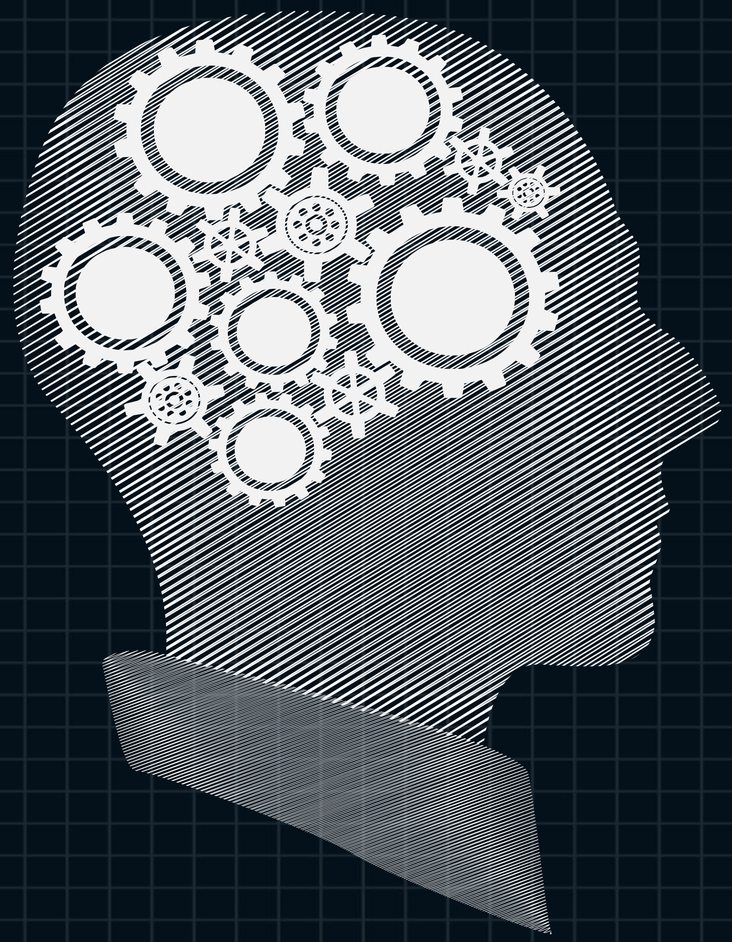


NumPy Master Class

Lecture.8
axis and keepdims Arguments

Lecture.8 axis and keepdims Arguments

- Related APIs

```
np.sum          np.var
np.cumsum       np.std
np.prod         np.amax
np.cumprod      np.argmax
np.diff         np.amin
np.mean         np.argmin
np.median       np.sort
```

Lecture.8 axis and keepdims Arguments - Vector Case

np.sum and ndarray.sum of Vector

```
import numpy as np

a = np.arange(5)
print("ndarray: ", a)

print("np.sum: ", np.sum(a))
print("ndarray.sum: ", a.sum())
```

```
ndarray:  [0 1 2 3 4]

np.sum:  10
ndarray.sum:  10
```

Lecture.8 axis and keepdims Arguments - Matrix Case

np.sum and ndarray.sum of Matrix

```
import numpy as np

a = np.arange(6).reshape((2, -1))
print("ndarray: \n", a)

print("np.sum: ", np.sum(a))
print("ndarray.sum: ", a.sum())
```

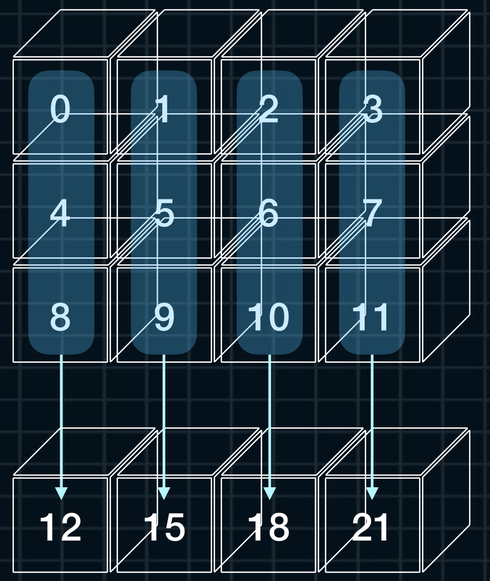
```
ndarray:
 [[0 1 2]
 [3 4 5]]

np.sum:  15
ndarray.sum:  15
```

