# Wizualizacja danych - wykład 3

dr Piotr Jastrzębski

# Biblioteka NumPy

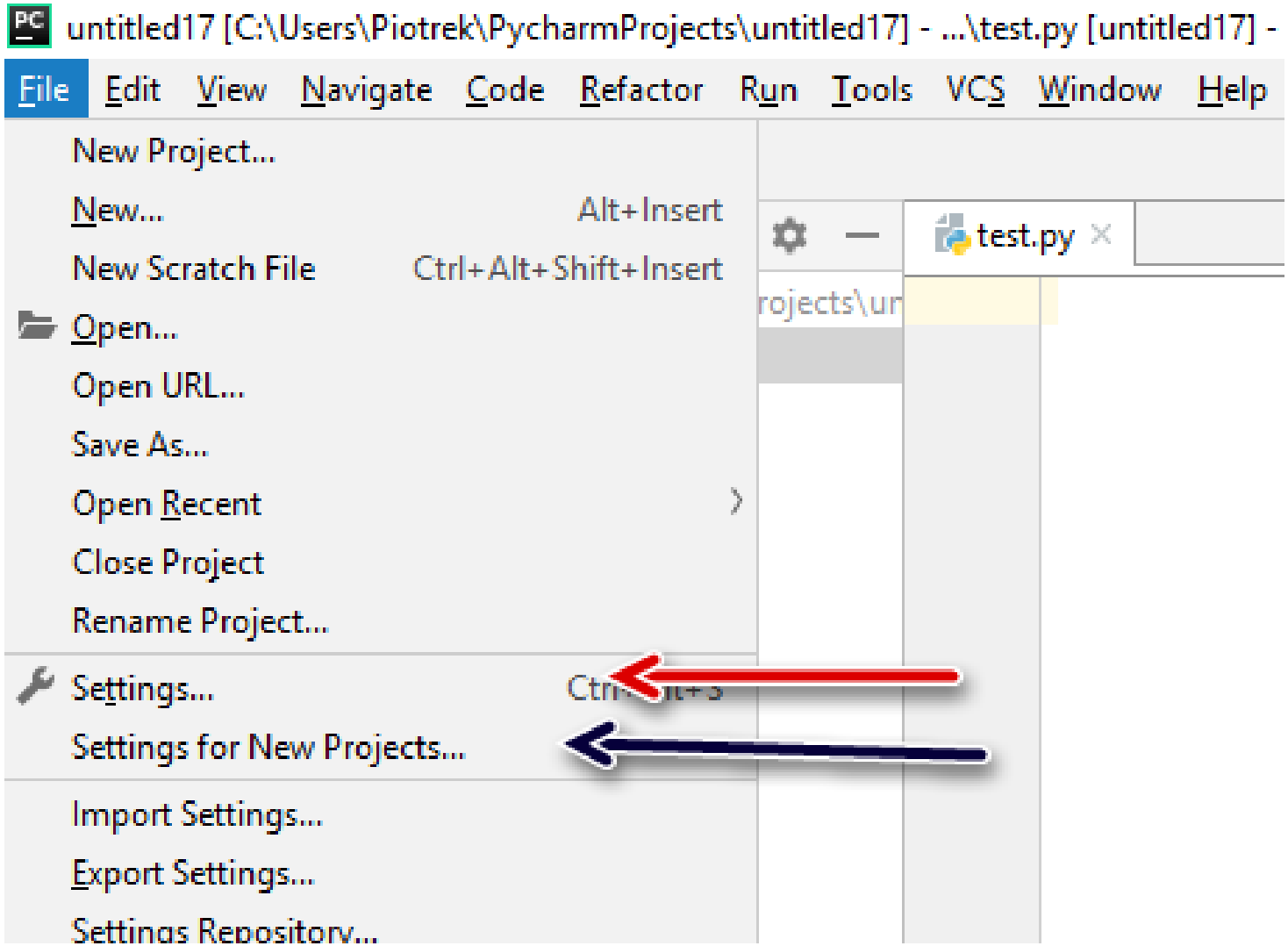
# NumPy

NumPy jest biblioteką Pythona służącą do obliczeń naukowych.

