ch_12_assignment

April 7, 2023

Ch_12_assignment

[ ]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

1 Vectors, Matrices, and Multimensional Arrays

[ ]:
```
import numpy as np
```

1.1 The Numpy Array Object

1.1.1 homogeneous data

[ ]:
```
data = np.array([[1, 2], [3, 4], [5, 6]])
type(data)
data
```

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

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

[ ]:
```
data.ndim

data.shape

data.size

data.dtype

data.nbytes
```

[ ]:
```
2
```

[ ]:
```
(3, 2)
```

[ ]: 6

[ ]: dtype('int64')

[ ]: 48

1.2 Data Types

[ ]:
```
np.array([1, 2, 3], dtype = np.int8)

np.array([1, 2, 3], dtype = np.float64)

np.array([1, 2, 3], dtype = np.cdouble)
```

[ ]: array([1, 2, 3], dtype=int8)

[ ]: array([1., 2., 3.])

[ ]: array([1.+0.j, 2.+0.j, 3.+0.j])

[ ]:
```
data = np.array([1, 2, 3], dtype = np.float64)
data
data.dtype

data = np.array([1, 2, 3], dtype = np.int64)
data
data.dtype
```

[ ]: array([1., 2., 3.])

[ ]: dtype('float64')

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

[ ]: dtype('int64')

[ ]:
```
d1 = np.array([1, 2, 3], dtype = float)
d2 = np.array([1, 2, 3], dtype = complex)

d1 + d2
(d1 + d2).dtype
```

[ ]: array([2.+0.j, 4.+0.j, 6.+0.j])

[ ]: dtype('complex128')

[ ]:
```
np.sqrt(np.array([-1, 0, 1]))
np.sqrt(np.array([-1, 0, 1], dtype = complex))
```

```
/var/folders/r1/8vnnkyjn3h3b_tnp2010w6nm0000gn/T/ipykernel_5658/433766370.py:1:
RuntimeWarning: invalid value encountered in sqrt
  np.sqrt(np.array([-1, 0, 1]))
```

[ ]:
```
array([nan,  0.,  1.])
```

[ ]:
```
array([0.+1.j, 0.+0.j, 1.+0.j])
```

1.3 Real and Imaginary

[ ]:
```
data = np.array([1, 2, 3], dtype = complex)
data

data.real
data.imag
```

[ ]:
```
array([1.+0.j, 2.+0.j, 3.+0.j])
```

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

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

1.4 Arrays Created from Lists

[ ]:
```
data = np.array([1, 2, 3, 4])

data

data.ndim

data.shape
```

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

[ ]:
```
1
```

[ ]:
```
(4,)
```

[ ]:
```
data = np.array([[1, 2], [3, 4]])

data

data.ndim
```

```
data.shape
```

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

```
2
```

```
(2, 2)
```

1.5 Arrays Filled with Constant Values

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

np.ones(4)
```

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

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

```
data = np.ones(4)
data.dtype

data = np.ones(4, dtype = int)
data.dtype
```

```
dtype('float64')
```

```
dtype('int64')
```

```
x1 = 5.4 * np.ones(10)
x1
x2 = np.full(10, 5.4)
x2
```

```
array([5.4, 5.4, 5.4, 5.4, 5.4, 5.4, 5.4, 5.4, 5.4, 5.4])
```

```
array([5.4, 5.4, 5.4, 5.4, 5.4, 5.4, 5.4, 5.4, 5.4, 5.4])
```

```
x1 = np.empty(5)
x1.fill(3.0)
x1

x2 = np.full(5, 3.0)
x2
```

```
array([3., 3., 3., 3., 3.])
```

[ ]:
```
array([3., 3., 3., 3., 3.])
```

1.6 Arrays Filled with Incremental Sequences

[ ]:
```
np.arange(0.0, 10, 1)
np.linspace(0, 10, 11)
np.logspace(0, 2, 5)
```

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

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

[ ]:
```
array([  1.        ,   3.16227766,  10.        ,  31.6227766 ,
       100.        ])
```

1.7 Meshgrid Arrays

[ ]:
```
x = np.array([-1, 0, 1])
y = np.array([-2, 0, 2])
X, Y = np.meshgrid(x, y)
Z = (X + Y) ** 2
X

Y

Z
```

[ ]:
```
array([[-1,  0,  1],
       [-1,  0,  1],
       [-1,  0,  1]])
```

[ ]:
```
array([[-2, -2, -2],
       [ 0,  0,  0],
       [ 2,  2,  2]])
```

[ ]:
```
array([[9, 4, 1],
       [1, 0, 1],
       [1, 4, 9]])
```

1.8 Creating Uninitialized Arrays

[ ]:
```
np.empty(3, dtype = np.single)
```

[ ]:
```
array([0.0000000e+00, 2.0000000e+00, 1.6914845e-26], dtype=float32)
```

1.9 Creaing Arrays with Properties of Other Arrays

[ ]:
```
def f(x):
    y = np.ones_like(x)
    return y

x = np.array([-1, 0, 1])
f(x)
```

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

1.10 Creating Matrix Arrays

[ ]:
```
np.identity(4)

np.eye(3, k = 1)

np.eye(3, k = -1)

