Numpy Arrays

Basic Python

Winter 2022: Dan Calderone

Python - Base Data Structures

list : x = [1, ’a’, func]

dict :

```
X = { ‘key1’:  1,
      ‘key2’:  ‘a’,
      ‘key3’:  func }
```

np.array:

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

np.array : A = np.array([[1, 2, 3], [3, 2, 1], [2, 1, 3]])

A[np.newaxis,:]   A[:,np.newaxis]

np.stack([x,x,x]) …stack along new axis

np.vstack([x,x,x]) …stack vertically

np.hstack([x,x,x]) …stack horizontally

```
np.block([[A,B]
          [C,D]])
```

…block matrix

np.eye(n)

np.ones([m,n])

np.zeros([m,n])

np.arange(start,stop,step=1)

np.arange(start,stop,num=50)

np.append(A,newarray) …adds newarray to the end

np.insert(A,index,newarray) …adds new array at index

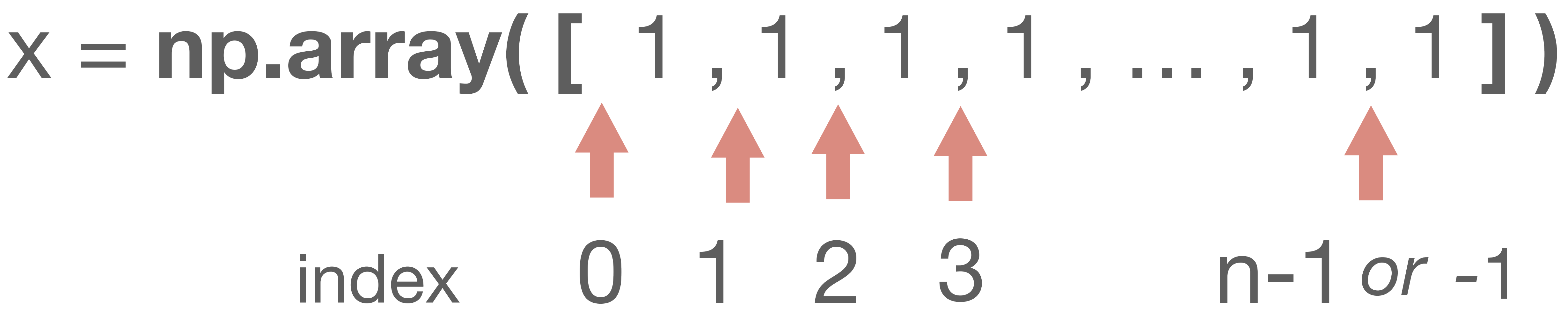
np.reshape(A,newshape) …cycle through deepest axes first

np.concatenate((A,B,C),axis=0) …must have same shape except along axis

np.flip(A,axis=None) …by default flips all axes

np.where(A,axis=None)

Python - Indexing



x = **np.array([** 1 , 1 , 1 , 1 , … , 1 , 1 **])**

index 0 1 2 3 n-1 *or* -1

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*

x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];

x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];

X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH

x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block

X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block

Python - Indexing

np.array: A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

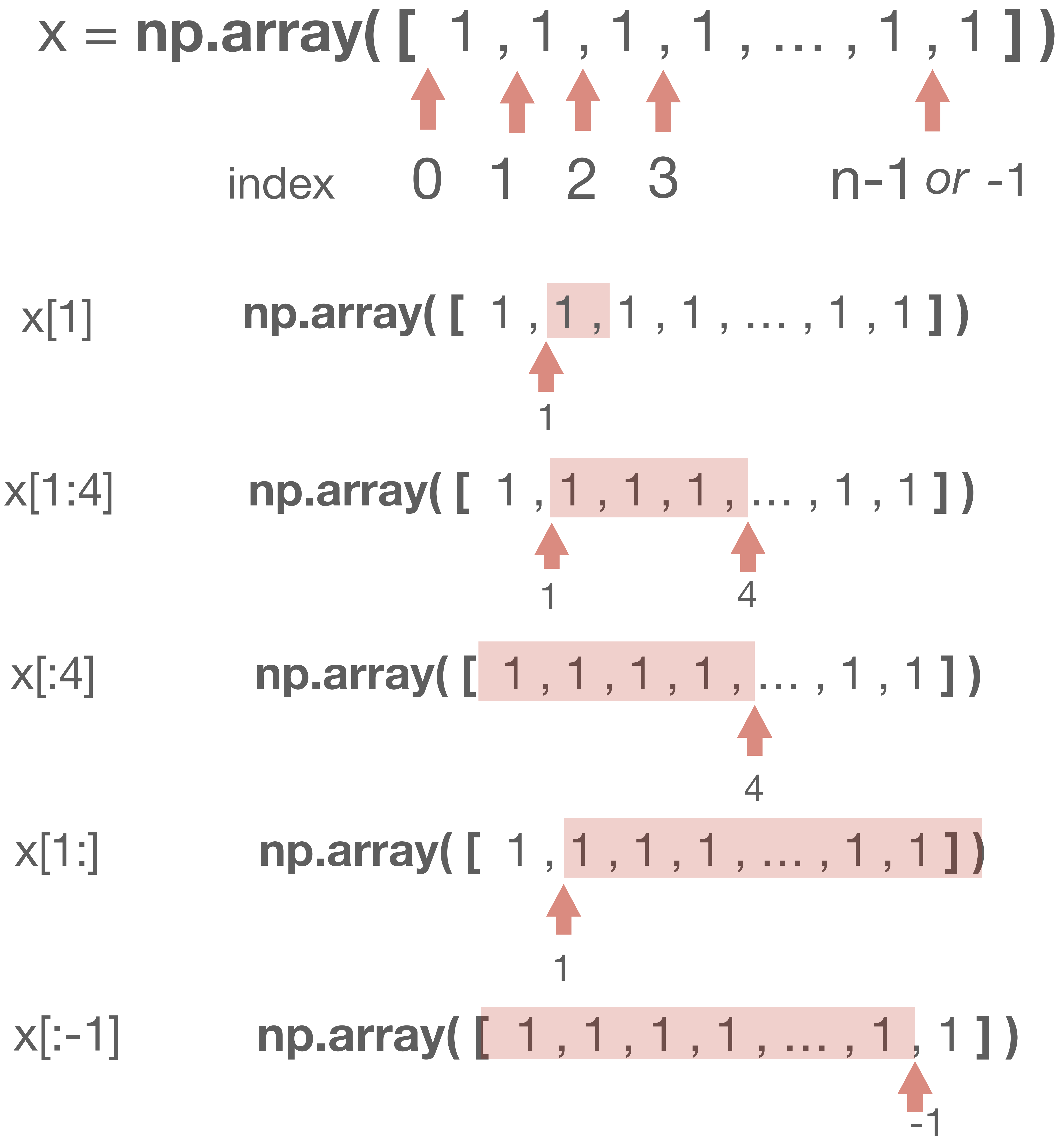
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