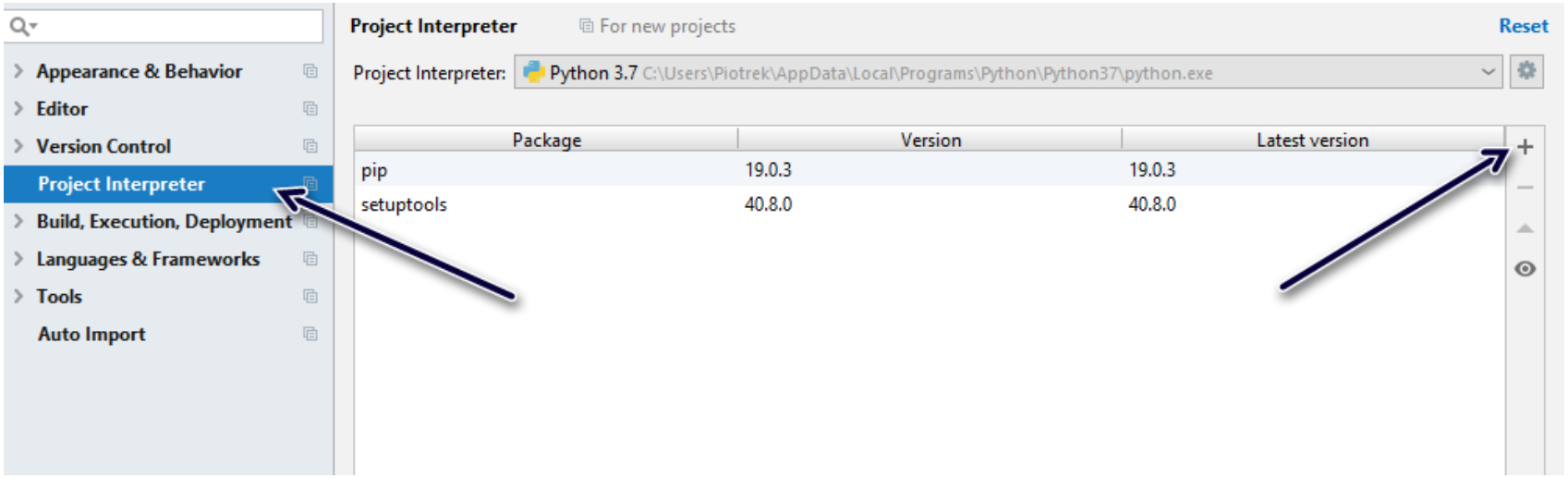
Zastosowania:

- algebra liniowa
- zaawansowane obliczenia matematyczne (numeryczne)
- całkowania
- rozwiązywanie równań
- ...

# Instalacja NumPy w PyCharm




**Rysunek 1:**


Project Interpreter
For new projects
Reset
Appearance & Behavior
Editor
Version Control
Project Interpreter
Build, Execution, Deployment
Languages & Frameworks
Tools
Auto Import
Project Interpreter:
Python 3.7 C:\Users\Piotrek\AppData\Local\Programs\Python\Python37\python.exe
Package
Version
Latest version
pip
19.0.3
19.0.3
setuptools
40.8.0
40.8.0

**Rysunek 2:**

Podstawowym bytem w bibliotece NumPy jest N-wymiarowa tablica zwana `ndarray`. Każdy element na tablicy traktowany jest jako typ `dtype`.

```
numpy.array(object, dtype=None, copy=True,
            order='K', subok=False, ndmin=0)
```