Lecture.8 axis and keepdims Arguments - Matrix Case

axis Argument

```
import numpy as np

a = np.arange(12).reshape((3, -1))

sum_ = a.sum(axis=0)

print("ndarray: {}\n{}".format(a.shape, a))
print("ndarray.sum(axis=0): {}\n{}".format(sum_.shape, sum_))
```

```
ndarray: (3, 4)
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
ndarray.sum(axis=0): (4,)
[12 15 18 21]
```



(3,4)

np.sum(axis=0)

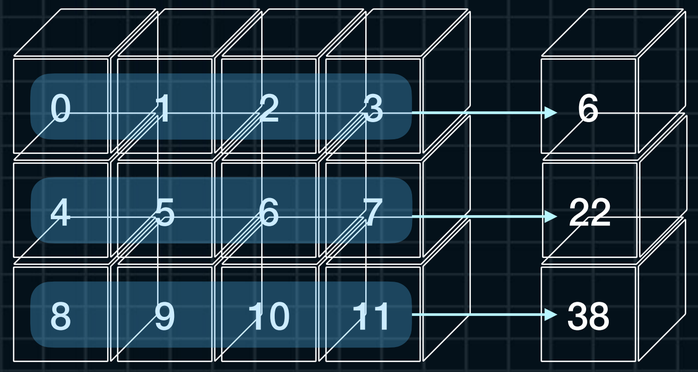
(4,)

Lecture.8 axis and keepdims Arguments - Matrix Case

axis Argument

```
import numpy as np

a = np.arange(12).reshape((3, -1))

sum_ = a.sum(axis=1)

print("ndarray: {}\n{}".format(a.shape, a))
print("ndarray.sum(axis=1): {}\n{}".format(sum_.shape, sum_))
```

```
ndarray: (3, 4)
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
ndarray.sum(axis=1): (3,)
[ 6 22 38]
```



(3,4) (3,)

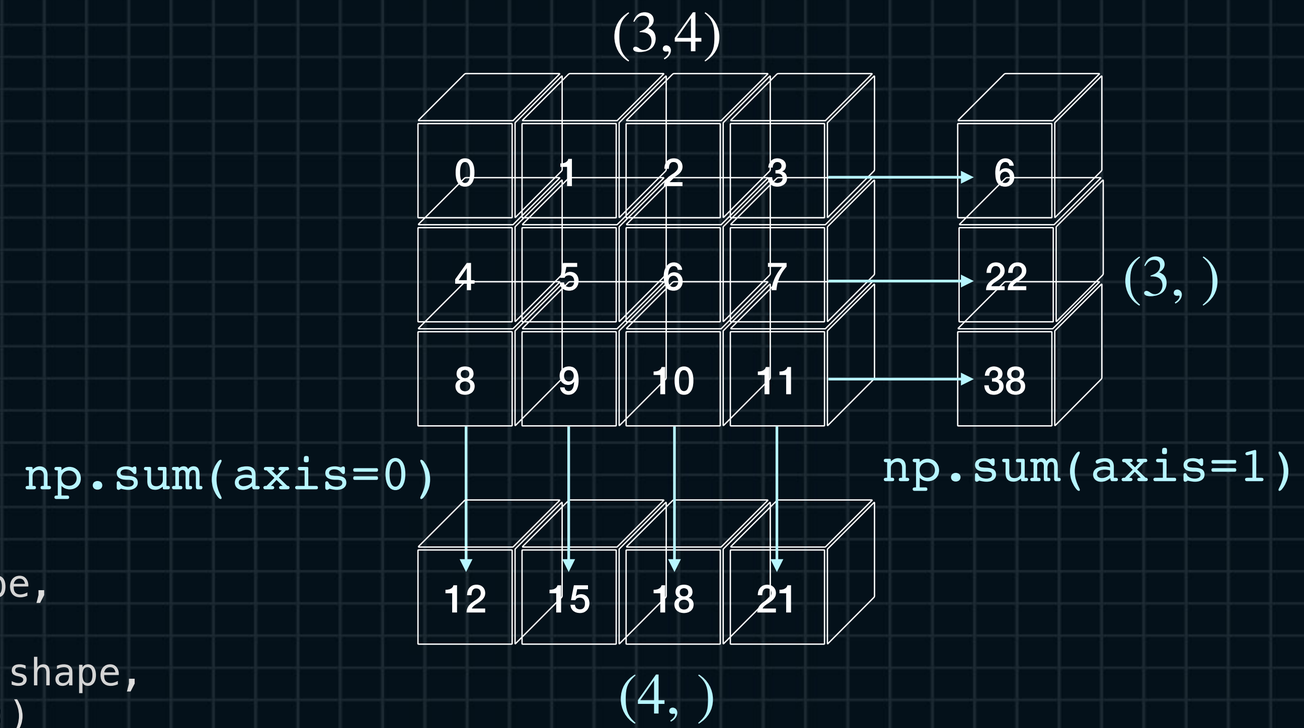
np.sum(axis=1)