x = **np.array([** 1 , 1 , 1 , 1 , … , 1 , 1 **])**

index 0 1 2 3 n-1 *or* -1

x[1] **np.array([** 1 , 1 , 1 , 1 , … , 1 , 1 **])** 1

x[1:4] **np.array([** 1 , 1 , 1 , 1 , … , 1 , 1 **])** 1 4

x[:4] **np.array([** 1 , 1 , 1 , 1 , … , 1 , 1 **])** 4

x[1:] **np.array([** 1 , 1 , 1 , 1 , … , 1 , 1 **])** 1

x[:-1] **np.array([** 1 , 1 , 1 , 1 , … , 1 , 1 **])** -1

Python - Indexing

np.array: A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

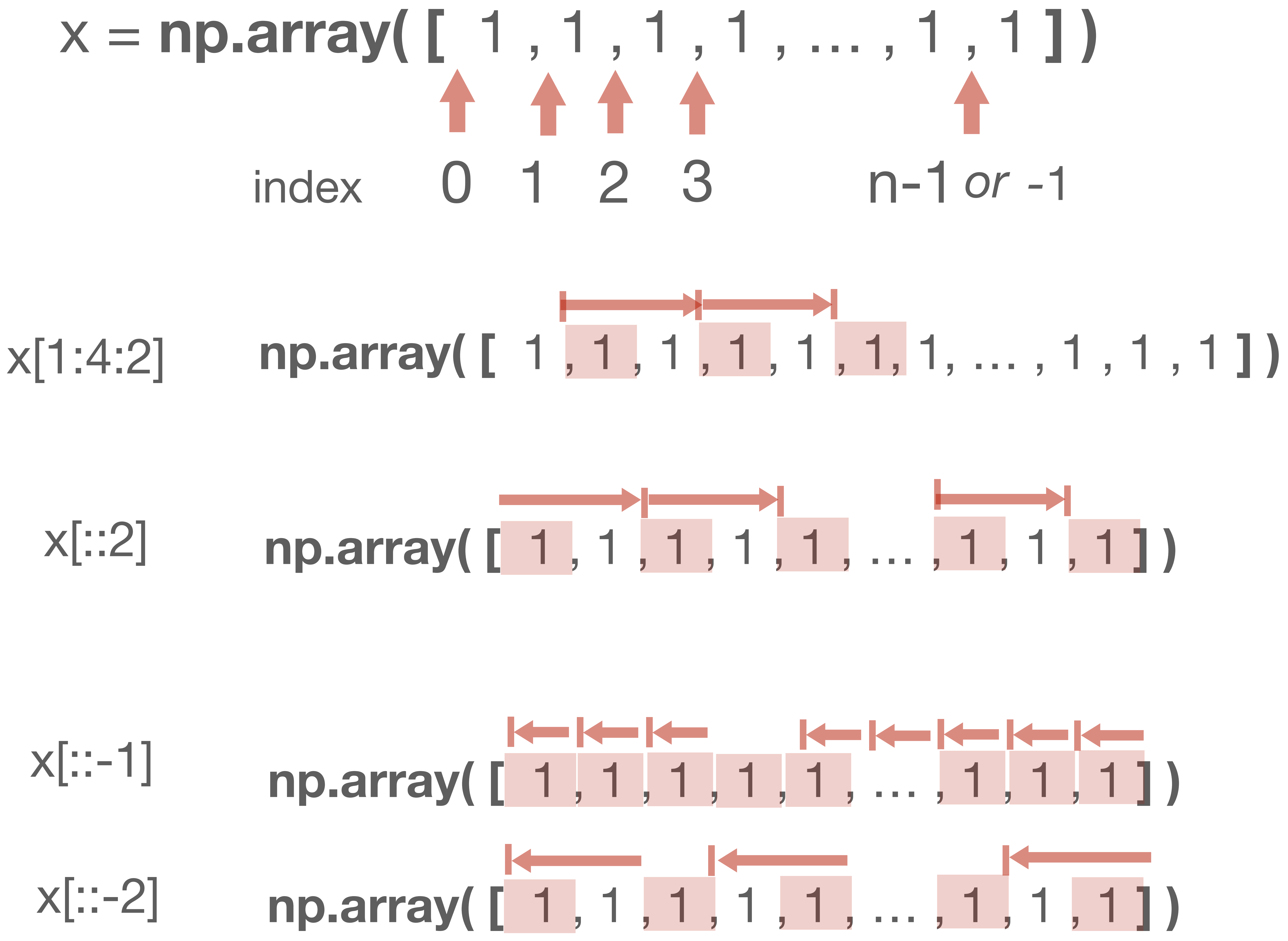
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



x = **np.array([** 1 , 1 , 1 , 1 , … , 1 , 1 **])**

index 0 1 2 3 n-1 *or* -1

x[1:4:2] **np.array([** 1 , 1 , 1 , 1 , 1 , 1, 1, … , 1 , 1 , 1 **])**

x[::2] **np.array([** 1 , 1 , 1 , 1 , 1 , … , 1 , 1 , 1 **])**

x[::-1] **np.array([** 1 , 1 , 1 , 1 , 1 , … , 1 , 1 , 1 **])**

x[::-2] **np.array([** 1 , 1 , 1 , 1 , 1 , … , 1 , 1 , 1 **])**

Python - Indexing

np.array:

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

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

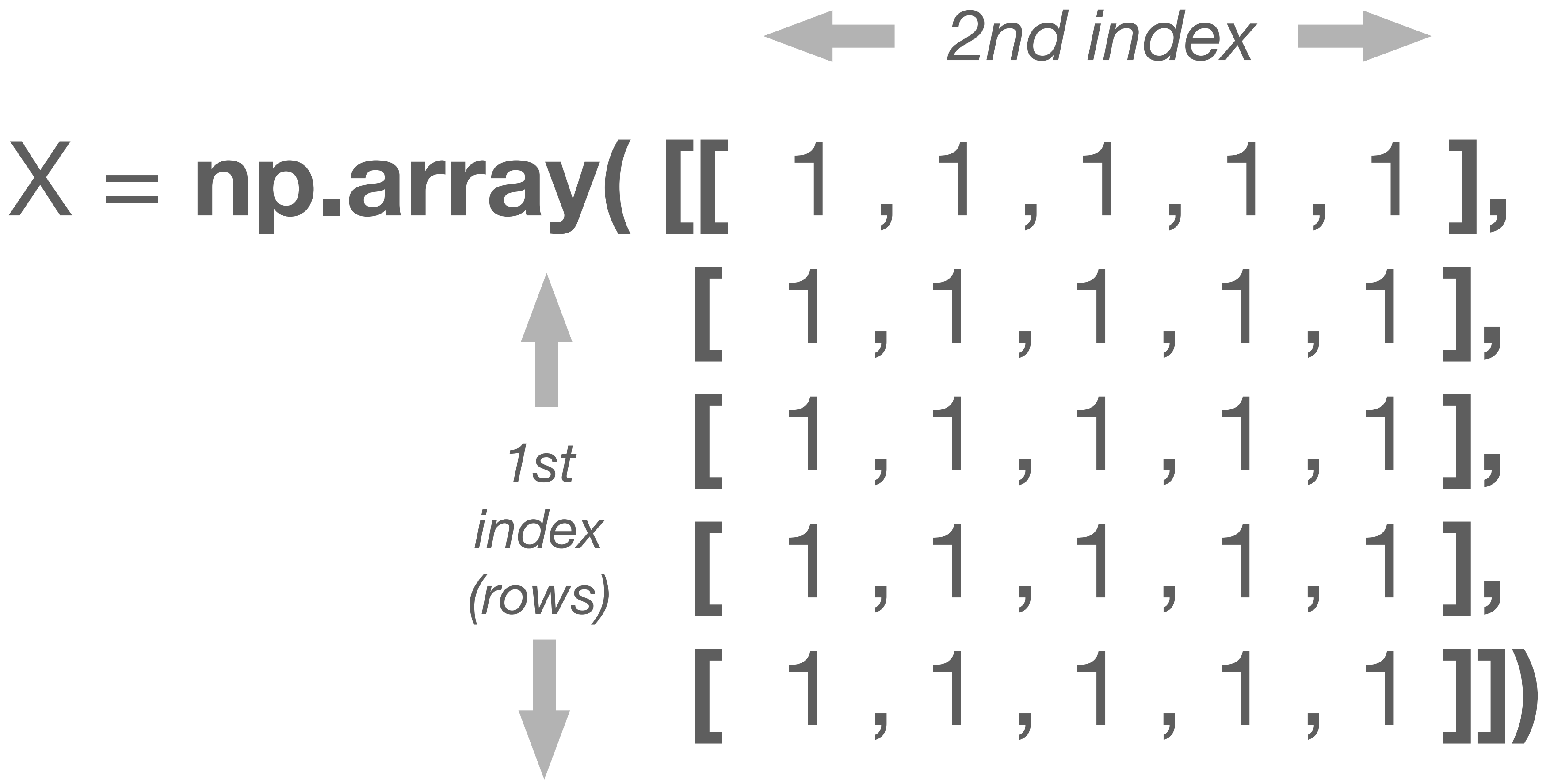
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block

