Numpy!

This section is all about the package, Numpy. Are you ready? Let's go!

You will find some small tasks in sections below. Most codes are hidden and you can only see the output.

> Try to figure out by yourself, or search for references. Being able to search and find information needed is an important skill that benefits you and your career for a long time.

Import the package and rename it as np

```
import numpy as np
```

Array Creation

Task: Create a numpy array from a list of integers

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

```
array([1, 2, 3, 4, 5])
```

Task: Create a numpy array of size 6, filled with zeros

```
x = np.zeros(6)
x
```

```
array([0., 0., 0., 0., 0., 0.])
```

Task: Create a numpy array of size 6, filled with ones

```
x = np.ones(6)
x
```

```
array([1., 1., 1., 1., 1., 1.])
```

Task: Create a numpy array with values from 0 to 10

```
x = np.arange(11)
x
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

Task: Create a numpy matrix of 2*3 integers, filled with zeros

```
X = np.zeros((2, 3))
X
```

```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

Task: Create a numpy matrix of 3*2 integers, filled with ones

```
X = np.ones((3, 2))
X
```

```
array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

Task: Create a numpy matrix of 3*2 integers, filled with random integers in [0, 10]

```
X = np.random.randint(0, 10, (3, 2))
X
```

```
array([[9, 3],
       [9, 8],
       [8, 2]])
```

- Task: Create a numpy matrix of 100 * 2, filled with random float value in [0, 1]

```
X = np.random.random((100, 2))
X
```

```
array([[0.86993325, 0.50804537],
       [0.42063141, 0.6547371 ],
       [0.91080167, 0.76571313],
       [0.14421514, 0.87630482],
       [0.79689773, 0.37132623],
       [0.22367667, 0.25620628],
       [0.26828924, 0.864981  ],
       [0.44090925, 0.45742673],
       [0.79426412, 0.78505407],
       [0.31754846, 0.0039239 ],
       [0.55931054, 0.36177125],
       [0.93447121, 0.49561493],
       [0.6201546 , 0.44076041],
       [0.4212216 , 0.61158727],
       [0.30128533, 0.32032619],
       [0.70988004, 0.12327019],
       [0.98866192, 0.03074422],
       [0.21173911, 0.2732957 ],
       [0.89852586, 0.02342201],
       [0.52150973, 0.57959506],
       [0.06279927, 0.00249743],
       [0.67876758, 0.89237965],
       [0.8615494 , 0.51298976],
       [0.71315969, 0.88377509],
       [0.45459553, 0.81958311],
       [0.88257519, 0.06860784],
       [0.30367137, 0.24203548],
       [0.68389771, 0.29885327],
       [0.41343673, 0.85597743],
       [0.41551043, 0.56397876],
       [0.95730278, 0.85408894],
       [0.25905323, 0.45816369],
       [0.96149841, 0.24695246],
       [0.91739689, 0.95608249],
       [0.27241871, 0.34315565],
       [0.34936447, 0.2781561 ],
       [0.34156247, 0.12346338],
       [0.83808144, 0.89470357],
       [0.98879631, 0.10001295],
       [0.01995122, 0.82294264],
       [0.71430589, 0.39172122],
       [0.66249683, 0.49356501],
       [0.15400298, 0.42408306],
       [0.10691027, 0.66227949],
       [0.5134917 , 0.70495927],
       [0.07569228, 0.22081919],
```

```
[0.36252956, 0.23528747],
[0.87228861, 0.93253899],
[0.55017096, 0.70916113],
[0.19087477, 0.23115305],
[0.13127514, 0.41891137],
[0.46080771, 0.41029969],
[0.90845217, 0.49680636],
[0.01340749, 0.38249083],
[0.02765392, 0.15603757],
[0.69499308, 0.33277539],
[0.06670758, 0.03846533],
[0.49938889, 0.86173192],
[0.79850098, 0.66799581],
[0.38817038, 0.39085880]
```

Task: Create a numpy matrix of 2*3*4, filled with integers from 1 to 24.

