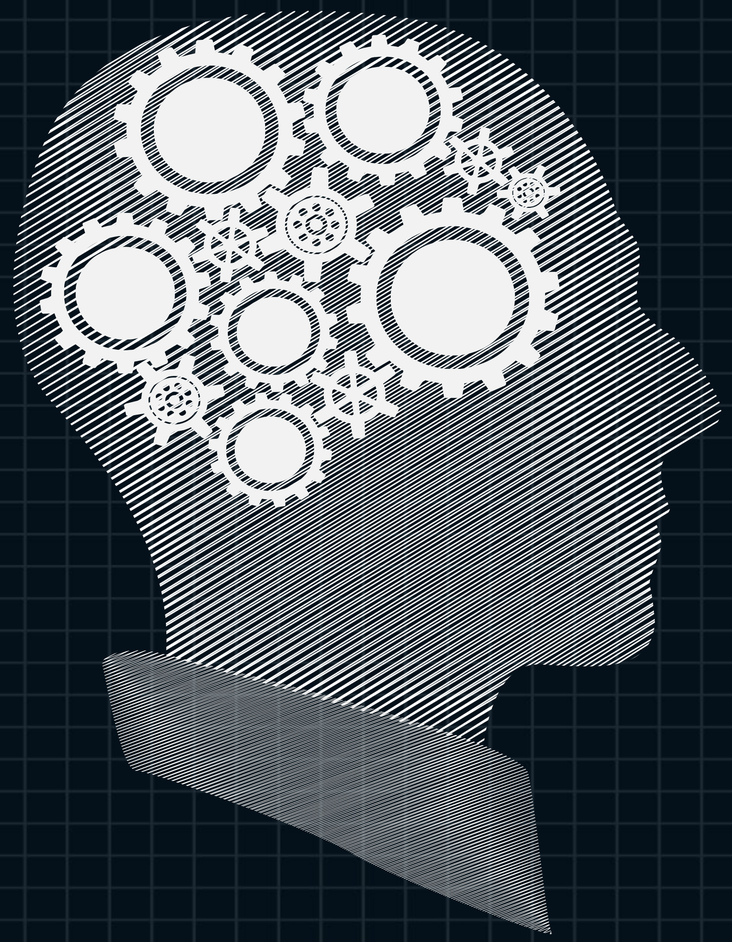


NumPy Master Class

Lecture.5
Changing ndarrays

Lecture.5 Changing ndarrays

- Changing ndarrays' shapes

Related APIs

Lecture.5 Changing ndarrays

- Changing ndarrays' shapes

np.reshape and ndarray.reshape

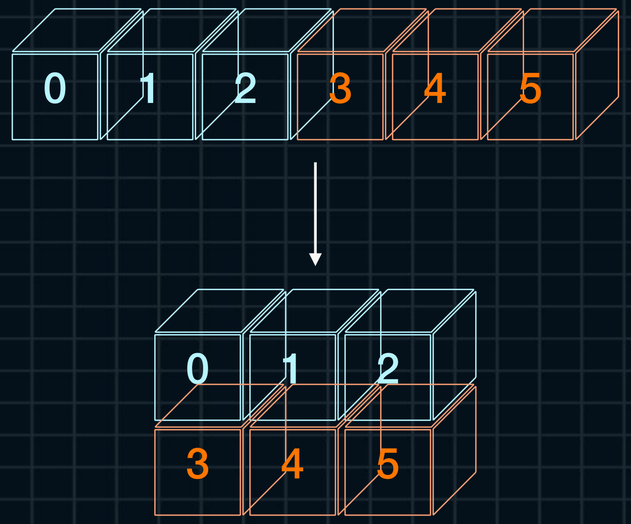
numpy.reshape(*a*, *newshape*, *order='C'*)

```
import numpy as np

a = np.arange(6)
b = np.reshape(a, (2, 3))

print("original ndarray: \n", a)
print("reshaped ndarray: \n", b)
```

```
original ndarray:
 [0 1 2 3 4 5]
reshaped ndarray:
 [[0 1 2]
 [3 4 5]]
```