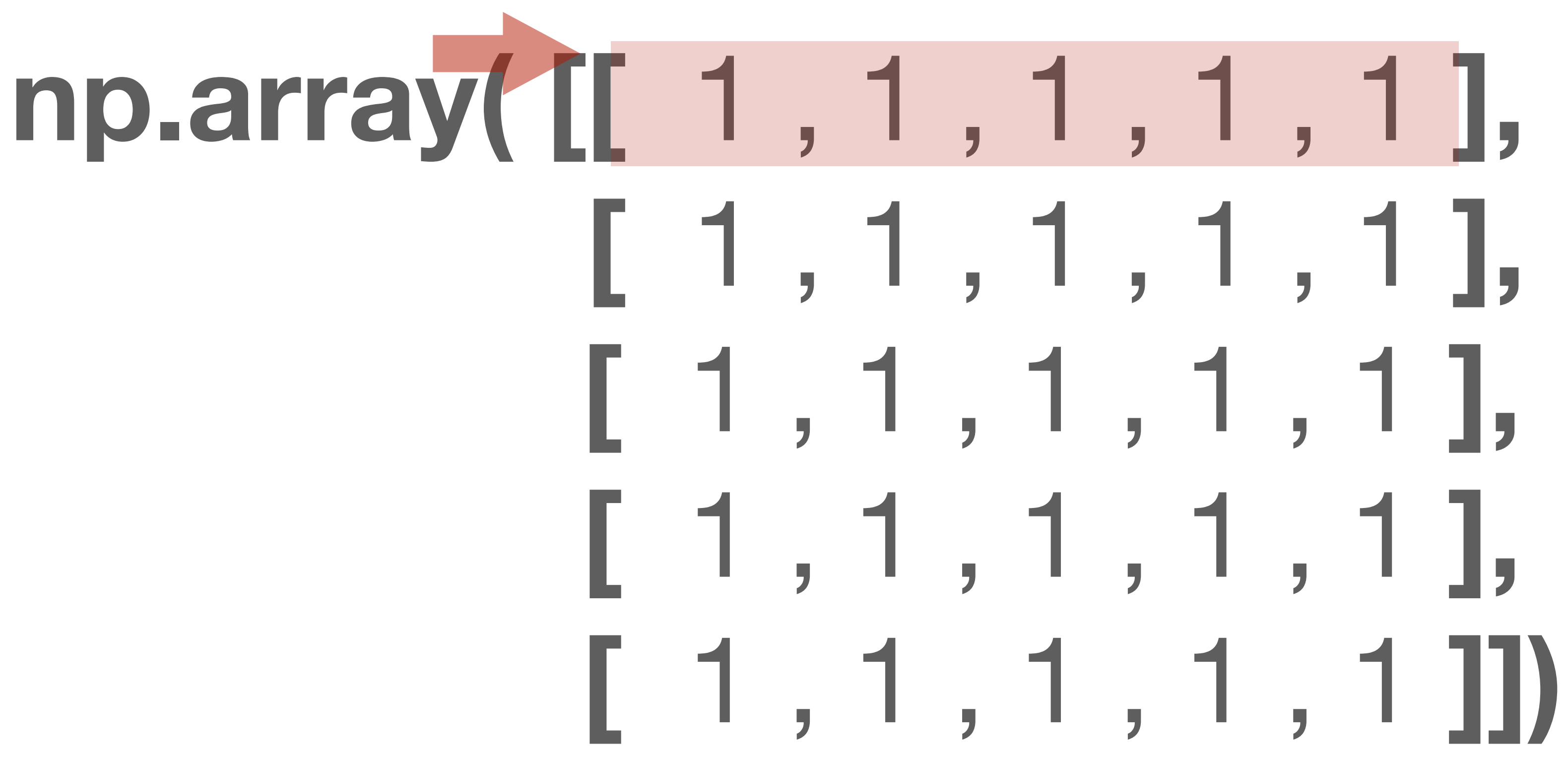


```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```

X[0]

or

X[0,:]



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

Python - Indexing

np.array:

A = **np.array(** [[1, 2, 3],
[3, 2, 1]])

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

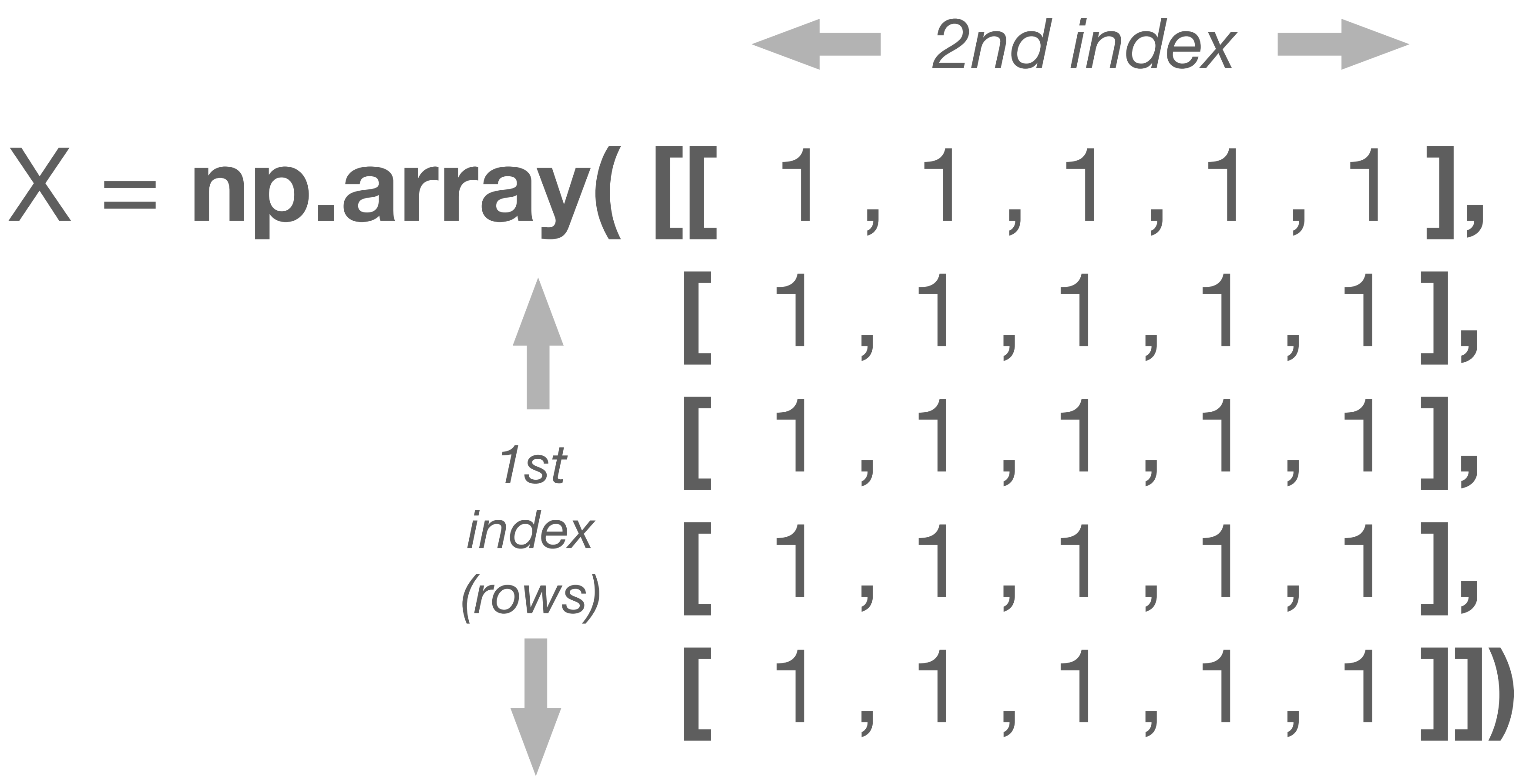
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



2nd index

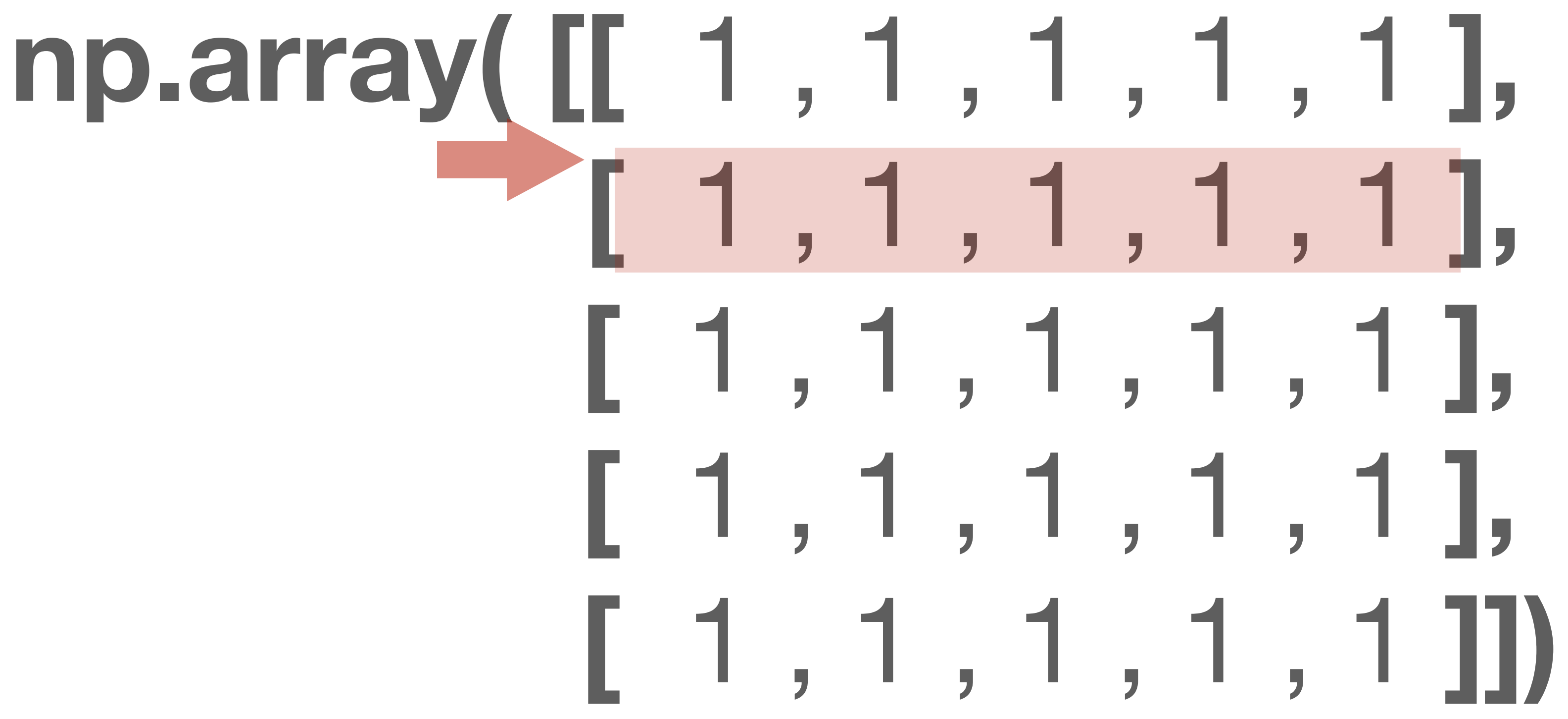
X = **np.array(** [[1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1]])

1st index (rows)

X[1]

or

X[1,:]



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

Python - Indexing

np.array:

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

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

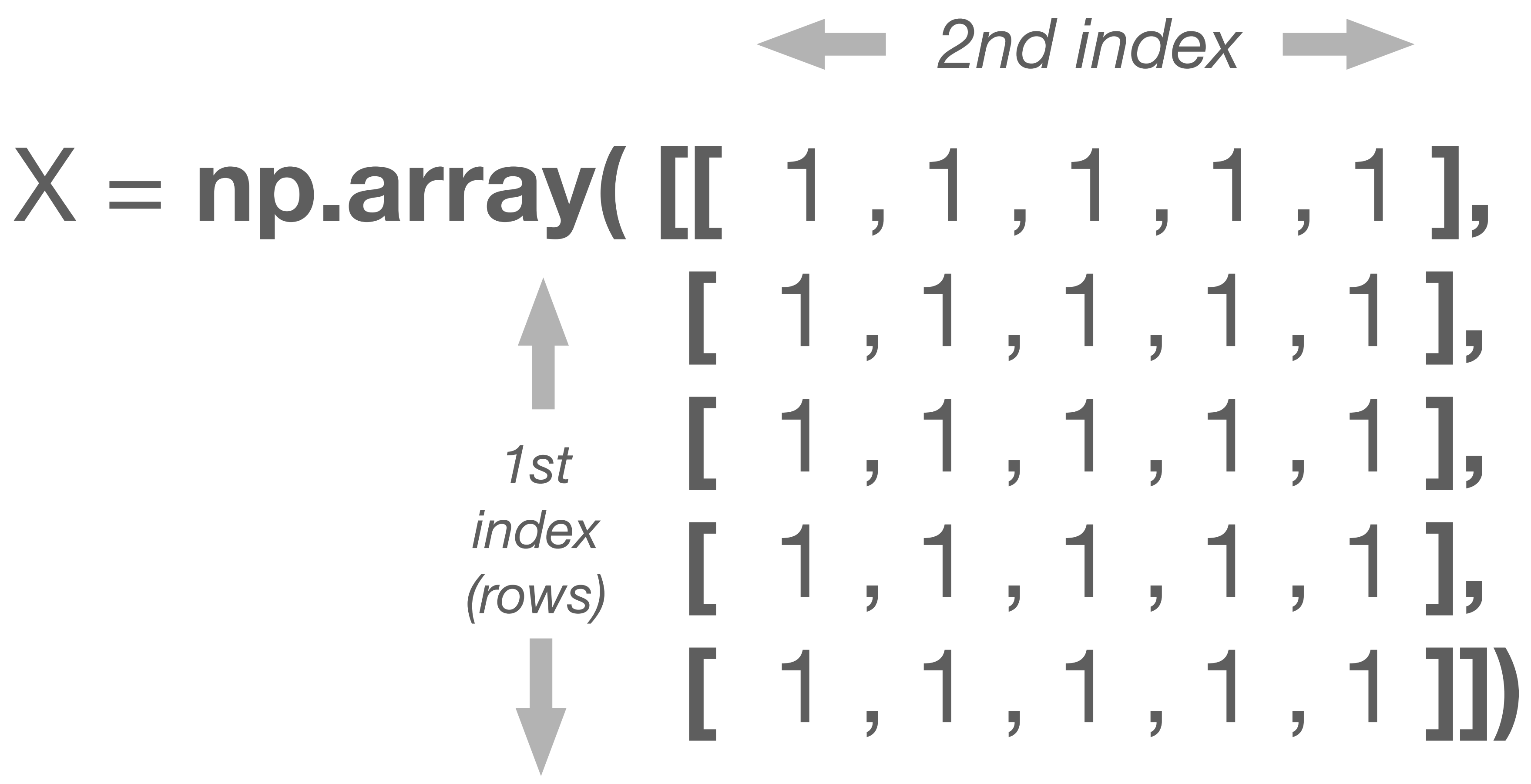
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

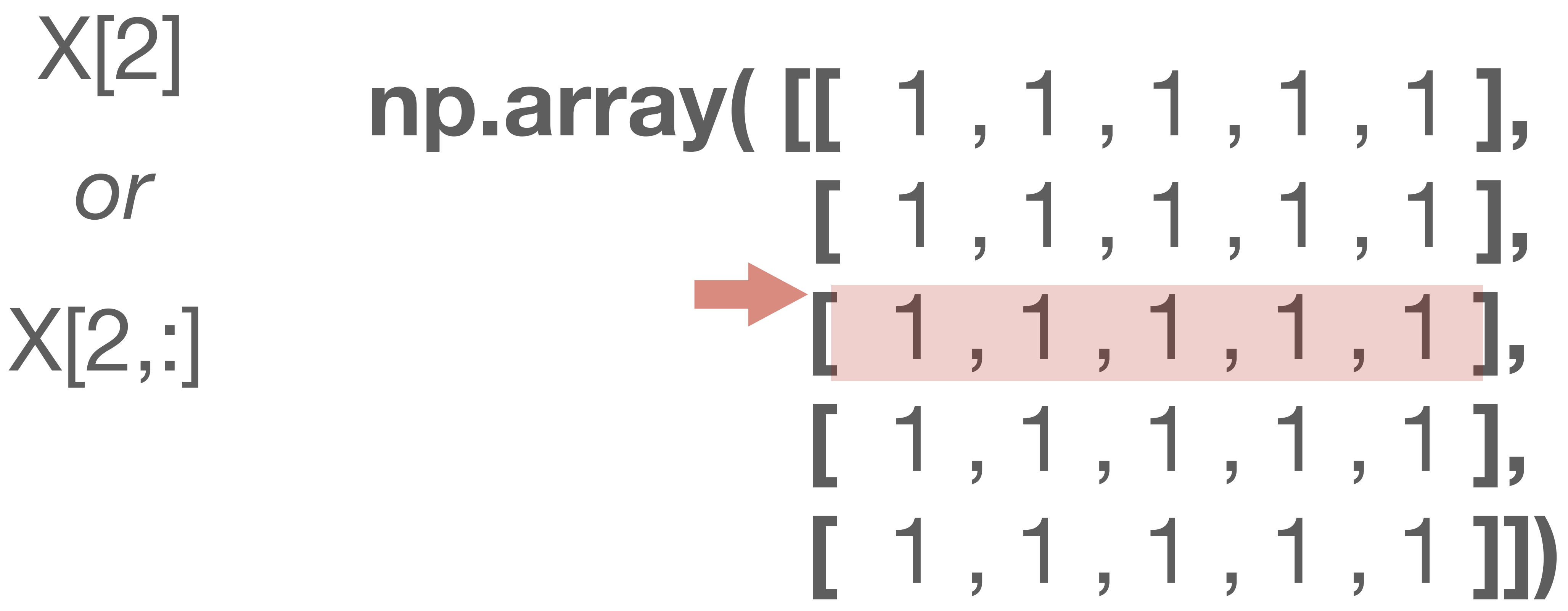
bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```



X[2]

or

X[2,:]

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

Python - Indexing

np.array:

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

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

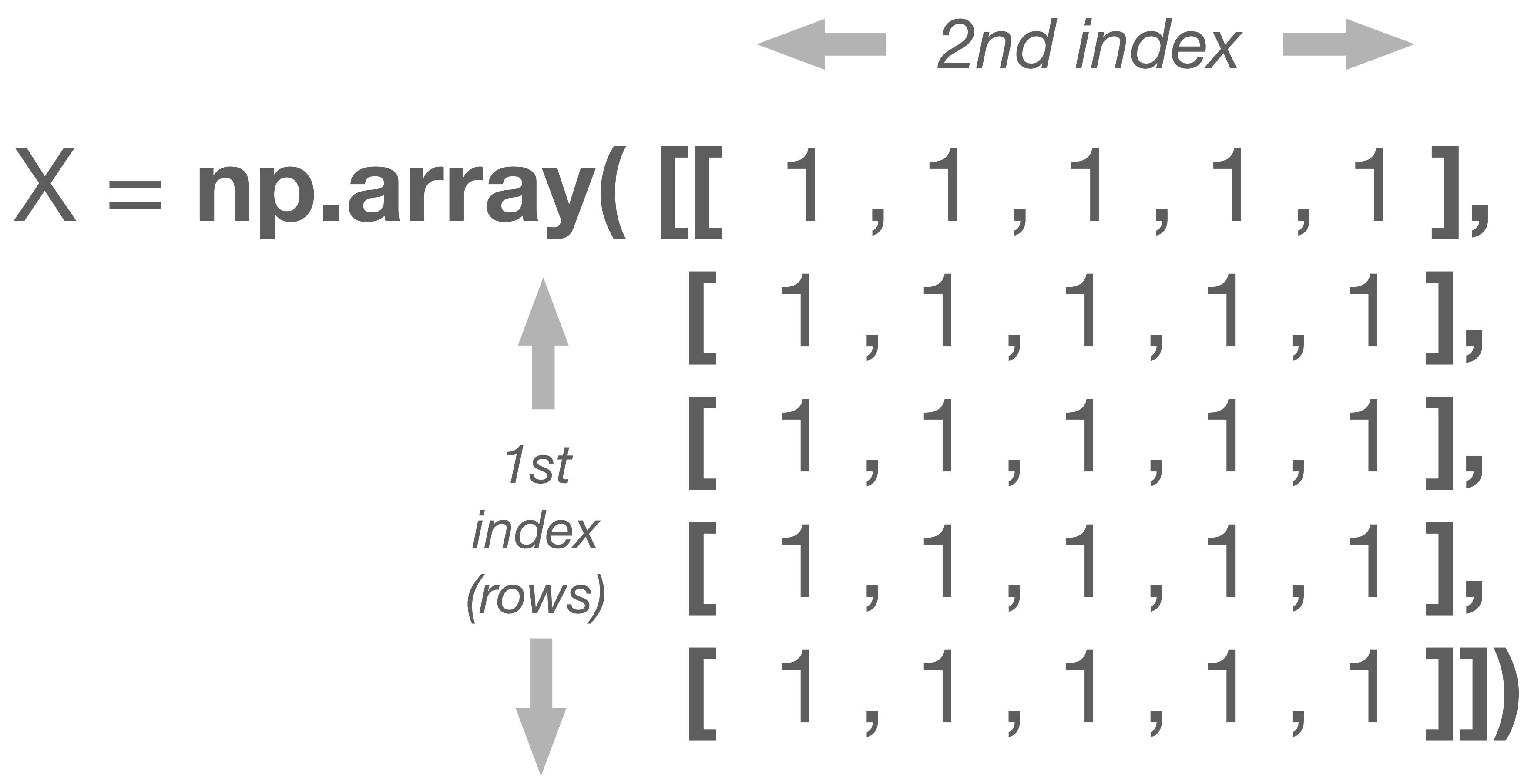
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block

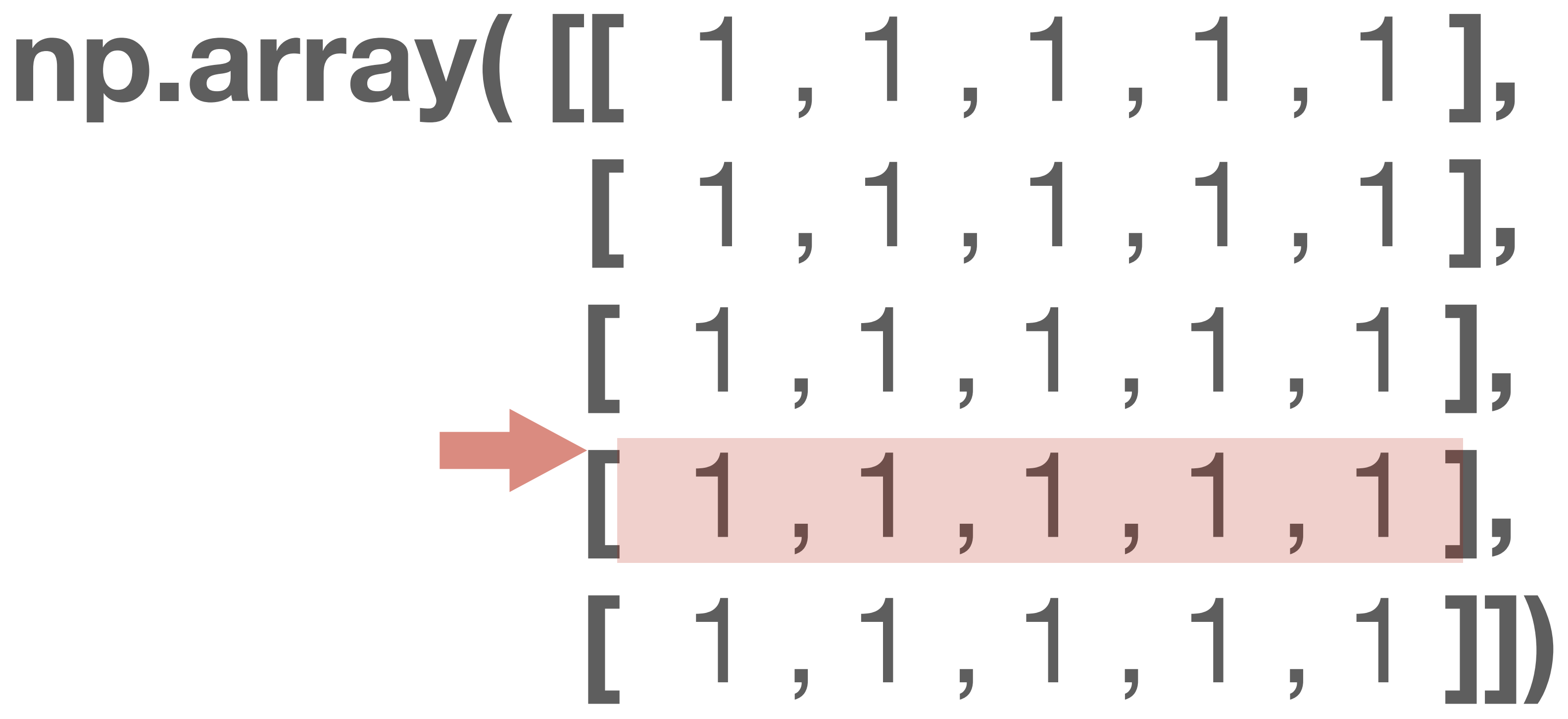