Lecture.8 axis and keepdims Arguments - Matrix Case

Application

```
import numpy as np

a = np.arange(12).reshape((3, -1))

sum_class = np.sum(a, axis=0)
sum_student = np.sum(a, axis=1)


print("scores: {}\n{}".format(a.shape, a))
print("sum_class: {}\n{}".format(sum_class.shape,
                                 sum_class))
print("sum_student: {}\n{}".format(sum_student.shape,
                                   sum_student))
```

```
scores: (3, 4)
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
sum_class: (4,)
[12 15 18 21]
sum_student: (3,)
[ 6 22 38]
```

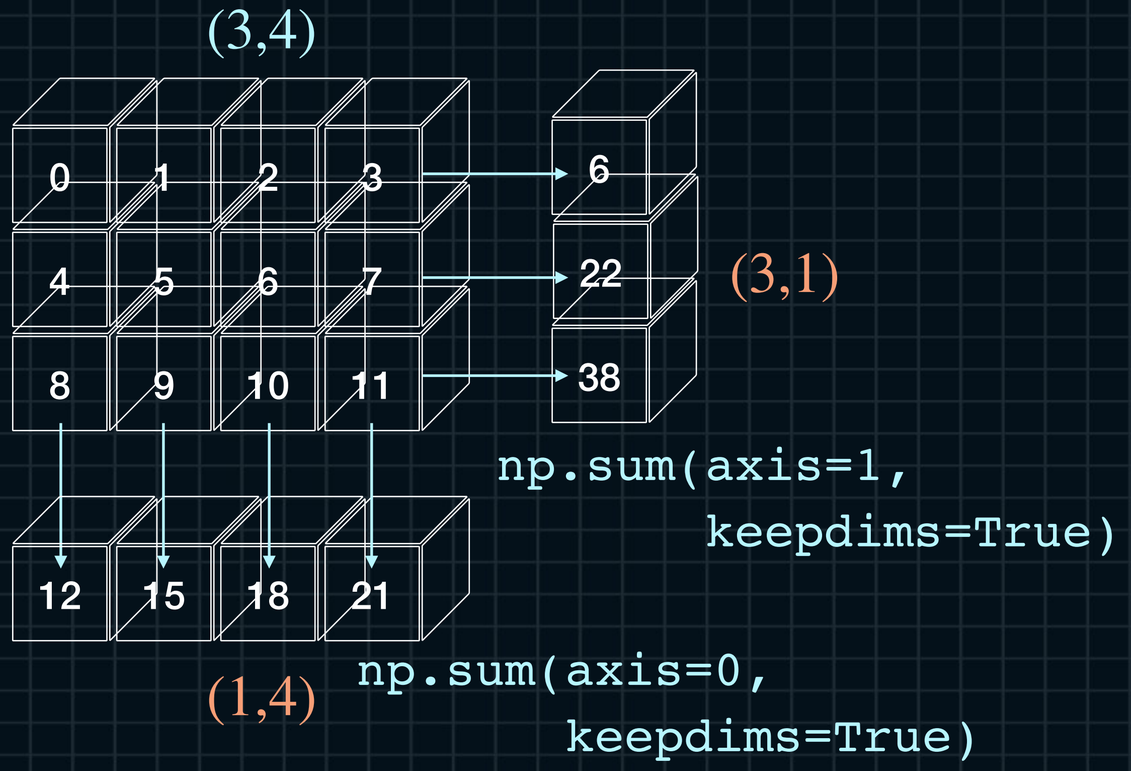


Lecture.8 axis and keepdims Arguments - Matrix Case

keepdims Argument

```
import numpy as np

a = np.arange(12).reshape((3, -1))

sum_class = np.sum(a, axis=0, keepdims=True)
sum_student = np.sum(a, axis=1, keepdims=True)

print("scores: {}\n{}".format(a.shape, a))
print("sum_class: {}\n{}".format(sum_class.shape,
                                 sum_class))
print("sum_student: {}\n{}".format(sum_student.shape,
                                   sum_student))
```

```
scores: (3, 4)
