Numpy - wstęp

In [1]:
```python
import numpy as np
```

In [2]:
```python
a = np.array([[[1, 2, 3]]])
print(type(a))
b = [1, 2, 3]
print(type(b))
print(a.shape) #wymiary
```
```
<class 'numpy.ndarray'>
<class 'list'>
(1, 1, 3)
```

In [4]:
```python
a = np.array([1, 2, 3])
print(a[0], a[1], a[2])
a[0] = 5
print(a)
```
```
1 2 3
[5 2 3]
```

In [5]:
```python
b = np.array([[1,2,3],[4,5,6]])
print(b.shape)
print(b[0, 0], b[0, 1], b[1, 0])
print(b[0])
print(b)
```
```
(2, 3)
1 2 4
[1 2 3]
[[1 2 3]
 [4 5 6]]
```

In [6]:
```python
a = np.zeros((5,2,3))
print(a)
```
```
[[[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.]]]
```

In [7]:
```python
b = np.ones((1,2))
print(b)
```
```
[[1. 1.]]
```

In [8]:
```python
c = np.full((2,2), 7)
print(c)
```
```
[[7 7]
 [7 7]]
```

In [9]:
```python
d = np.eye(5)
print(d)
```
```
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
```

In [10]:
```python
e = np.random.random((2,2))
print(e)
```
```
[[0.33710166 0.24716308]
 [0.51206397 0.96574697]]
```

Indeksowanie

In [11]:
```python
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print(a)
```
```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

In [12]:
```python
b = a[:2, 1:3] #zewnętrzny wymiar, wewnętrzny wymiar
print(b)
print(a[0, 1])
b[0, 0] = 77
print(a[0, 1])
```
```
[[2 3]
 [6 7]]
2
77
```

In [13]:
```python
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
a
```
Out[13]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [14]:
```python
row_r1 = a[1, :]
row_r2 = a[1:2, :]
print(row_r1, row_r1.shape)
print(row_r2, row_r2.shape)
```
```
[5 6 7 8] (4,)
[[5 6 7 8]] (1, 4)
```

In [15]:
```python
col_r1 = a[:, 1]
col_r2 = a[:, 1:2]
print(col_r1, col_r1.shape)
print(col_r2, col_r2.shape)
```
```
[ 2  6 10] (3,)
[[ 2]
 [ 6]
 [10]] (3, 1)
```

In [16]:
```python
a = np.array([[1,2], [3, 4], [5, 6]])
print(a)
print(a[[0, 1, 2], [0, 1, 0]])

print(np.array([a[0, 0], a[1, 1], a[2, 0]]))
```
```
[[1 2]
 [3 4]
 [5 6]]
[1 4 5]
[1 4 5]
```

In [17]:
```python
print(a[[0, 0], [1, 1]])

print(np.array([a[0, 1], a[0, 1]]))
```
```
[2 2]
[2 2]
```

In [18]:
```python
a = np.arange(10)
print(a)
```
```
[0 1 2 3 4 5 6 7 8 9]
```

In [19]:
```python
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
print(a)

b = np.array([0, 2, 0, 1])
print(b)
print(a[np.arange(4), b])
```
```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[0 2 0 1]
[ 1  6  7 11]
```

In [20]:
```python
a[np.arange(4), b] += 10

print(a)
```
```
[[11  2  3]
 [ 4  5 16]
 [17  8  9]
 [10 21 12]]
```

In [21]:
```python
lista_zwykla = [1,2,3,4]
lista_zwykla_2 = [5,6,7,8]

lista_numpy = np.array([1,2,3,4])
lista_numpy_2 = np.array([5,6,7,8])
print(lista_zwykla + lista_zwykla_2)
print(lista_numpy + lista_numpy_2)
```
```
[1, 2, 3, 4, 5, 6, 7, 8]
[ 6  8 10 12]
```

Indeksowanie warunkowe

In [24]:
```python
a = np.array([[1,2], [3, 4], [5, 6]])
print(a)
bool_idx = (a > 2)
print(bool_idx)
```
```
[[1 2]
 [3 4]
 [5 6]]
[[False False]
 [ True  True]
 [ True  True]]
```

In [25]:
```python
print(a[bool_idx])

print(a[a > 2])
```
```
[3 4 5 6]
[3 4 5 6]
```

Typy danych

In [27]:
```python
x = np.array([1, 2])
print(x.dtype)

x = np.array([1.0, 2.0])
print(x.dtype)

x = np.array([1, 2], dtype=np.float64)
print(x.dtype)
```
```
int32
float64
float64
```

In [28]:
```python
x = np.array(["Ala", "kot"])
print(x.dtype)
```
```
<U3
```

Operacje na macierzach

In [29]:
```python
x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)
print(x)
print(y)
print(x + y)
print(np.add(x, y))
```
```
[[1. 2.]
 [3. 4.]]
[[5. 6.]
 [7. 8.]]
[[ 6.  8.]
 [10. 12.]]
[[ 6.  8.]
 [10. 12.]]
```

In [30]:
```python
lista_1 = [[1,2],[3,4]] #python
lista_2 = [[5,6],[7,8]]
print(lista_1 + lista_2)
```
```
[[1, 2], [3, 4], [5, 6], [7, 8]]
```

In [31]:
```python
print(x - y)
print(np.subtract(x, y))
```
```
[[-4. -4.]
 [-4. -4.]]
[[-4. -4.]
 [-4. -4.]]
```

In [32]:
```python
print(x * y)
print(np.multiply(x, y))
```
```
[[ 5. 12.]
 [21. 32.]]
[[ 5. 12.]
 [21. 32.]]
```

In [33]:
```python
print(x / y)
print(np.divide(x, y))
```
```
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
```

In [34]:
```python
print(np.sqrt(x))
```
```
[[1.         1.41421356]
 [1.73205081 2.        ]]
```

In [ ]:
```python

```

In [35]:
```python
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

v = np.array([9,10])
w = np.array([11, 12])
```

In [36]:
```python
print(v.dot(w))
print(np.dot(v, w))
```
```
219
219
```

In [37]:
```python
print(x.dot(v))
print(np.dot(x, v))
```
```
[29 67]
[29 67]
```

In [38]:
```python
print(x.dot(y))
print(np.dot(x, y))
```
```
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
```

In [ ]:
```python

```

In [39]:
```python
x = np.array([[1,2],[3,4]])

print(np.sum(x))
print(np.sum(x, axis=0))
print(np.sum(x, axis=1))
```
```
10
[4 6]
[3 7]
```

In [40]:
```python
x = np.array([[1,2], [3,4]])
print(x)

print(x.T)

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

Broadcasting

In [43]:
```python
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = np.empty_like(x)
print(y)
for i in range(4):
    y[i, :] = x[i, :] + v

print(y)
```
```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

In [44]:
```python
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
vv = np.tile(v, (4, 1))
print(vv)
print(x)
y = x + vv
print(y)
```
```
[[1 0 1]
 [1 0 1]
 [1 0 1]
 [1 0 1]]
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

In [54]:
```python
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = x + v
print(y)
```
```
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

Porównanie Python vs Numpy

In [45]:
```python
def pure_python(n):
    X = range(n)
    Y = range(n)
    Z = []
    for i in range(n):
        Z.append(X[i]+Y[i])

def pure_numpy(n):
    X = np.arange(n)
    Y = np.arange(n)
    Z = X + Y

%timeit pure_python(100000)
%timeit pure_numpy(100000)
```
```
43 ms ± 1.21 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
715 µs ± 84.3 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

Materiały

- https://docs.scipy.org/doc/numpy/reference/