```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```

X[3]

or

X[3,:]



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

Python - Indexing

np.array: A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

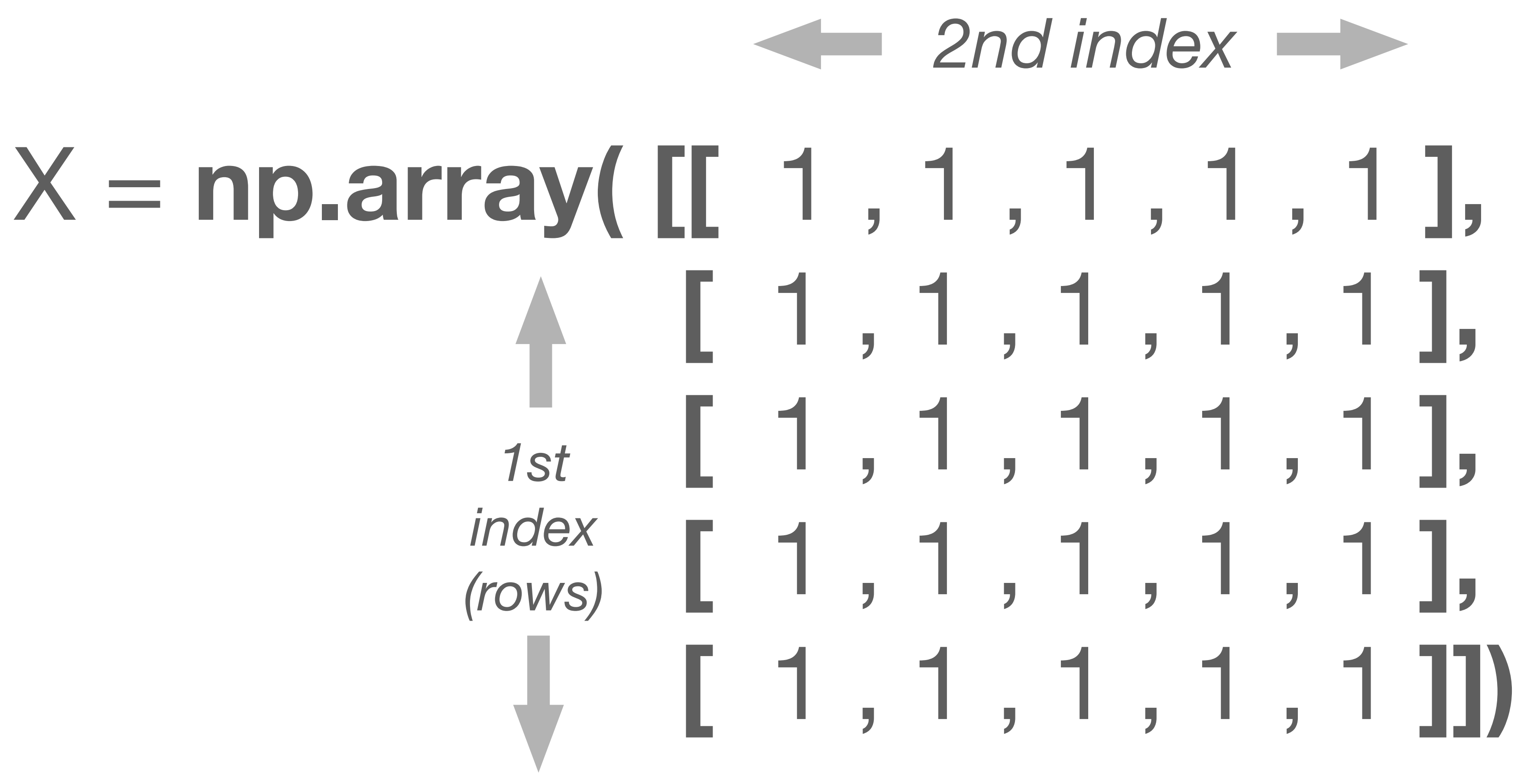
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