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
sum_class: (1, 4)
[[12 15 18 21]]
sum_student: (3, 1)
[[ 6]
 [22]
 [38]]
```



Lecture.8 axis and keepdims Arguments - Matrix Case

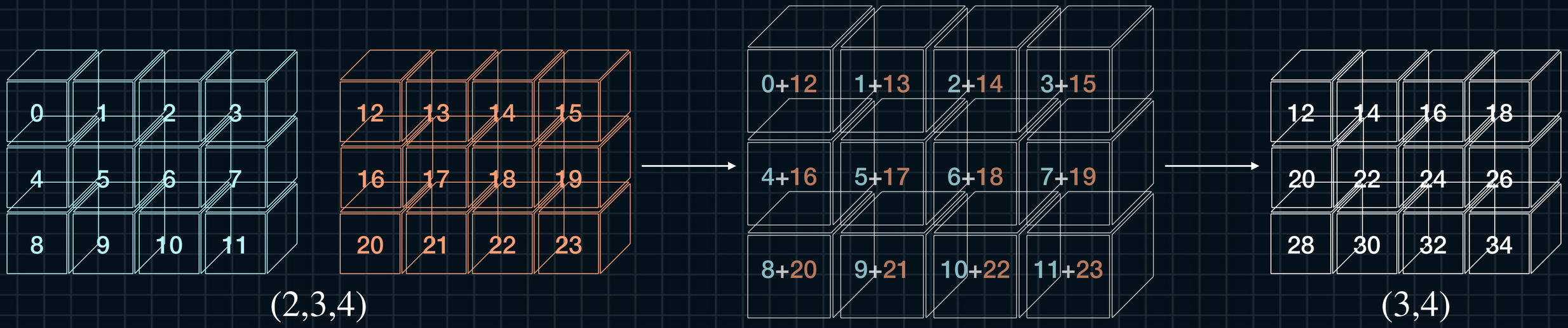
Application

```
import numpy as np

n_student, n_class = 3, 4
m_score, M_score = 0, 100
scores = np.random.randint(low=m_score,
                           high=M_score,
                           size=(n_student,n_class))


mean_class = np.mean(scores, axis=0, keepdims=True)
mean_student = np.mean(scores, axis=1, keepdims=True)

print("Shapes: ")
print(scores.shape, mean_class.shape, mean_student.shape, '\n')

print("Mean subtraction by classes\n", scores - mean_class)
print("Mean subtraction by students\n", scores - mean_student)
```

```
Shapes:
(3, 4) (1, 4) (3, 1)

Mean subtraction by classes
 [[ 18.          42.66666667 -32.33333333 -31.33333333]
 [-24.         -24.33333333 -16.33333333  46.66666667]
 [  6.         -18.33333333  48.66666667 -15.33333333]]
Mean subtraction by students
 [[  2.    57.   -35.   -24.  ]
 [-36.25  -6.25 -15.25  57.75]
 [-16.   -10.    40.   -14.  ]]
```

