1. Import numpy package

2. # create an array as:

    data = np.arange(12).reshape(3, 4)

    (1) Print the data
    (2) Check the number of dimensions, shape, size and data type of data

3. create numpy arrays:

    (1) Create a 3-by-4 zero array

    (2) Create a 3-by-4 one array

    (3) Create a 5-by-2 empty array

    (4) Create an array array([1, 6, 11, 16]) using arrange

    (5) create a 2-by-3 zero array by specifying the data type as float64

4. Show the name of the data type of the array

    data_one=np.array([[1, 2, 3], [4, 5, 6]])

5. Change the data type of data_one to float64

6. Change the data type of float_data = np.array([1.2, 2.3, 3.5]) to int64

7. Change the data type of str_data = np.array(['1', '2', '3']) to int64

8. Create two arrays:

    data1 = np.array([[1, 2, 3], [4, 5, 6]])

    data2 = np.array([[1, 2, 3], [4, 5, 6]])

Calculate

    (1) data1 + data2
    (2) data1 * data2
    (3) data1 - data2
    (4) data1 / data2

9. Create two arrays

    arr1 = np.array([[0], [1], [2], [3]])

    arr2 = np.array([1, 2, 3])

Calculate arr1 + arr2

10. Create two arrays:

    data1 = np.array([[1, 2, 3], [4, 5, 6]]) data2 = 10

Calculate:

(1) data1 + data2
(2) data1 * data2
(3) data1 - data2
(4) data1 / data2

11. Create an array as arr = np.arange(8)

(1) obtain the 6th element

(2) obtain the 4th to 5th elements using “:” sign

(3) obtain the 2nd to 7th elements with a step of 2 using “:” sign

12. Create an array as arr2d = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])

(1) obtain the 2nd row

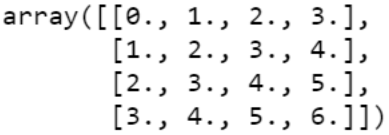
(2) obtain the element of the 1st row and the 2nd column

(3) obtain the first two rows using “:” sign

(4) obtain the array as array([[1, 2], [4, 5]]), using “:” sign

(5) obtain the array as array([4, 5]), using “:” sign

13. Create an array of



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

Obtain the two arrays using fancy indexing

(1) array([[0., 1., 2., 3.],[2., 3., 4., 5.]])
(2) array([2., 5.])

14. Create an array of names

student_name = np.array(['Tom', 'Lily', 'Jack', 'Rose'])

Create a score matrix

student_score = np.array([[79, 88, 80], [89, 90, 92], [83, 78, 85], [78, 76, 80]])

Obtain the Jack’s score using a bool array

15. Create an array

arr = np.arange(16).reshape((2, 2, 4))

(1) Perform the transpose using two methods mentioned in the class
(2) Try arr.transpose(1, 2, 0) and write down your understanding of this operation
(3) Swap the axis 1 and 0

16. Create an array x = np.array([12, 9, 13, 15])

(1) calculate the square root of x

(2) calculate the absolute value of x

(3) calculate the square of x

17. Create two arrays

arr_x = np.array([1, 5, 7])

arr_y = np.array([2, 6, 8])

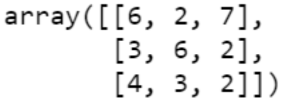
Use np.where() to obtain the array array([1, 6, 7])

18. Create an array

arr = np.arange(10)

(1) Calculate the summation of the elements in arr
(2) Calculate the average
(3) Calculate the minimal
(4) Calculate the maximal

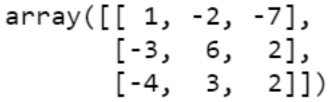
19. Create an array



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

(1) Sort each row
(2) Sort each column

20. Create an array



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

(1) Check if all the elements is greater than 0
(2) Check if at least one element is greater than 0

21. Create an array arr = np.array([12, 11, 34, 23, 12, 8, 11])

(1) find unique values

(2) each element of arr is also present in the array [11,12]

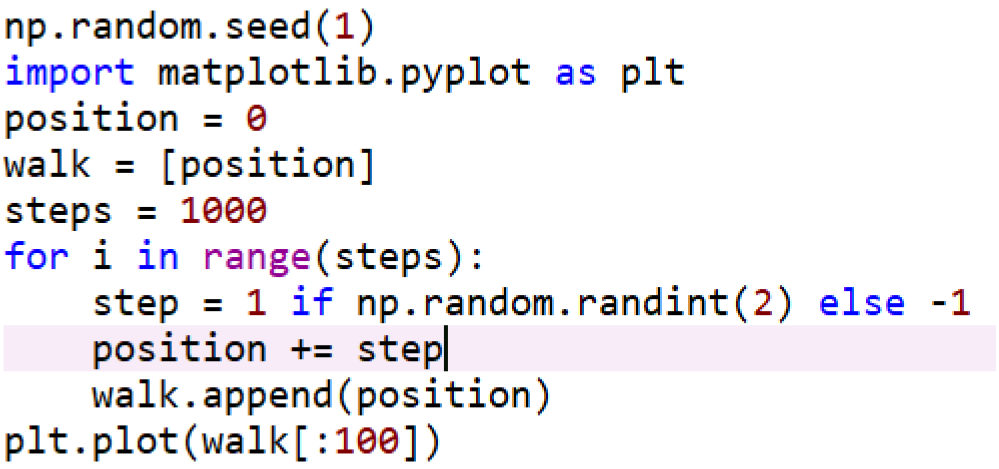
22. Create two matrix

arr_x = np.array([[1, 2, 3], [4, 5, 6]])

arr_y = np.array([[1, 2], [3, 4], [5, 6]])

Calculate the matrix multiplication using three methods.

23. Try this 1D random walk example



```
np.random.seed(1)
import matplotlib.pyplot as plt
position = 0
walk = [position]
steps = 1000
for i in range(steps):
    step = 1 if np.random.randint(2) else -1
    position += step
    walk.append(position)
plt.plot(walk[:100])
```

Now write a 3D random walk based on this example and plot the trace (consider direction of up, down, forward, backward, right, and left)

24. 实现基于牛顿法的优化算法

Python 函数接口

def newton(func,x0,K):

func: 输入要优化的函数

x0: 初始点

K: 迭代次数

测试函数:

1) x^2
2) sinc(x)
3) humps function

$$\mathrm{humps}(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04} - 6$$