Lecture.5 Changing ndarrays

- Changing ndarrays' shapes

np.reshape and ndarray.reshape

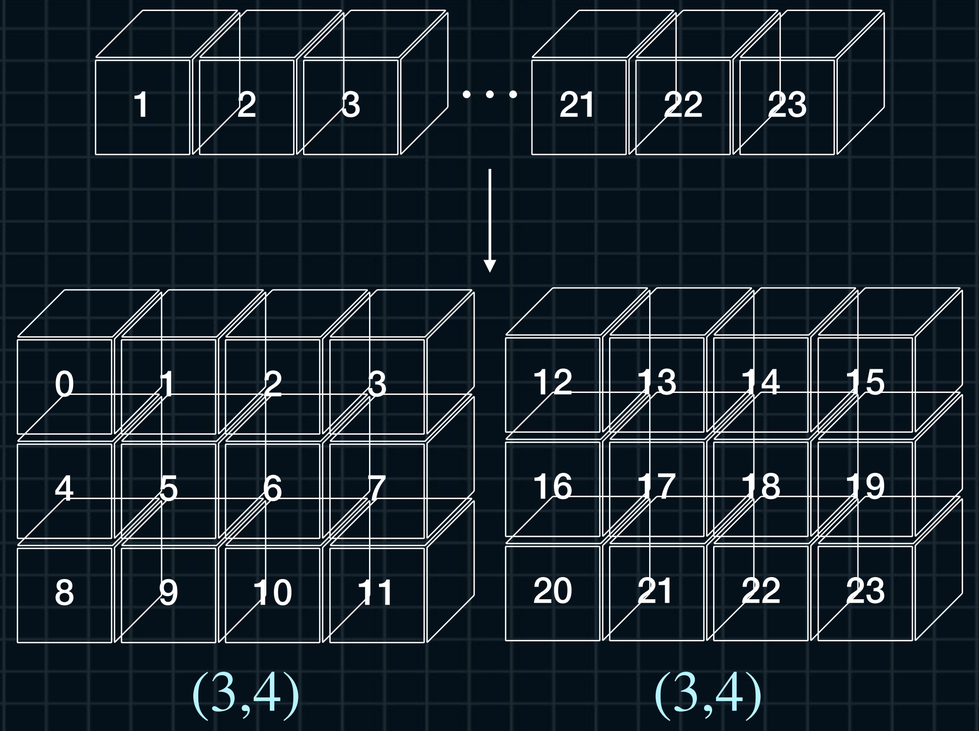
```
import numpy as np

a = np.arange(24)
b = np.reshape(a, (2, 3, 4))

print("original ndarray: \n", a)
print("reshaped ndarray: \n", b)
```

```
original ndarray:
 [ 0  1  2  3  4  5  6  7  8  9 10
   11 12 13 14 15 16 17 18 19 20
   21 22 23]
reshaped ndarray:
 [[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```



Lecture.5 Changing ndarrays

- Changing ndarrays' shapes

np.reshape and ndarray.reshape

ndarray.reshape(*shape*, *order='C'*)

```
import numpy as np

a = np.arange(6)
b = a.reshape((2, 3))

print("original ndarray: \n", a)
print("reshaped ndarray: \n", b, '\n')
```

```
original ndarray:
 [0 1 2 3 4 5]
reshaped ndarray:
 [[0 1 2]
 [3 4 5]]
```

```
b = a.reshape((2, 3, 4))

print("original ndarray: \n", a)
print("reshaped ndarray: \n", b)
```

```
original ndarray:
 [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14
   15 16 17 18 19 20 21 22 23]
reshaped ndarray:
 [[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

Lecture.5
Changing ndarrays

- Changing ndarrays' shapes

np.reshape and ndarray.reshape

```
import numpy as np

a = np.random.randint(0, 100, (100, ))

print(a.reshape((20, 5)).mean(axis=0).max())
print(np.max(np.mean(np.reshape(a, (20, 5)), axis=0)))
```

```
48.35
48.35
```