Lecture.8 axis and keepdims Arguments

- 3rd Order Tensor Case



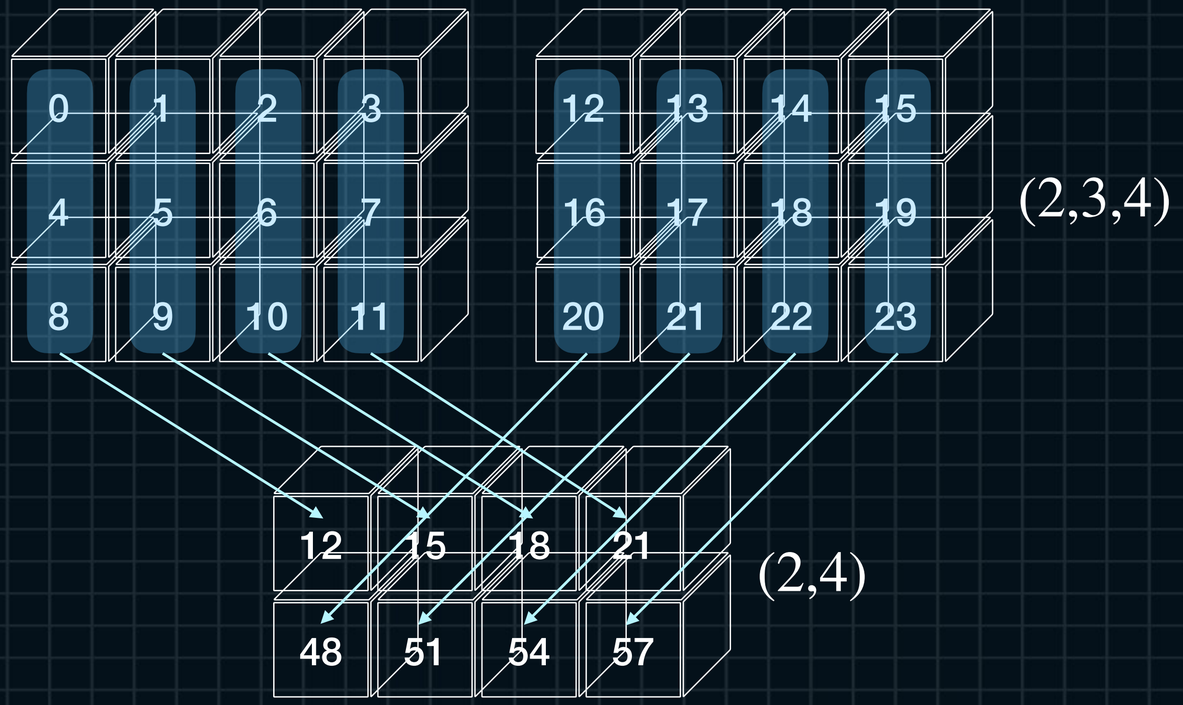
Lecture.8 axis and keepdims Arguments - 3rd Order Tensor Case

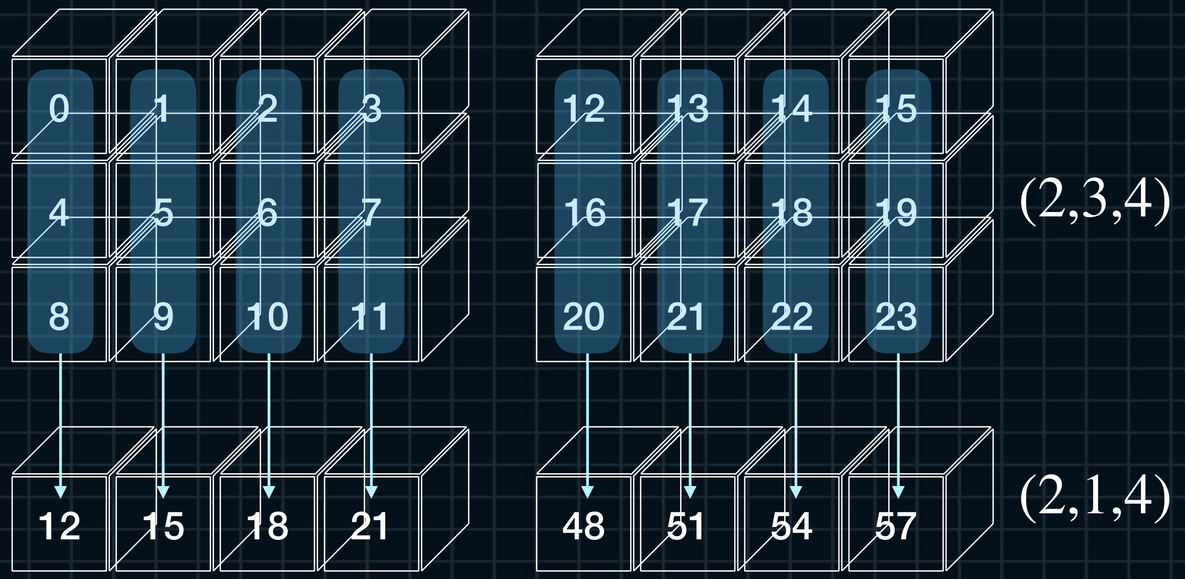
```
import numpy as np

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

sum_ = a.sum(axis=0)

print("ndarray: {}\n{}".format(a.shape, a))
print("ndarray.sum(axis=0): {}\n{}".format(sum_.shape, sum_))
```

```
ndarray: (2, 3, 4)
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
ndarray.sum(axis=0): (3, 4)
[[12 14 16 18]
 [20 22 24 26]
 [28 30 32 34]]
```

