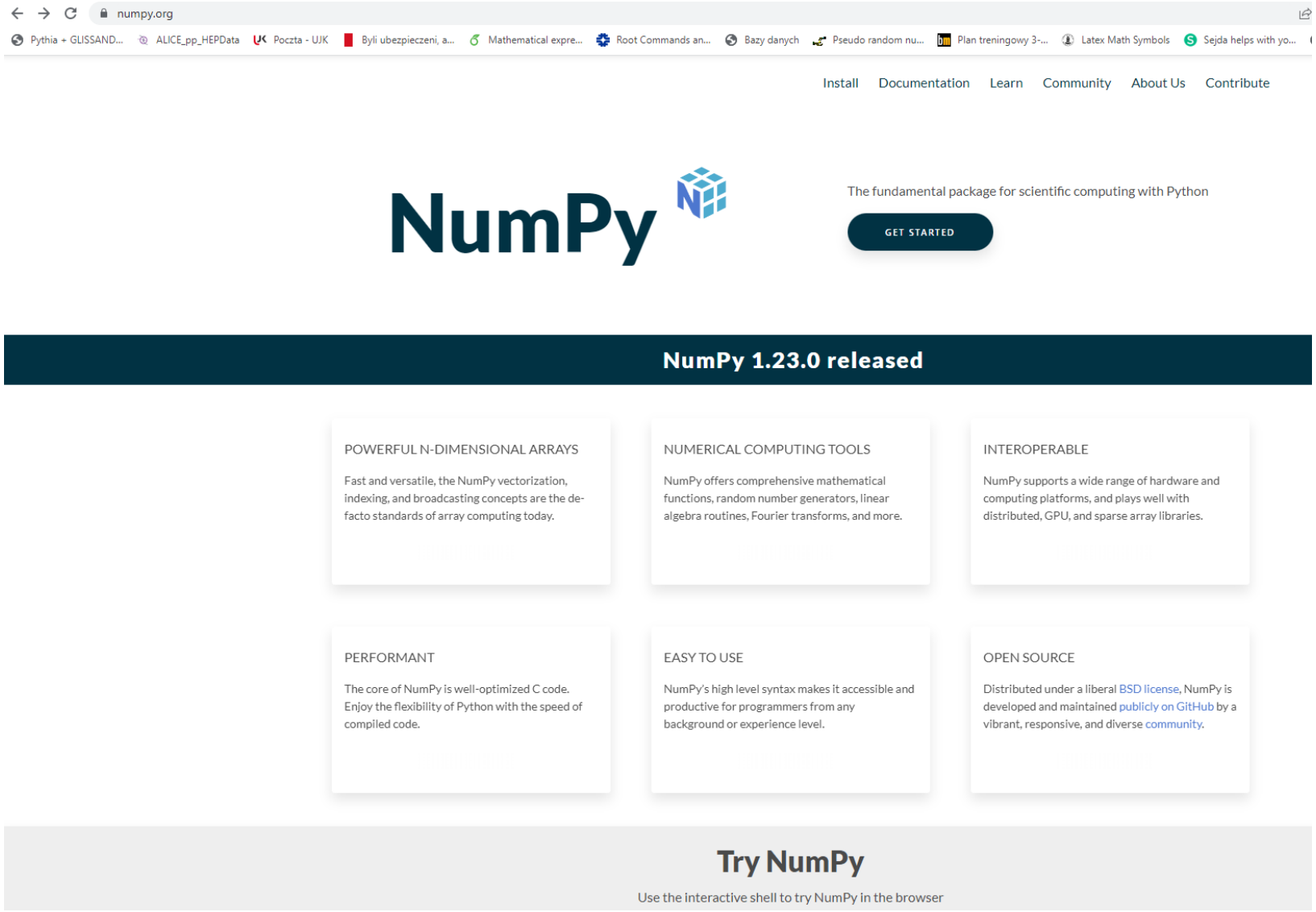
Języki i techniki programowania

Wykład 4

Maciej Rybczyński

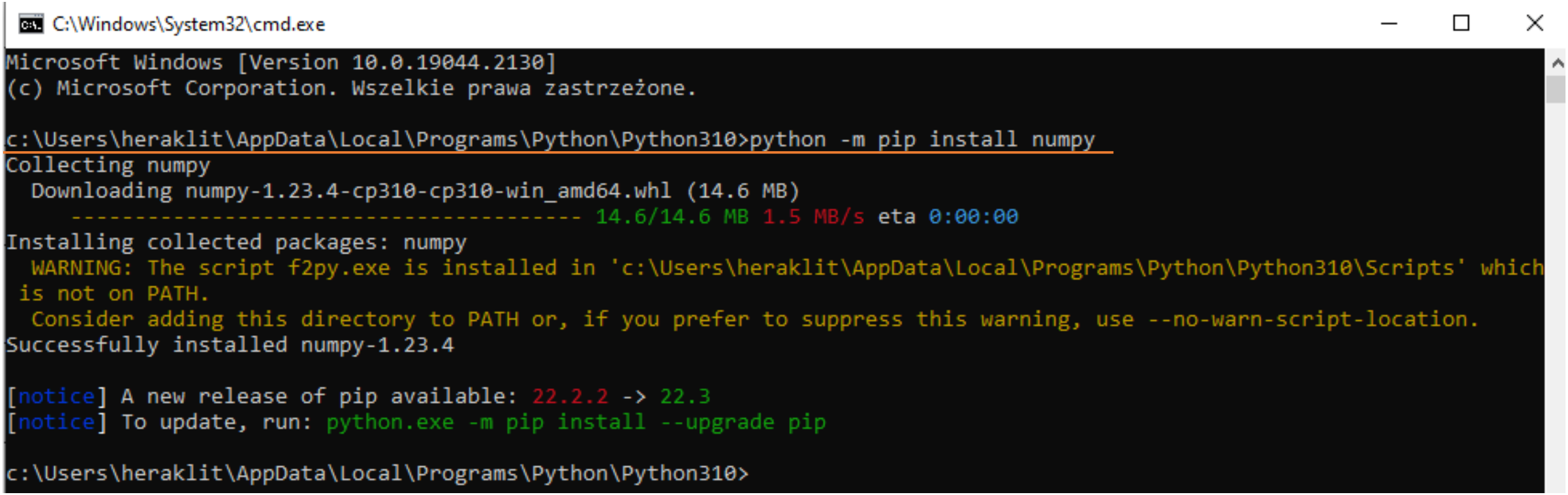
NumPy

NumPy (Numerical Python for scientific computing)
https://numpy.org



Otwórz wiersz poleceń,
przejdź do katalogu gdzie jest zainstalowany python

Łatwa instalacja!



wpisz polecenie:
python –m pip install numpy

Wymaganie połączenie z internetem

NumPy - array

lec_4a.py - E:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wyklad_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/wyklad_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/wyklad_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 ... 9999997 9999998 9999999]
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