Lecture.5
Changing ndarrays

- Changing ndarrays' shapes

-1 in np.reshape

```
import numpy as np

a = np.arange(12)

b = a.reshape((2, -1))
c = a.reshape((3, -1))
d = a.reshape((4, -1))
e = a.reshape((6, -1))

print(b.shape, c.shape,     (2, 6) (3, 4) (4, 3) (6, 2)
      d.shape, e.shape)
```

Lecture.5
Changing ndarrays

- Changing ndarrays' shapes

-1 in np.reshape

```
import numpy as np

a = np.arange(12)

b = a.reshape((-1, 2))
c = a.reshape((-1, 3))
d = a.reshape((-1, 4))
e = a.reshape((-1, 6))

print(b.shape, c.shape,    (6, 2) (4, 3) (3, 4) (2, 6)
      d.shape, e.shape)
```

Lecture.5
Changing ndarrays

- Changing ndarrays' shapes

-1 in np.reshape

```
import numpy as np

a = np.arange(24)

b = a.reshape((2, 3, -1))
c = a.reshape((2, -1, 4))
d = a.reshape((-1, 3, 4))

print(b.shape, c.shape, d.shape)
```

```
(2, 3, 4) (2, 3, 4) (2, 3, 4)
```

Lecture.5
Changing ndarrays

- Changing ndarrays' shapes

-1 in np.reshape

```
import numpy as np

a = np.random.randint(0, 10, size=(2, 2))

print(a)                                                    [[9 6]
                                                            [7 1]]

row_vector = a.reshape(1, -1)
col_vector = a.reshape(-1, 1)

print(row_vector.shape, col_vector.shape)                   (1, 4) (4, 1)
```

Lecture.5
Changing ndarrays

- Changing ndarrays' shapes

np.resize and ndarray.resize

numpy.resize(*a*, *new_shape*)

```
import numpy as np

a = np.arange(6)

b = np.resize(a, (2, 3))
print("original ndarray: \n", a)
print("resized ndarray: \n", b)
```

```
original ndarray:
 [0 1 2 3 4 5]
resized ndarray:
 [[0 1 2]
 [3 4 5]]
```

Lecture.5 Changing ndarrays

- Changing ndarrays' shapes

np.resize and ndarray.resize

If the new array is larger than the original array, then the new array is filled with repeated copies of *a*.
Note that this behavior is different from a.resize(new_shape) which fills with zeros instead of repeated copies of *a*.

```
import numpy as np

a = np.arange(6)

b = np.reshape(a, (9, ))
print("original ndarray: \n", a)
print("resized ndarray: \n", b)
```

```
ValueError: cannot reshape array of size 6 into shape (9,)
```

```
b = np.resize(a, (9, ))
print("original ndarray: \n", a)
print("resized ndarray: \n", b)
```

```
original ndarray:
 [0 1 2 3 4 5]
resized ndarray:
 [0 1 2 3 4 5 0 1 2]
```

Lecture.5 Changing ndarrays

- Changing ndarrays' shapes

np.resize and ndarray.resize

```
import numpy as np

a = np.arange(6)

b = np.resize(a, (3, 4))
print("original ndarray: \n", a)
print("resized ndarray: \n", b)
```

```
original ndarray:
 [0 1 2 3 4 5]
resized ndarray:
 [[0 1 2 3]
 [4 5 0 1]
 [2 3 4 5]]
```

Lecture.5
Changing ndarrays

- Changing ndarrays' shapes

np.resize and ndarray.resize

```
import numpy as np

a = np.arange(9)

b = np.resize(a, (2, 3, 3))
print("original ndarray: \n", a)
print("resized ndarray: \n", b)
```

```
original ndarray:
 [0 1 2 3 4 5 6 7 8]
resized ndarray:
 [[[0 1 2]
  [3 4 5]
  [6 7 8]]

 [[0 1 2]
  [3 4 5]
  [6 7 8]]]
```