Lecture.8 axis and keepdims Arguments

- 3rd Order Tensor Case





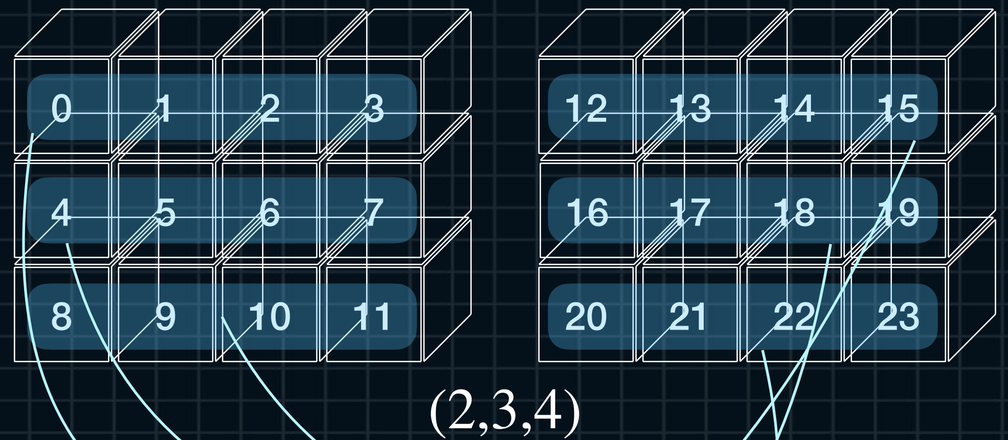
Lecture.8 axis and keepdims Arguments - 3rd Order Tensor Case

