numpy Learn

basic

1.创建数组

[11]:
```python
import numpy as np
import numpy.matlib as matlib
```

将 **Python** 序列转换为 **NumPy** 数组

[20]:
```python
a=np.array([1,2,3])
b=np.array([[1,1,1],[1,1,2]])
```

使用 **numpy** 内部函数

[54]:
```python
# arange
print("np.arange(1,10)=",np.arange(1,10))
print("np.arange(1,10,dtype=float)=",np.arange(1,10,dtype=float))
# eye
print("np.eye(2)=\n",np.eye(2))
print("np.eye(2,3)=\n",np.eye(2,3))
# zeros
print("np.zeros((2,3))=\n",np.zeros((2,3)))
# ones
print("np.ones((2,3))=\n",np.ones((2,3)))

from numpy.random import default_rng
print("default_rng(42).random((2,3))=\n", default_rng(42).random((2,3)))
```

```
np.arange(1,10)= [1 2 3 4 5 6 7 8 9]
np.arange(1,10,dtype=float)= [1. 2. 3. 4. 5. 6. 7. 8. 9.]
np.eye(2)=
 [[1. 0.]
 [0. 1.]]
np.eye(2,3)=
 [[1. 0. 0.]
 [0. 1. 0.]]
np.zeros((2,3))=
 [[0. 0. 0.]
 [0. 0. 0.]]
np.ones((2,3))=
 [[1. 1. 1.]
 [1. 1. 1.]]
default_rng(42).random((2,3))=
 [[0.77395605 0.43887844 0.85859792]
 [0.69736803 0.09417735 0.97562235]]
```

[44]:
```python
#diag
print("np.diag([1,2])=\n",np.diag([1,2]))
print("np.diag([1,2])=\n",np.diag([1,2,1],1)) # 1 is the diagonal offset <0 is below, >0 is above
print("np.diag(np.diag([1,2,3]))=",np.diag(np.diag([1,2,3]))) #diag 也可以用来对角化矩阵, 比如这里对角化一个向量
```

```
np.diag([1,2])=
 [[1 0]
 [0 2]]
np.diag([1,2])=
 [[0 1 0 0]
 [0 0 2 0]
 [0 0 0 1]
 [0 0 0 0]]
np.diag(np.diag([1,2,3]))= [1 2 3]
```

np.vander(x,N) 生成一个 Vandermonde 矩阵，x 是输入的向量，N 是矩阵的列数 $V_i = x^{(n-i)}$

[45]:
```python
print("np.vander([1,2,3],3)=\n",np.vander([1,2,3],3))
```

```
np.vander([1,2,3],3)=
 [[1 1 1]
 [4 2 1]
 [9 3 1]]
```

复制、连接或更改现有数组

[68]:
```python
a = np.array([1, 2, 3, 4, 5, 6])
print(a[:2]) # [1 2]
print(a[2:]) # [3 4 5 6]

A=np.ones((2,2))
B = np.round(default_rng(42).random((2, 2)), 1)
C=np.eye(2)
S=np.block([[A,B],[C,A]])
print("np.block([[A,B],[C,A]])=\n",S)
```

```
[1 2]
[3 4 5 6]
np.block([[A,B],[C,A]])=
 [[1.  1.  0.8 0.4]
 [1.  1.  0.9 0.7]
 [1.  0.  1.  1. ]
 [0.  1.  1.  1. ]]
```

索引

[0]:
```python
a=np.array([[1,3,5,7],[2,4,6,8]])
print("a[0,0]=",a[0,0],a[0][0])

x = np.array([[1, 2], [3, 4], [5, 6]])
print("x[[0, 1, 2], [0, 1, 0]]=",x[[0, 1, 2], [0, 1, 0]])


print("a=",a)
print("a[0,:]",a[0,1:3]) # [1,3)
```

```
a[0,0]= 1 1
x[[0, 1, 2], [0, 1, 0]]= [1 4 5]
a= [[1 3 5 7]
 [2 4 6 8]]
a[0,:] [3 5]
```

[14]:
```python
np.identity(5) # 5x5 identity matrix
matlib.rand(3,3) # 3x3 random matrix
```

```
matrix([[0.84808365, 0.9577231 , 0.95592187],
        [0.59442739, 0.01748583, 0.49900773],
        [0.09914628, 0.54757987, 0.12028771]])
```

数学运算

[82]:
```python
a=np.array([1,2,3])
b=np.array([2,2,2])
np.dot(a,b)
```

```
12
```

linalg

[88]:
```python
print("QR=\n",np.linalg.qr(np.array([[1,2],[3,4]])))

print("SVD=\n",np.linalg.svd(np.array([[1,2],[3,4]])))
```

```
QR=
 (array([[-0.31622777, -0.9486833 ],
       [-0.9486833 ,  0.31622777]]), array([[-3.16227766, -4.42718872],
       [ 0.        , -0.63245553]]))
SVD=
 (array([[-0.40455358, -0.9145143 ],
       [-0.9145143 ,  0.40455358]]), array([5.4649857 , 0.36596619]),
array([[-0.57604844, -0.81741556],
       [ 0.81741556, -0.57604844]]))
```