```
X = np.arange(1, 25).reshape(2, 3, 4)
X
```

```
array([[[ 1,  2,  3,  4],
        [ 5,  6,  7,  8],
        [ 9, 10, 11, 12]],

       [[13, 14, 15, 16],
        [17, 18, 19, 20],
        [21, 22, 23, 24]]])
```

Array Indexing

Task: Let X be a numpy array, and X = np.array([1, 2, 3, 4, 5])

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

Task: Show the first element of X

```
X[0]
```

```
1
```

▾ Task: Show the last element of X

```
X[-1]
```

```
5
```

▾ Task: Show the first 3 elements of X

```
X[:3]
```

```
array([1, 2, 3])
```

▾ Task: Show the last 3 elements of X

```
X[2:]
```

```
array([3, 4, 5])
```

▾ Task: Show the elements of X whose index is odd (1, 3, 5, ...)

```
X[1::2]
```

```
array([2, 4])
```

▾ Task: Show the elements of X except the last one

```
X[:-1]
```

```
array([1, 2, 3, 4])
```

▾ Task: Let X be a numpy matrix with dimension 3 * 4, and filled from 0 to 11.

```
X = np.arange(12).reshape(3, 4)
X
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Task: Show the first row of X

```
X[0]
```

```
array([0, 1, 2, 3])
```

Task: Show the first column of X

```
X[:,0]
```

```
array([0, 4, 8])
```

Task: Show the first element of the first row of X

```
X[0,0]
```

```
0
```

Task: Show the last row of X

```
X[-1]
```

```
array([ 8,  9, 10, 11])
```

Task: Show the last column of X

```
X[:,-1]
```

```
array([ 3,  7, 11])
```

Task: Show the last element of the last row of X

```
X[-1,-1]
```

```
11
```

Task: Show all the rows except the last row of X

```
X[:-1]
```

```
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

Task: Show all the columns except the last column of X

```
X[:,:-1]
```

```
array([[ 0,  1,  2],
       [ 4,  5,  6],
       [ 8,  9, 10]])
```

Task: Show the first two elements of the first two rows

```
X[:2,:2]
```

```
array([[0, 1],
       [4, 5]])
```

Array Statistics

Task: Create a numpy array X, print the sum, mean, std, var, max, min of it

```
print(X.sum(),X.mean(), X.std(), X.var(), X.max(), X.min())
```

```
66 5.5 3.452052529534663 11.916666666666666 11 0
```

Task: Create a numpy matrix X as n * m, print the sum, mean, std, var, max, min of the 1st row of it.

```
r1 = X[0]
print(r1.sum(),r1.mean(), r1.std(), r1.var(), r1.max(), r1.min())
```

```
6 1.5 1.118033988749895 1.25 3 0
```

Task: Create a numpy matrix X as n * m, print the sum, mean, std, var, max, min of the 1st column of it.

```
c1 = X[:,0]
print(c1.sum(),c1.mean(), c1.std(), c1.var(), c1.max(), c1.min())
```

```
12 4.0 3.265986323710904 10.666666666666666 8 0
```

Array Linear Algebra

Task: Create two numpy matrix X with 2 * 2, and Y with 2 * 2. Practice Linear Algebra operations, such as +, -, *, .T, @, etc.

```
X = np.array([[0, 1], [2, 3]])
Y = np.array([[4, 5], [6, 7]])
plus = X + Y
minus = X - Y
times = X * Y
trans = X.T
prod = X @ Y
print(X)
print(Y)
print(plus)
print(minus)
print(times)
print(trans)
print(prod)
```

```
[[0 1]
 [2 3]]
[[4 5]
 [6 7]]
[[ 4  6]
 [ 8 10]]
[[-4 -4]
 [-4 -4]]
[[ 0  5]
 [12 21]]
[[0 2]
 [1 3]]
[[ 6  7]
 [26 31]]
```