Day12_Numpy_1

May 30, 2025

1 NumPy Introduction

1.1 What is NumPy?

NumPy stands for Numerical Python. It is a powerful Python library used for scientific computing. It provides support for working with large arrays, matrices, and a wide range of mathematical functions.

1.2 Why use NumPy?

- NumPy is faster and more memory-efficient than regular Python lists.
- It allows you to perform operations on entire arrays without writing loops.
- It is used in data analysis, machine learning, deep learning, simulations, image processing, and more.

1.3 Where is NumPy used?

- Data Science and Data Analysis
- Machine Learning and AI
- Scientific Research
- Signal and Image Processing
- Statistical Computing

1.4 Python List vs NumPy Array

Feature	Python List	NumPy Array
Speed	Slower	Faster
Memory	More memory usage	Less memory usage
Operations	Manual loops	Vectorized
Math Functions	Not built-in	Built-in support

[2]:
```
import numpy as np # Import numpay library (package) to use
```

[3]:
```
np.__version__
```

[3]:
```
'1.26.4'
```

[4]:
```
import sys
sys.version
```

[4]:
```
'3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.1929
64 bit (AMD64)]'
```

1.5 Create a list

[5]:
```
my_list = [0,1,2,3,4,5]
my_list
```

[5]:
```
[0, 1, 2, 3, 4, 5]
```

[6]:
```
type(my_list)
```

[6]:
```
list
```

1.6 List to Array Conversion

[7]:
```
arr = np.array(my_list)
arr
```

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

[8]:
```
type(arr) #N dimention array
```

[8]:
```
numpy.ndarray
```

1.7 Some of Numpy function

[9]:
```
np.arange(10)
```

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

[10]:
```
np.arange(5.0)
```

[10]:
```
array([0., 1., 2., 3., 4.])
```

[11]:
```
np.arange(9)
```

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

[12]:
```
np.arange(0,5)
```

[12]:
```
array([0, 1, 2, 3, 4])
```

[13]:
```
np.arange(10,20)
```

[13]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

1.7.1 Frist Argument/value must be smaller than second else return empty []

[14]:
```
np.arange(20,10)
```

[14]:
```
array([], dtype=int32)
```

[15]:
```
np.arange(-20,10)
```

[15]:
```
array([-20, -19, -18, -17, -16, -15, -14, -13, -12, -11, -10,  -9,  -8,
        -7,  -6,  -5,  -4,  -3,  -2,  -1,   0,   1,   2,   3,   4,   5,
         6,   7,   8,   9])
```

[16]:
```
np.arange(10,50,5)
```

[16]:
```
array([10, 15, 20, 25, 30, 35, 40, 45])
```

[17]:
```
np.arange(10,30,5,8)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[17], line 1
----> 1 np.arange(10,30,5,8)

TypeError: Cannot interpret '8' as a data type
```

[18]:
```
np.zeros(10, dtype=int) #parameter tunning (hyper parameter tunning)
```

[18]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

[19]:
```
np.zeros(10) #parameter tunning
```

[19]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

[20]:
```
print(np.zeros(5,dtype=int))
print(np.zeros(5,dtype=float))
print(np.zeros(5,dtype=bool))
print(np.zeros(5,dtype=complex))
```

```
[0 0 0 0 0]
[0. 0. 0. 0. 0.]
[False False False False False]
[0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j]
```

[21]:
```
np.zeros((2,2,),dtype=int) #zero with 2d
```

[21]:
```
array([[0, 0],
       [0, 0]])
```

1.7.2 Left side = row , Right side = Column

[22]:
```
np.zeros((2,10))
```

[22]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

[23]:
```
np.zeros((10,10),dtype = int)
```

[23]:
```
array([[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],
       [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],
       [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],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

[24]:
```
np.ones(6,dtype=int)
```

[24]:
```
array([1, 1, 1, 1, 1, 1])
```

[25]:
```
np.ones((3,3))
```

[25]:
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

1.7.3 There is no twos or three function only ones & zeros

[26]:
```
np.two((2,3))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[26], line 1
----> 1 np.two((2,3))

File ~\AppData\Local\anaconda3\Lib\site-packages\numpy\__init__.py:333, in␣
 ↪__getattr__(attr)
    330     "Removed in NumPy 1.25.0"
    331     raise RuntimeError("Tester was removed in NumPy 1.25.")
--> 333 raise AttributeError("module {!r} has no attribute "
    334                      "{!r}".format(__name__, attr))
```

```
AttributeError: module 'numpy' has no attribute 'two'
```

1.7.4 By default random.rand it gives value as float

[27]:
```
np.random.rand(5)
```

[27]:
```
array([0.6846088 , 0.97501809, 0.36898135, 0.13012996, 0.07861444])
```

[28]:
```
np.random.rand(3) # Every time diffrent value
```

[28]:
```
array([0.1322339 , 0.90361504, 0.83000177])
```

[29]:
```
np.random.rand(3)  # Every time diffrent value
```

[29]:
```
array([0.95971909, 0.79454639, 0.53423658])
```

[30]:
```
np.random.randint(4,6) # only print 4 Or 5 not 6
```

[30]:
```
5
```

1.7.5 randint function return int value

[31]:
```
np.random.randint(2,20,5)
```

[31]:
```
array([ 4, 13,  6, 17, 10])
```

[32]:
```
np.random.randint(1,2) # Always gives 1
```

[32]:
```
1
```