Lecture.5
Changing ndarrays

- Changing ndarrays' shapes

np.resize and ndarray.resize

```
import numpy as np

a = np.arange(9)


b = np.resize(a, (2, 2))
print("original ndarray: \n", a)
print("resized ndarray: \n", b)
```

```
original ndarray:
 [0 1 2 3 4 5 6 7 8]
resized ndarray:
 [[0 1]
 [2 3]]
```

Lecture.5
Changing ndarrays

- Changing ndarrays' shapes

np.resize and ndarray.resize

`ndarray.resize`(*new_shape*, *refcheck=True*)

Change shape and size of array in-place.

```
import numpy as np

a = np.arange(9)

b = a.resize((2, 2))
print("original ndarray: \n", a)
print("resized ndarray: \n", b)
```

```
original ndarray:
 [[0 1]
 [2 3]]
resized ndarray:
 None
```

Lecture.5
Changing ndarrays

- Changing ndarrays' shapes

np.resize and ndarray.resize

```
import numpy as np

a = np.arange(9)
a.resize((3, 3))

print(a)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

Lecture.5 Changing ndarrays

- Changing ndarrays' shapes

np.flatten and np.ravel

`ndarray.flatten(order='C')`

```
import numpy as np

M = np.arange(9)
N = M.reshape((3, 3))
O = N.flatten()

print(M, '\n')
print(N, '\n')
print(O, '\n')
```

```
[0 1 2 3 4 5 6 7 8]

[[0 1 2]
 [3 4 5]
 [6 7 8]]

[0 1 2 3 4 5 6 7 8]
```

Lecture.5
Changing ndarrays

- Changing ndarrays' shapes

np.flatten and np.ravel

```
import numpy as np

M = np.arange(27)
N = M.reshape((3, 3, 3))
O = N.flatten()

print(M, '\n')
print(N, '\n')
print(O, '\n')
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11
  12 13 14 15 16 17 18 19 20 21 22 23
  24 25 26]

[[[ 0  1  2]
  [ 3  4  5]
  [ 6  7  8]]

 [[ 9 10 11]
  [12 13 14]
  [15 16 17]]

 [[18 19 20]
  [21 22 23]
  [24 25 26]]]

[ 0  1  2  3  4  5  6  7  8  9 10 11
  12 13 14 15 16 17 18 19 20 21 22 23
  24 25 26]
```

Lecture.5 Changing ndarrays

- Changing ndarrays' shapes

np.flatten and np.ravel

```
import numpy as np

M = np.arange(27)
N = M.reshape((3, 3, 3))
O = N.flatten()

print(O, '\n')
```

```
 [ 0  1  2  3  4  5  6  7  8  9 10 11
  12 13 14 15 16 17 18 19 20 21 22 23
  24 25 26]
```

```
N0, N1, N2 = N[0], N[1], N[2]

print("N0: \n", N0, '\n')
print("N1: \n", N1, '\n')
print("N2: \n", N2, '\n')
```

```
N0:
 [[0 1 2]
 [3 4 5]
 [6 7 8]]

N1:
 [[ 9 10 11]
 [12 13 14]
 [15 16 17]]

N2:
 [[18 19 20]
 [21 22 23]
 [24 25 26]]
```

Lecture.5 Changing ndarrays

- Changing ndarrays' shapes

np.flatten and np.ravel

`ndarray.ravel([order])`

```
import numpy as np

M = np.arange(9)
N = M.reshape((3, 3))
O = N.ravel()

print(M, '\n')
print(N, '\n')
print(O, '\n')
```

```
[0 1 2 3 4 5 6 7 8]

[[0 1 2]
 [3 4 5]
 [6 7 8]]

[0 1 2 3 4 5 6 7 8]
```

Lecture.5
Changing ndarrays

- Memory Optimization

copy and view of ndarrays

Copy  View

Lecture.5 Changing ndarrays - Memory Optimization

copy and view of ndarrays