[0]:
```python
# eig
print("eig=\n",np.linalg.eig(np.array([[1,2],[3,4]]))) # 特征值和特征向量
print("eigvals=\n",np.linalg.eigvals(np.array([[1,2],[3,4]])))

# norm
print("norm=\n",np.linalg.norm(np.array([1,2,3])))

# matrix_rank
print("matrix_rank=\n",np.linalg.matrix_rank(np.array([[1,0,0],[1,1,0],[0,1,1]])))

#inv/pinv
print("inv=\n",np.linalg.inv(np.array([[1,2],[3,4]])))
print("pinv=\n",np.linalg.pinv(np.array([[1,2],[3,4],[1,1]])))

#lstsq 线性最小二乘解
x = np.array([0, 1, 2, 3])
y = np.array([-1, 0.2, 0.9, 2.1])
A = np.vstack([x, np.ones(len(x))]).T
m, c = np.linalg.lstsq(A, y, rcond=None)[0]
print("m=",m,"c=",c)
```

```
eig=
 (array([-0.37228132,  5.37228132]), array([[-0.82456484, -0.41597356],
       [ 0.56576746, -0.90937671]]))
eigvals=
 [-0.37228132  5.37228132]
norm=
 3.7416573867739413
matrix_rank=
 3
inv=
 [[-2.   1. ]
 [ 1.5 -0.5]]
pinv=
 [[-1.5         0.5         1.        ]
 [ 1.16666667 -0.16666667 -0.66666667]]
m= 0.9999999999999997 c= -0.9499999999999999
```

Sympy Learn

[106]:
```python
import sympy as sp
```

定义变量

[108]:
```python
x,y,z,x0,y0,z0=sp.symbols('x y z x0 y0 z0')
sp.expand((x-x0)**2+(y-y0)**2+(z-z0)**2)
```

```
x**2 - 2*x*x0 + x0**2 + y**2 - 2*y*y0 + y0**2 + z**2 - 2*z*z0 + z0**2
```

dervervate of $\sin(x)e^x$

[109]:
```python
sp.diff(sp.exp(x**2),x)
```

```
2*x*exp(x**2)
```

compute

$$\int_{-\infty}^{+\infty} \sin(x^2)\mathrm{d}x$$

[114]:
```python
from sympy import oo
sp.integrate(sp.sin(x**2), (x, -oo, +oo))
```

```
sqrt(2)*sqrt(pi)/2
```

[116]:
```python
sp.limit(sp.sin(x)/x, x, 0)
```

```
1
```

[119]:
```python
sp.solve(x**3-3*x**2+3*x+4,x)[0]
```

```
1 - 5**(1/3)
```

求解微分方程

[122]:
```python
y=sp.Function('y')
eq=sp.Eq(y(x).diff(x,x)-y(x),0)
sp.dsolve(eq,y(x))
```

```
Eq(y(x), C1*exp(-x) + C2*exp(x))
```

[129]:
```python
# Define x and y as symbols
x, y = sp.symbols('x y')
sp.Matrix([[x, y,4], [3, 4,x],[x,2,3]]).det()
```

```
x**2*y - 2*x**2 - 4*x - 9*y + 24
```

Eq

[136]:
```python
print(x+1==4)
print(sp.Eq(x+1,4))
```

```
False
Eq(x + 1, 4)
```

simplify

[140]:
```python
sp.simplify((x+1)**2-(x**2+2*x+1))
```

```
0
```

逻辑

[142]:
```python
True ^ False # True | False
sp.Xor(True,False)
```

```
True
```

[152]:
```python
expr=sp.diff(sp.sin(x)+sp.cos(x)+sp.ln(x),x)
x_=np.arange(1,10)
f=sp.lambdify(x,expr,"numpy")
print(f(x_))
```

```
[ 0.69883132 -0.82544426 -0.79777917  0.35315887  1.44258646  1.40625245
  0.2397728  -1.00985828 -1.21213764]
```

[165]:
```python
m=sp.Matrix([[1,2],[2,3],[4,7]])
print(m,m.T)
m.pinv()
```

```
Matrix([[1, 2], [2, 3], [4, 7]]) Matrix([[1, 2, 4], [2, 3, 7]])
```

```
Matrix([
[-5/3,  8/3, -2/3],
[   1, -3/2,  1/2]])
```

[181]:
```python
m.rref()[0]
```

```
Matrix([
[1, 0, 0],
[0, 1, 0],
[0, 0, 1]])
```

[189]:
```python
m=sp.diag(sp.Matrix([[1,1,2],[3,6,8],[3,7,8]]))
# display(sp.Matrix(m,sp.eye(3)))
m = sp.Matrix.hstack(m, sp.eye(3))
display(m.rref()[0])
```

```
Matrix([
[1, 0, 0,    4, -3,    2],
[0, 1, 0,    0, -1,    1],
[0, 0, 1, -3/2,  2, -3/2]])
```

[0]:
```python
m=sp.diag(sp.Matrix([[1,1,2],[3,6,8],[2,2,4]]))
display(m.nullspace()[0]) # nullspace
display(m.columnspace()[0]) # column space
```

```
Matrix([
[-4/3],
[-2/3],
[   1]])
```

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

[191]:
```python
P, D = m.diagonalize()
display(P)
display(D)
```

```
Matrix([
[-4,                1/2,                1/2],
[-2, 1/4 - sqrt(77)/4, 1/4 + sqrt(77)/4],
[ 3,                  1,                  1]])
```

```
Matrix([
[0,                  0,                  0],
[0, 11/2 - sqrt(77)/2,                  0],
[0,                  0, sqrt(77)/2 + 11/2]])
```