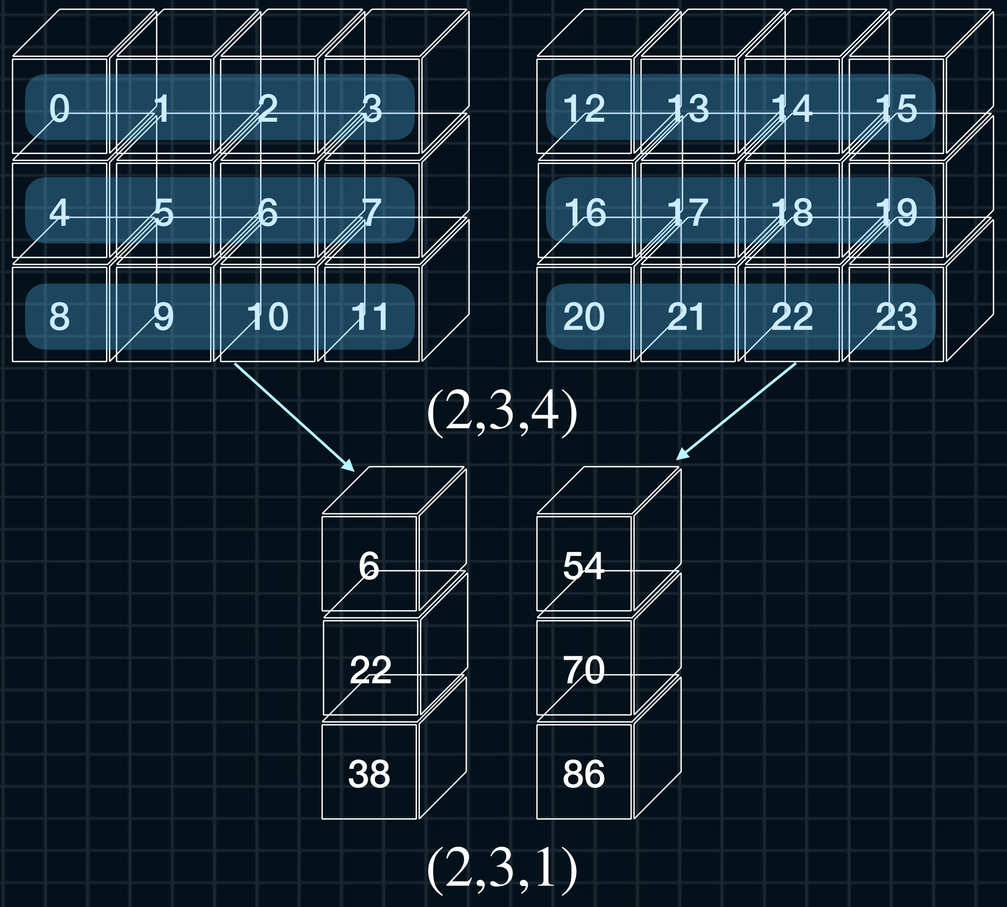
```python
import numpy as np

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

sum_ = a.sum(axis=1)
sum_k = a.sum(axis=1, keepdims=True)

print("ndarray: {}\n{}".format(a.shape, a))
print("axis=1: {}\n{}".format(sum_.shape, sum_))
print("axis=1, keepdims=True: {}\n{}"\
      .format(sum_k.shape,sum_k))
```

```
ndarray: (2, 3, 4)
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
axis=1: (2, 4)
[[12 15 18 21]
 [48 51 54 57]]
axis=1, keepdims=True: (2, 1, 4)
[[[12 15 18 21]]

 [[48 51 54 57]]]
```

Lecture.8 axis and keepdims Arguments

- 3rd Order Tensor Case





Lecture.8 axis and keepdims Arguments - 3rd Order Tensor Case

```python
import numpy as np

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

sum_ = a.sum(axis=2)
sum_k = a.sum(axis=2, keepdims=True)

print("ndarray: {}\n{}".format(a.shape, a))
print("axis=2: {}\n{}".format(sum_.shape, sum_))
print("axis=2, keepdims=True: {}\n{}"\
      .format(sum_k.shape,sum_k))
```

```
ndarray: (2, 3, 4)
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
axis=2: (2, 3)
[[ 6 22 38]
 [54 70 86]]
axis=2, keepdims=True: (2, 3, 1)
[[[ 6]
  [22]
  [38]]

 [[54]
  [70]
  [86]]]
```

Lecture.8 axis and keepdims Arguments - 3rd Order Tensor Case

Application

```
import numpy as np

n_test_time, n_student, n_class = 4, 3, 4
m_score, M_score = 0, 100
scores = np.random.randint(low=m_score,
                           high=M_score,
                           size=(n_test_time,
                                 n_student,
                                 n_class))
print("scores: \n", scores)
```

```
scores:
 [[[82 18 90 95]  [[34 51  8 12]
  [99 36  6 74]    [67 29 85 50]
  [50 46 99 77]]   [98 91  3 37]]

 [[78 59 66 91]   [[83 16 34 92]
  [62 98 16 17]    [ 4 99 32 42]
  [30 41 12 34]]   [95 36 19 58]]]
```

```
score_mean = np.mean(scores, axis=0)
print("score mean: ",
      score_mean.shape, '\n',
      score_mean)
```

```
score mean:  (3, 4)
 [[69.25 36.   49.5  72.5 ]
 [58.   65.5  34.75 45.75]
 [68.25 53.5  33.25 51.5 ]]
```

```
score_mean = np.mean(scores, axis=1)
print("score mean: ",
      score_mean.shape, '\n',
      score_mean)
```