```
import numpy as np

a = np.arange(5)
b = a.view()

b[0] = 100

print(a)
print(b)
```

```
[100   1   2   3   4]
[100   1   2   3   4]
```

```
import numpy as np

a = np.arange(5)
b = a[0:3]

b[...] = 10

print(a)
print(b)
```

```
[10 10 10  3  4]
[10 10 10]
```

```
import numpy as np

a = np.arange(5)
b = a.copy()

b[0] = 100

print(a)
print(b)
```

```
[0 1 2 3 4]
[100   1   2   3   4]
```

Lecture.5 Changing ndarrays

- Memory Optimization

base of ndarrays

```
import numpy as np

a = np.arange(5)
b = a.copy()
c = a.view()
d = a[0:3]

print(b.base is a)    False
print(c.base is a)    True
print(d.base is a)    True
```

Lecture.5 Changing ndarrays - Memory Optimization

APIs and copy, view

```
import numpy as np

a = np.arange(4)
b = np.reshape(a, (2, 2))

b[0, 0] = 100

print(b.base is a, '\n')
print(a)
print(b)
```

```
True

[100   1   2   3]
[[100   1]
 [  2   3]]
```

```
import numpy as np

a = np.arange(5)
b = np.resize(a, (2, 2))

b[0, 0] = 100

print(b.base is a, '\n')
print(a)
print(b)
```

```
False

[0 1 2 3 4]
[[100   1]
 [  2   3]]
```

Lecture.5 Changing ndarrays - Memory Optimization

APIs and copy, view

```
import numpy as np

a = np.arange(4)
b = np.reshape(a, (2, 2)).copy()

b[0, 0] = 100

print(b.base is a, '\n')
print(a)
print(b)
```

```
False

[0 1 2 3]
[[100   1]
 [  2   3]]
```

Lecture.5 Changing ndarrays - Memory Optimization

APIs and copy, view

```python
import numpy as np
from numpy.random import randint

a = randint(0, 10, (2, 3))

b = a.ravel()
b[0] = -10

print(b.base is a, '\n')
print(a)
print(b)
```

```
True

[[-10   4   1]
 [  0   3   5]]
[-10   4   1   0   3   5]
```

```python
import numpy as np
from numpy.random import randint

a = randint(0, 10, (2, 3))

b = a.flatten()
b[0] = -10

print(b.base is a, '\n')
print(a)
print(b)
```

```
False

[[9 7 7]
 [9 1 3]]
[-10   7   7   9   1   3]
```

Lecture.5
Changing ndarrays

- Changing ndarrays' dtype

ndarray.astype

```
import numpy as np

M = np.array([1, 2, 3], np.int8)
print(M.dtype)                          int8

N = M.astype(np.uint32)
O = M.astype(np.float32)
print(N.dtype)                          uint32
print(O.dtype)                          float32
```

Lecture.5
Changing ndarrays

- Changing ndarrays' dtype

ndarray.astype

```
import numpy as np

M = np.random.uniform(low=-10, high=10, size=(3, 3))
print(M, '\n')
print(M.astype(np.int32))
```

```
[[ 8.24614266  5.05611674 -7.91162982]
 [-7.99606061 -1.21236783  9.96390134]
 [ 5.99259012 -7.27320016 -6.43903195]]

[[ 8  5 -7]
 [-7 -1  9]
 [ 5 -7 -6]]
```

Lecture.5 Changing ndarrays

- Changing ndarrays' dtype

bool dtype ndarrays

```
import numpy as np

bools = np.array([True, False])
print(f"bool: \n{bools}")


bools2ints = bools.astype(np.int)
print(f"int: \n{bools2ints}")

bools2floats = bools.astype(np.float)
print(f"float: \n{bools2floats}")
```

```
bool:
[ True False]


int:
[1 0]


float:
[1. 0.]
```

Lecture.5 Changing ndarrays

- Changing ndarrays' dtype

bool dtype ndarrays

```
import numpy as np

ints = np.array([-2, -1, 0, 1, 2])
floats = np.array([-2.5, -1.5, 0., 1.5, 2.5])

print(f"ints: \n{ints}")
print(f"floats: \n{floats}\n”)


ints2bools = ints.astype(np.bool)
print(f"ints -> bools: \n{ints2bools}")

floats2bools = ints.astype(np.bool)
print(f"floats -> bools: \n{floats2bools}")
```