np.diag(np.arange(0, 20, 5))
```

[ ]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

[ ]:
```
array([[0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 0.]])
```

[ ]:
```
array([[0., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.]])
```

[ ]:
```
array([[ 0,  0,  0,  0],
       [ 0,  5,  0,  0],
       [ 0,  0, 10,  0],
       [ 0,  0,  0, 15]])
```

1.11 Indexing and Slicing

1.11.1 One-Dimensional Arrays

[ ]:
```
a = np.arange(0, 11)
a

a[0]
```

```
a[-1]

a[4]

a[1:-1]

a[1: -1: 2]

a[:5]

a[-5:]

a[::-2]
```

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

[ ]:
```
0
```

[ ]:
```
10
```

[ ]:
```
4
```

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

[ ]:
```
array([1, 3, 5, 7, 9])
```

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

[ ]:
```
array([ 6,  7,  8,  9, 10])
```

[ ]:
```
array([10,  8,  6,  4,  2,  0])
```

1.12 Multidimensional Arrays

[ ]:
```
f = lambda m, n: n + 10 * m
A = np.fromfunction(f, (6, 6), dtype = int)
A
```

[ ]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [10, 11, 12, 13, 14, 15],
       [20, 21, 22, 23, 24, 25],
       [30, 31, 32, 33, 34, 35],
       [40, 41, 42, 43, 44, 45],
       [50, 51, 52, 53, 54, 55]])
```

```
A[:, 1]
A[1, :]
A[:3, :3]
A[3:, :3]
A[::2, ::2]
A[1::2, 1::3]
```

```
array([ 1, 11, 21, 31, 41, 51])
```

```
array([10, 11, 12, 13, 14, 15])
```

```
array([[ 0,  1,  2],
       [10, 11, 12],
       [20, 21, 22]])
```

```
array([[30, 31, 32],
       [40, 41, 42],
       [50, 51, 52]])
```

```
array([[ 0,  2,  4],
       [20, 22, 24],
       [40, 42, 44]])
```

```
array([[11, 14],
       [31, 34],
       [51, 54]])
```

1.13 Views

```
A
```

```
array([[ 0,  1,  2,  3,  4,  5],
       [10, 11, 12, 13, 14, 15],
       [20, 21, 22, 23, 24, 25],
       [30, 31, 32, 33, 34, 35],
       [40, 41, 42, 43, 44, 45],
       [50, 51, 52, 53, 54, 55]])
```

```
B = A[1:5, 1:5]
B

B[:, :] = 0
A
```

```
array([[11, 12, 13, 14],
       [21, 22, 23, 24],
       [31, 32, 33, 34],
```

```
       [41, 42, 43, 44]])
```

[ ]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [10,  0,  0,  0,  0, 15],
       [20,  0,  0,  0,  0, 25],
       [30,  0,  0,  0,  0, 35],
       [40,  0,  0,  0,  0, 45],
       [50, 51, 52, 53, 54, 55]])
```

[ ]:
```
C = B[1:3, 1:3].copy()
C

C[:, :] = 1
C

B
```

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

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

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

1.14 Reshaping and Resizing

[ ]:
```
data = np.array([[1, 2], [3, 4]])

np.reshape(data, (1, 4))

data.reshape(4)
```

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

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

[ ]:
```
data = np.array([[1, 2], [3, 4]])
data

data.flatten()
data.flatten().shape
```

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

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

[ ]:
```
(4,)
```

[ ]:
```
data =np.arange(0, 5)
column = data[:, np.newaxis]
column

row = data[np.newaxis, :]
row
```

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

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

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

np.vstack((data, data, data))
np.hstack((data, data, data))
```

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

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

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

1.15 Arithmetic Operations

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

x + y

x - y

x * y
```

```
y / x
```

```
[ ]: array([[ 6,  8],
            [10, 12]])
```

```
[ ]: array([[-4, -4],
            [-4, -4]])
```

```
[ ]: array([[ 5, 12],
            [21, 32]])
```

```
[ ]: array([[5.        , 3.        ],
            [2.33333333, 2.        ]])
```

```
[ ]: x * 2

     2 ** x

     y / 2

     (y / 2).dtype
```

```
[ ]: array([[2, 4],
            [6, 8]])
```

```
[ ]: array([[ 2,  4],
            [ 8, 16]])
```

```
[ ]: array([[2.5, 3. ],
            [3.5, 4. ]])
```

```
[ ]: dtype('float64')
```

```
[ ]: z = np.array([[2, 4]])
     z.shape

     zz = np.concatenate([z, z], axis = 0)
     zz
```

```
[ ]: (1, 2)
```

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

```
[ ]: x / z

     x / zz
```

[ ]:
```
array([[0.5, 0.5],
       [1.5, 1. ]])
```

[ ]:
```
array([[0.5, 0.5],
       [1.5, 1. ]])
```

[ ]:
```
z = np.array([[2], [4]])
z.shape
x / z
```

[ ]:
```
(2, 1)
```

[ ]:
```
array([[0.5 , 1.  ],
       [0.75, 1.  ]])
```

Reference * Title: Physics Programming Lecture Note (INU) * Author: Jeongwoo Kim, Ph.D. * Availability: https://sites.google.com/view/jeongwookim

---