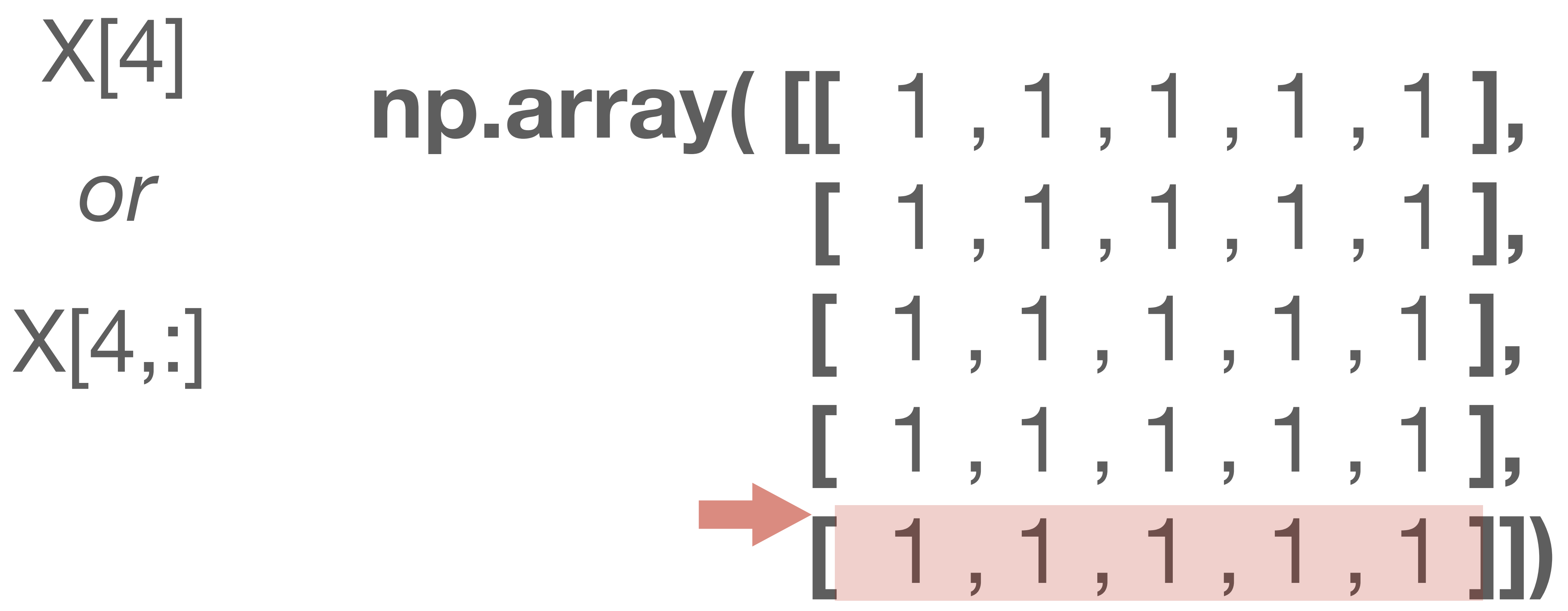
X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



2nd index

```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]])
```

1st index (rows)



X[4]

or

X[4,:]

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

Python - Indexing

np.array:

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

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

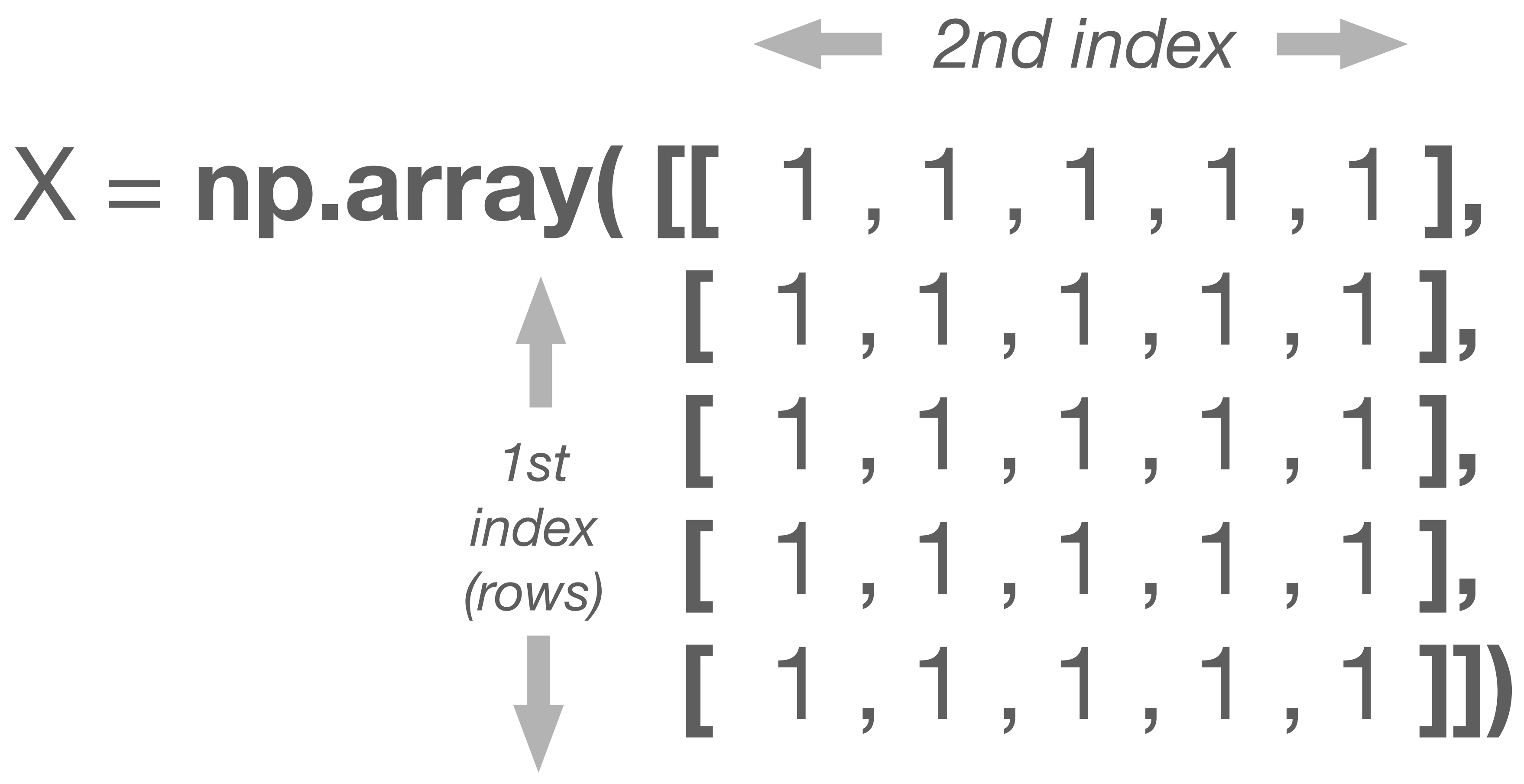
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