```
ints:
[-2 -1  0  1  2]
floats:
[-2.5 -1.5  0.   1.5  2.5]


ints -> bools:
[ True  True False  True  True]


floats -> bools:
[ True  True False  True  True]
```

Lecture.5 Changing ndarrays

- Changing ndarrays' dtype

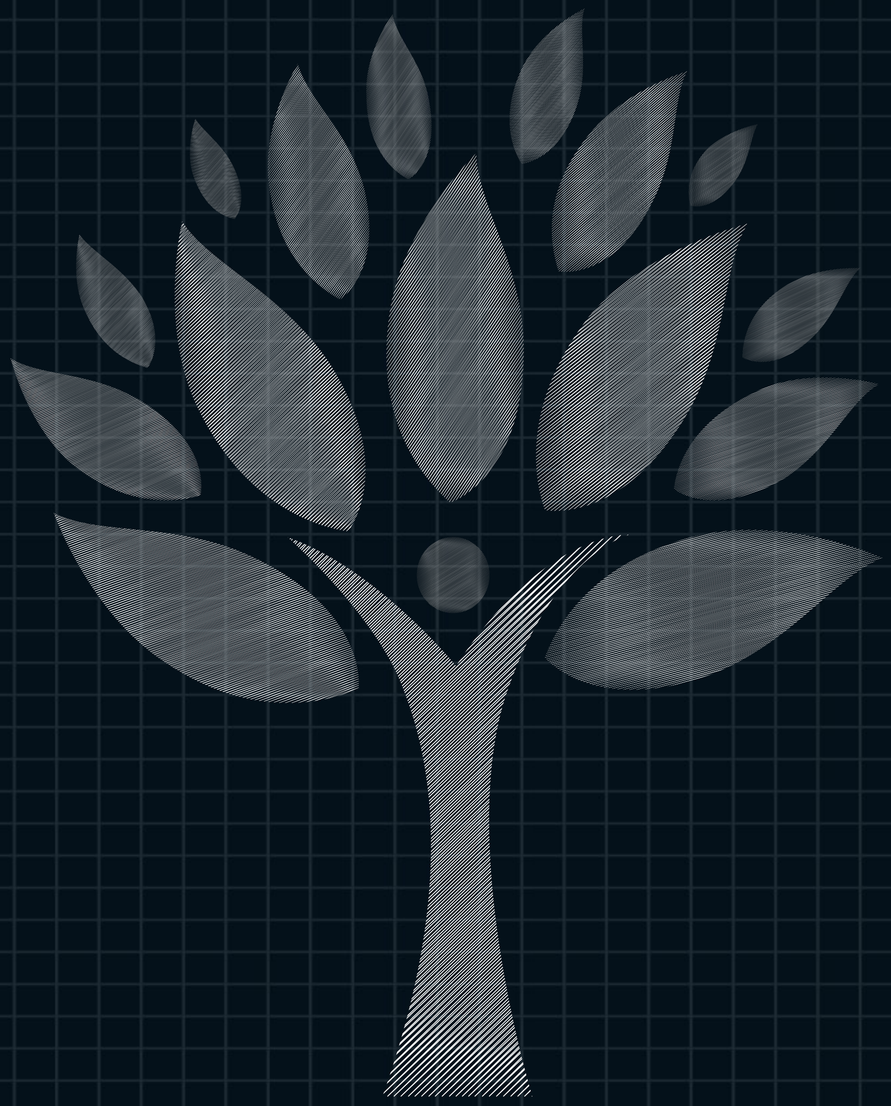
bool dtype ndarrays

```
print(-3 == True, -3 == False)
print(3.14 == True, 3.14 == False)
```

```
False False
False False
```

```
print(0. == True, 0. == False)
print(1. == True, 1. == False)
```

```
False True
True False
```



NumPy Master Class

Lecture.5
Changing ndarrays