- object - to co ma być wrzucone do tablicy
- dtype - typ
- copy - czy obiekty mają być skopiowane, domyślne `True`
- order - sposób układania: C (rzędy), F (kolumny), A, K
- subok - realizowane przez podklasy (jeśli `True`), domyślnie `False`
- ndmin - minimalny rozmiar (wymiar) tablicy

```
import numpy as np
a = np.array([1, 2, 3])
print(a)
```

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

```
b = np.array([1, 2, 3.0])
print(b)
```

```
## [1. 2. 3.]
```

```
c = np.array([[1, 2], [3, 4]])
print(c)
```

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

```
d = np.array([1, 2, 3], ndmin=2)
print(d)
```

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

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

```
## [1.+0.j 2.+0.j 3.+0.j]
```

```
f = np.array(np.mat('1 2; 3 4'))
print(f)
```

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

```
g = np.array(np.mat('1 2; 3 4'), subok=True)
print(g)
```

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

Typy danych:

- link1.
- link2.

```
dt = np.dtype(np.int32)
print(dt)
```

```
## int32
```

```
dt = np.dtype('i4')
print(dt)
```

```
## int32
```

```
dt = np.dtype('f8')
print(dt)
```

```
## float64
```

```
dt = np.dtype('c16')
print(dt)
```

```
## complex128
```

```
dt = np.dtype('a25')
print(dt)
```

```
## |S25
```

```
dt = np.dtype('U25')
print(dt)
```

```
## <U25
```

ndarray.shape - wymiary tablicy

```
x = np.array([1, 2, 3, 4])
print(x.shape)
```

```
## (4,)
```

```
y = np.zeros((2, 3, 4))
print(y.shape)
```

```
## (2, 3, 4)
```

```
print(y)
```

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

```
y.shape = (3, 8)
print(y)
```

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

```
y.shape = (3, 6)
```

```
## ValueError: cannot reshape array of size 24 into shape (
```

# Tworzenie tablicy

Składnia: `numpy.empty(shape, dtype = float, order = 'C')`

```
a = np.empty([2, 2])
print(a)
```

```
## [[3.23295009e-315 9.88131292e-324]
##  [3.25992679e-315 3.25992671e-315]]
```

```
b = np.empty([2, 4], dtype=int)
print(b)
```

```
## [[654356384         0         2         0]
##  [659816528         0 659816512         0]]
```

Składnia: `numpy.zeros(shape, dtype=float, order='C')`

```
a = np.zeros(5)
print(a)
```

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

```
b = np.zeros((5,), dtype=int)
print(b)
```

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

```
a = np.zeros((2, 1))
print(a)
```

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

```
s = (2, 2)
b = np.zeros(s)
print(b)
```

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

Składnia: `numpy.ones(shape, dtype=None, order='C')`

```
a = np.ones(5)
print(a)
```

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

```
s = (2, 2)
b = np.ones(s)
print(b)
```

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

Składnia: `numpy.asarray(a, dtype=None, order=None)`

```
a = [1, 2]
t1 = np.asarray(a)
print(t1)
```

```
## [1 2]
```

Składnia: `numpy.arange([start, ]stop, [step, ]dtype=None)`

```
a = np.arange(3)
print(a)
```

```
## [0 1 2]
```

```
b = np.arange(3.0)
print(b)
```

```
## [0. 1. 2.]
```

```
c = np.arange(3, 7)
print(c)
```

```
## [3 4 5 6]
```

```
d = np.arange(3, 11, 2)
print(d)
```

```
## [3 5 7 9]
```

Składnia: `numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)`

```
a = np.linspace(2.0, 3.0, num=5)
print(a)
```

```
## [2.   2.25 2.5  2.75 3.  ]
```

```
b = np.linspace(2.0, 3.0, num=5, endpoint=False)
print(b)
```

```
## [2.  2.2 2.4 2.6 2.8]
```

```
c = np.linspace(2.0, 3.0, num=5, retstep=True)
print(c)
```

```
## (array([2.  , 2.25, 2.5 , 2.75, 3.  ]), 0.25)
```