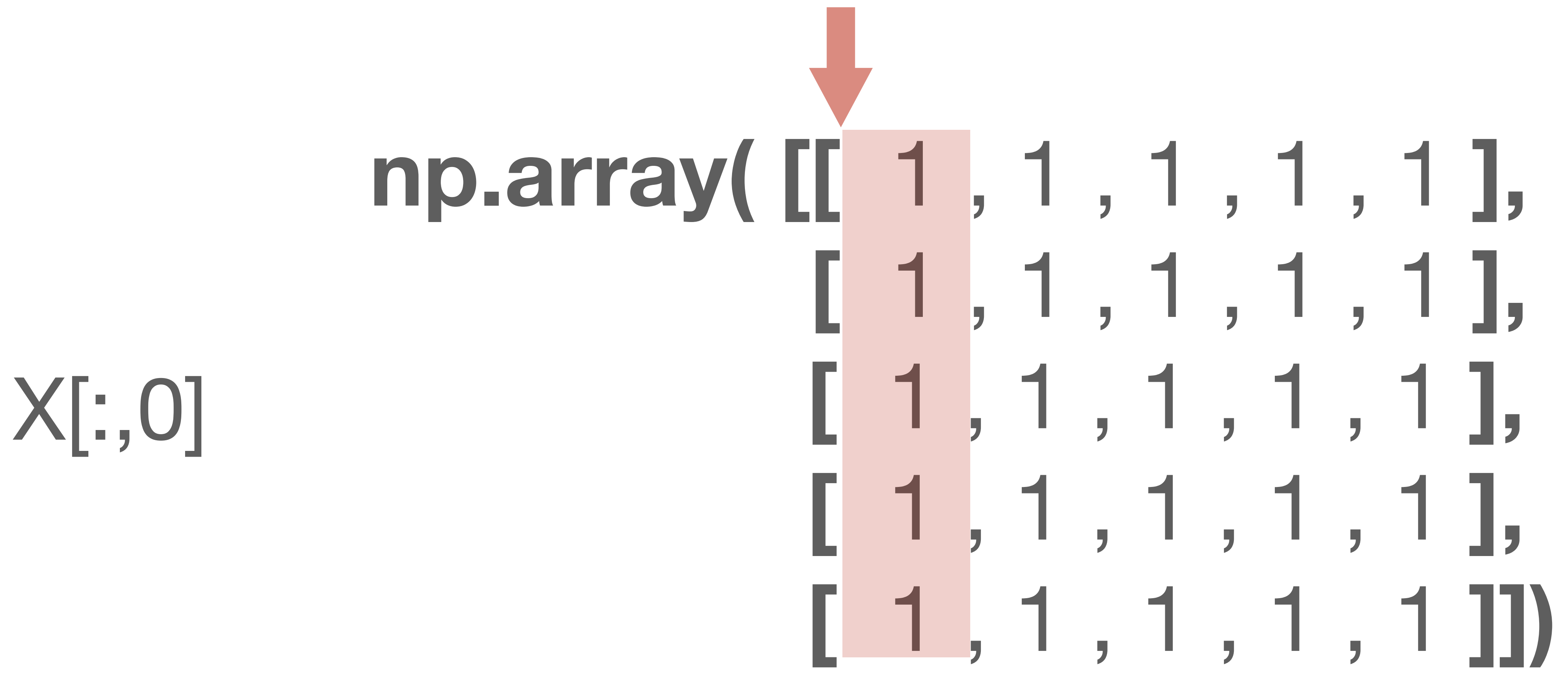
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```



X[:,0]

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

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

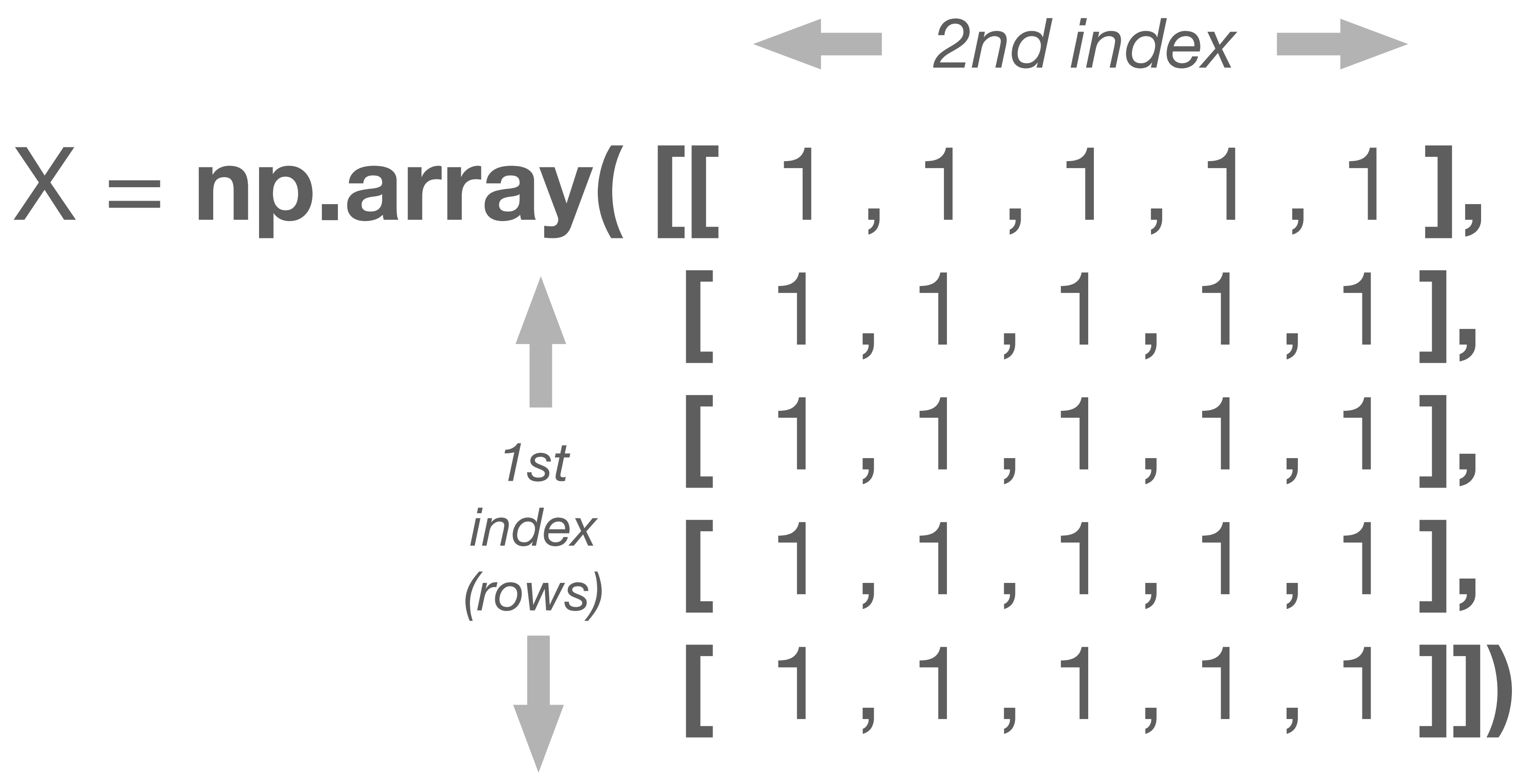
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

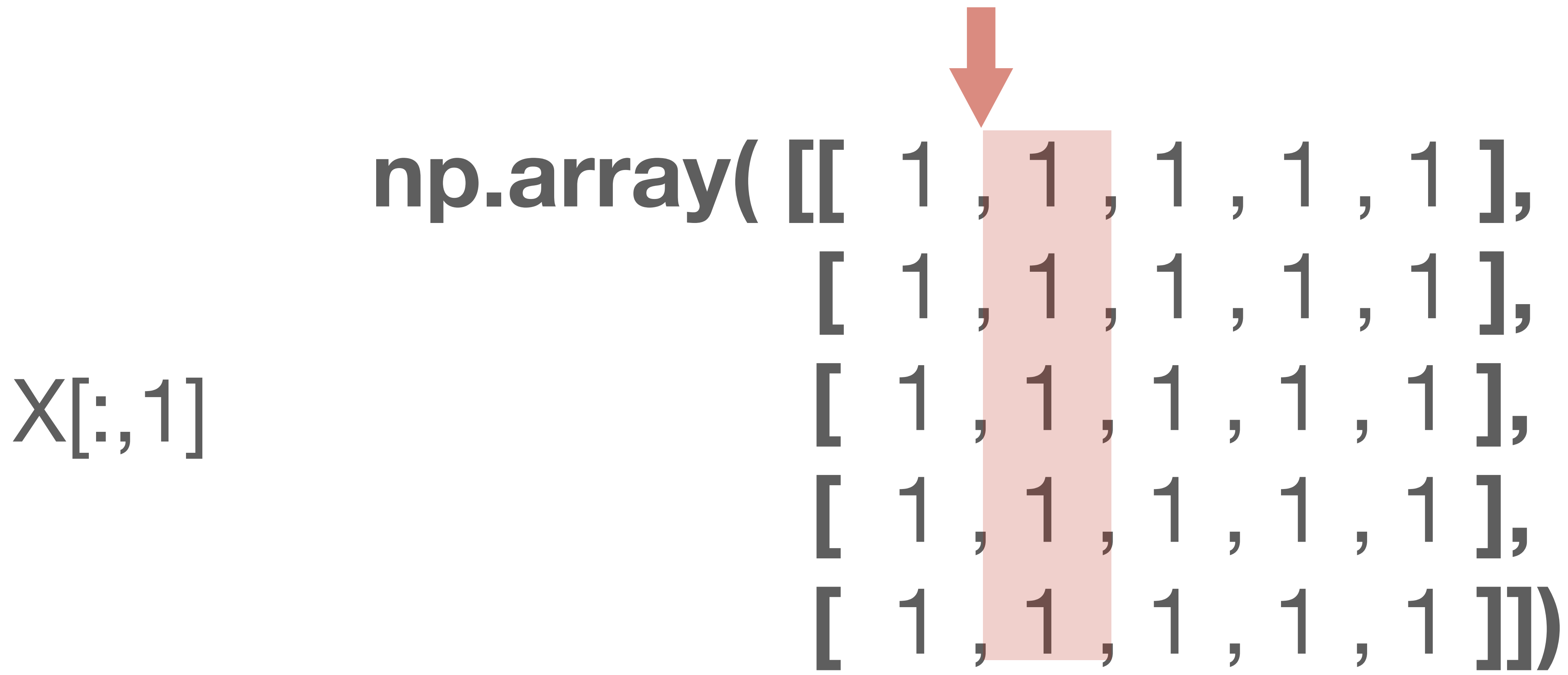
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```