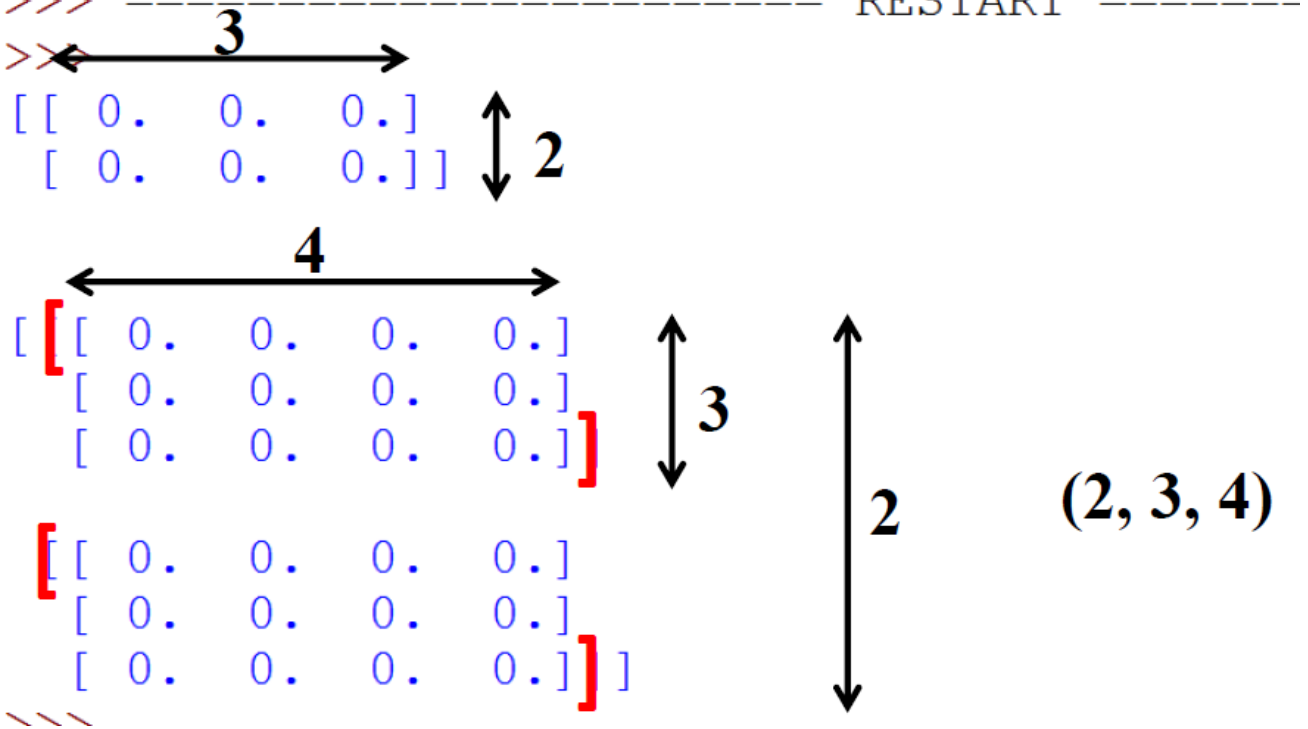
```
import numpy as np

print(np.zeros((2,3)))
print('\n')
print(np.zeros((2,3,4)))
```

```
>>> ========================= RESTART =======
>>>
[[ 0.  0.  0.]
 [ 0.  0.  0.]]

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

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



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

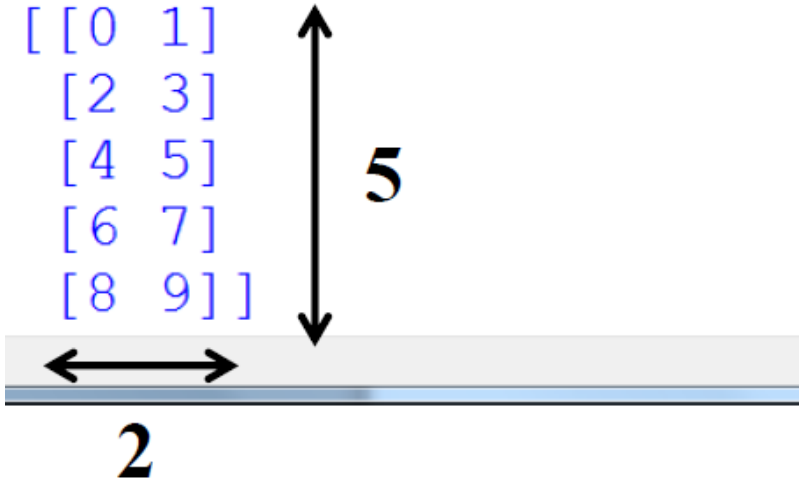
```
import numpy as np

a = np.arange(0, 10, 1)
print(a)
print('\n')
b = a.reshape(5, 2)
print(b)
```

5 wierszy, 2 kolumny

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

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



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/