Składnia: `numpy.logspace(start, stop, num=50, endpoint=True, base=10.0, dtype=None, axis=0)`

```
a = np.logspace(2.0, 3.0, num=4)
print(a)
```

```
## [ 100.          215.443469    464.15888336 1000.
```

```
b = np.logspace(2.0, 3.0, num=4, endpoint=False)
print(b)
```

```
## [100.          177.827941    316.22776602 562.34132519]
```

```
c = np.logspace(2.0, 3.0, num=4, base=2.0)
print(c)
```

```
## [4.          5.0396842   6.34960421 8.        ]
```

# Indeksowanie i “krojenie”

```
x = np.arange(10)
print(x[2])
```

```
## 2
```

```
print(x[-3])
```

```
## 7
```

```
x.shape = (2, 5)
print(x)
```

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

```
print(x[1, 3])
```

```
## 8
```

```
print(x[1, -1])
```

```
## 9
```

```
print(x[1])
```

```
## [5 6 7 8 9]
```

```
print(x[0][2])
```

```
## 2
```

# Ineksowanie wg tablicy

```
x = np.arange(10, 1, -1)
print(x)
```

```
## [10  9  8  7  6  5  4  3  2]
```

```
y = x[np.array([3, 3, 1, 8])]
print(y)
```

```
## [7 7 9 2]
```

```
a =  x[np.array([3,3,-3,8])]
print(a)
```

```
## [7 7 4 2]
```

```
b = x[np.array([[1,1],[2,3]])]
print(b)
```

```
## [[9 9]
##  [8 7]]
```

```
y = np.arange(35).reshape(5,7)
print(y)
```

```
## [[ 0  1  2  3  4  5  6]
##  [ 7  8  9 10 11 12 13]
##  [14 15 16 17 18 19 20]
##  [21 22 23 24 25 26 27]
##  [28 29 30 31 32 33 34]]
```

```
z=y[np.array([0,2,4]), np.array([0,1,2])] ## [W.,kol.]
print(z)
```

```
## [ 0 15 30]
```

```
b = y>20
print(y[b])
```

```
## [21 22 23 24 25 26 27 28 29 30 31 32 33 34]
```

# Operacja na macierzach

```
a = np.array([1, 2, 3, 4])
b = np.array([10, 20, 30, 40])
c = a * b
print(c)
```

```
## [ 10  40  90 160]
```

```
a = np.array([[0.0,0.0,0.0],[10.0,10.0,10.0],
              [20.0,20.0,20.0],[30.0,30.0,30.0]])
b = np.array([1.0,2.0,3.0])
print(a+b)
```

```
## [[ 1.  2.  3.]
##  [11. 12. 13.]
##  [21. 22. 23.]
##  [31. 32. 33.]]
```

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

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

```
b = a.T
print(b)
```

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

```
c = np.transpose(a)
print(c)
```

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

```
a = np.arange(6).reshape((3, 2))
print(a)
```

```
## [[0 1]
##  [2 3]
##  [4 5]]
```

```
x = np.arange(1, 7).reshape(2, 3)
f = x.flat
print(f)
```

```
## <numpy.flatiter object at 0x000000000270461C0>
```

```
print(x.flat[3])
```

```
## 4
```

```
a = np.array([[1,2], [3,4]])
f = a.flatten()
print(f)
```

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

# Pytanie na kolejny wykład?

Która funkcja/operator odpowiada "matematycznemu" mnożeniu macierzy?

- gwiazdka *
- numpy.dot - link
- numpy.multiply - link2
- numpy.matmul - link3

# Bibliografia


- https://www.tutorialspoint.com/numpy/index.htm, dostęp online 20.03.2019.
- https://docs.scipy.org/doc/numpy/reference/, dostęp online 20.03.2019.