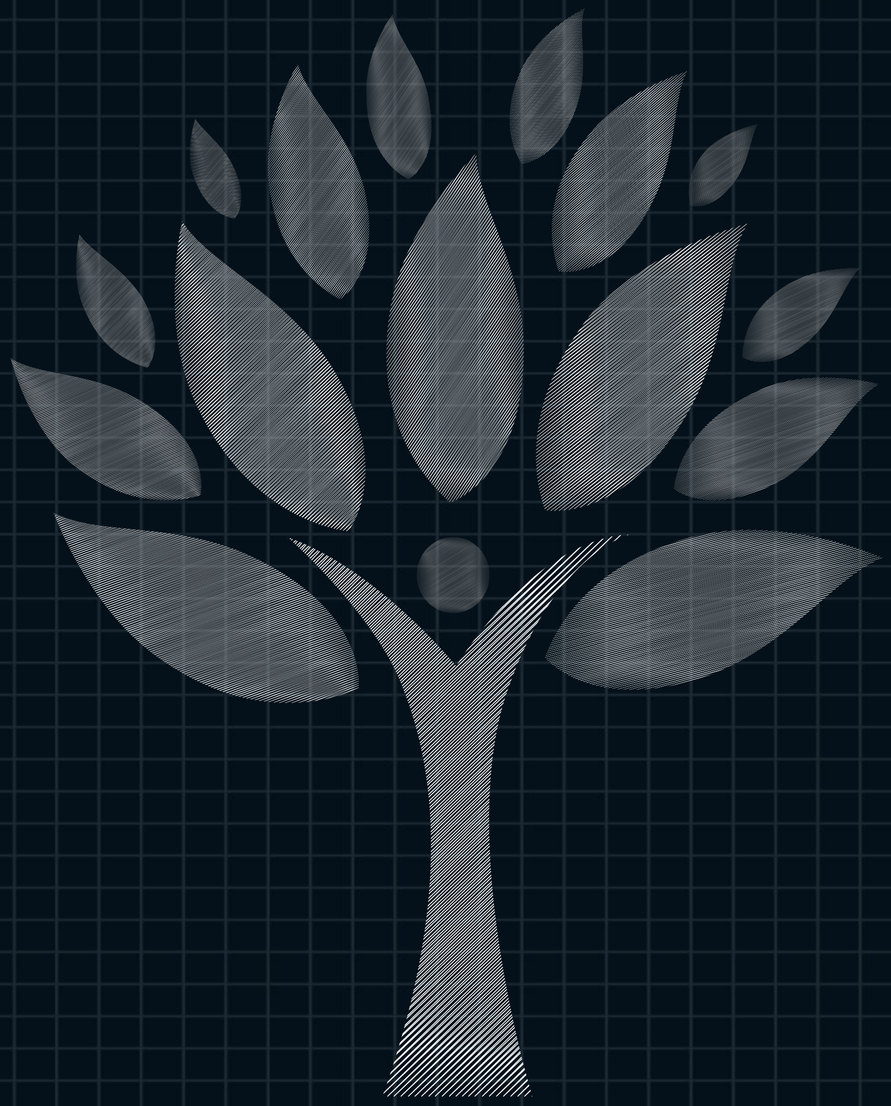
```
score mean:  (4, 4)
 [[77.         33.33333333 65.         82.        ]
 [56.66666667 66.         31.33333333 47.33333333]
 [66.33333333 57.         32.         33.        ]
 [60.66666667 50.33333333 28.33333333 64.        ]]
```

Lecture.8 axis and keepdims Arguments - 3rd Order Tensor Case

Application

```
import numpy as np

C, H, W = 3, 100, 200

# (C, H, W) case
images = np.random.randint(0, 256,
                           size=(C, H, W))
print("Shape of original image:", images.shape)

gray_image = np.mean(images, axis=0)
print("Shape of gray-scaled image:", gray_image.shape, '\n')

# (H, W, C) case
images = np.random.randint(0, 256,
                           size=(H, W, C))
print("Shape of original image:", images.shape)


gray_image = np.mean(images, axis=-1)
print("Shape of gray-scaled image:", gray_image.shape)
```

```
Shape of original image: (3, 100, 200)
Shape of gray-scaled image: (100, 200)

Shape of original image: (100, 200, 3)

Shape of gray-scaled image: (100, 200)
```



NumPy Master Class

Lecture.8
axis and keepdims Arguments