X[:,1]

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

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

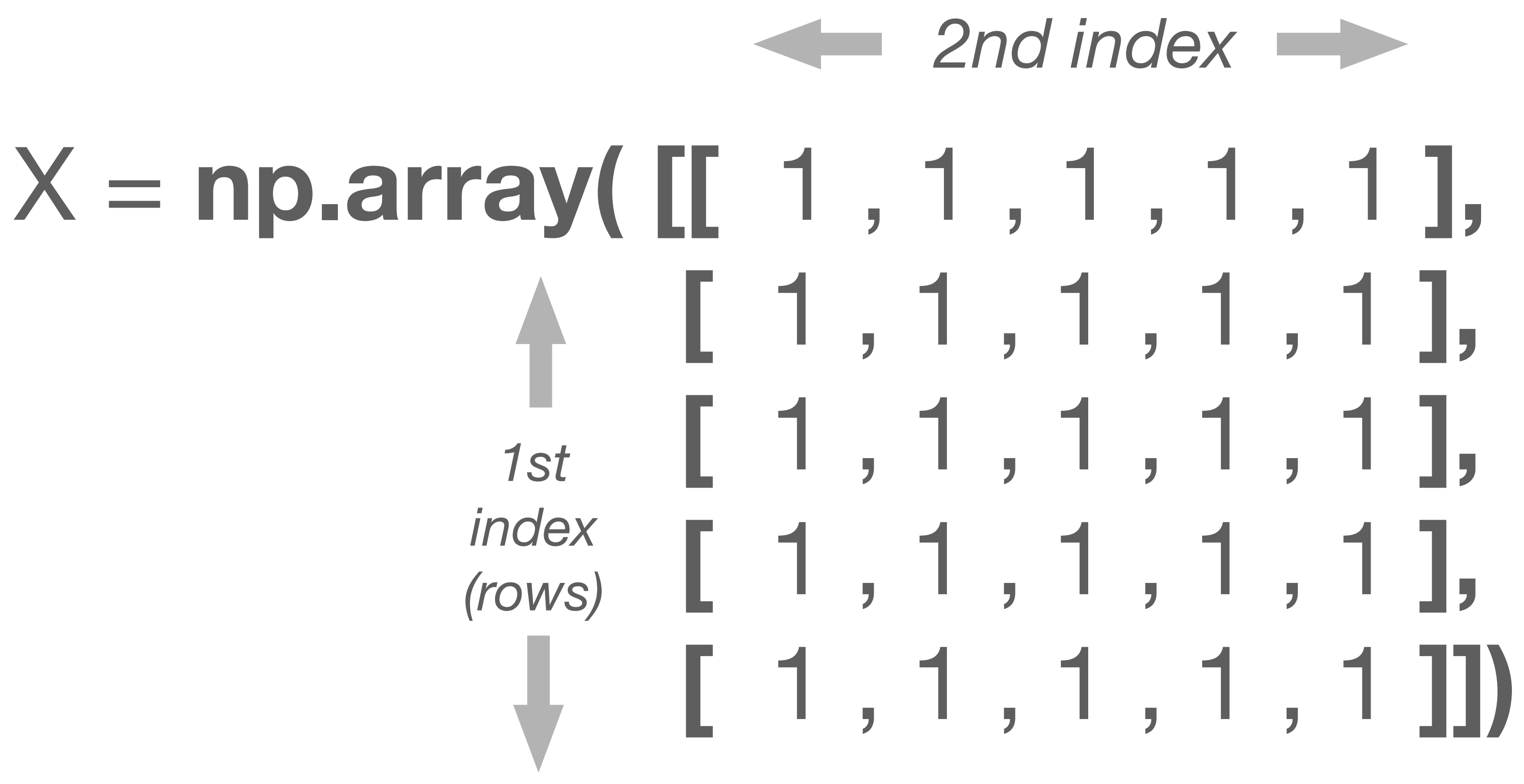
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

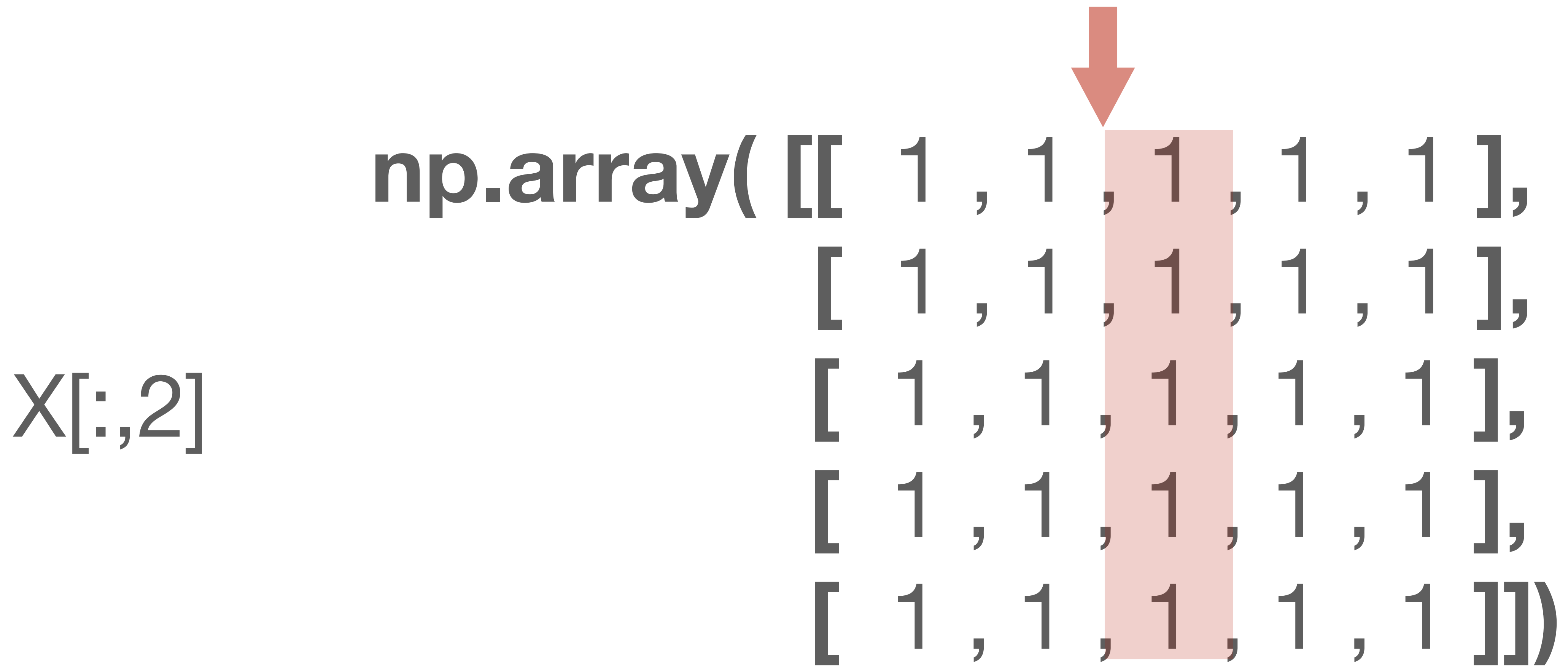
X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



2nd index

X = **np.array([[** 1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **]])**

1st index (rows)



X[:,2]

np.array([[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **]])**

Python - Indexing

np.array: A = **np.array(** **[[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

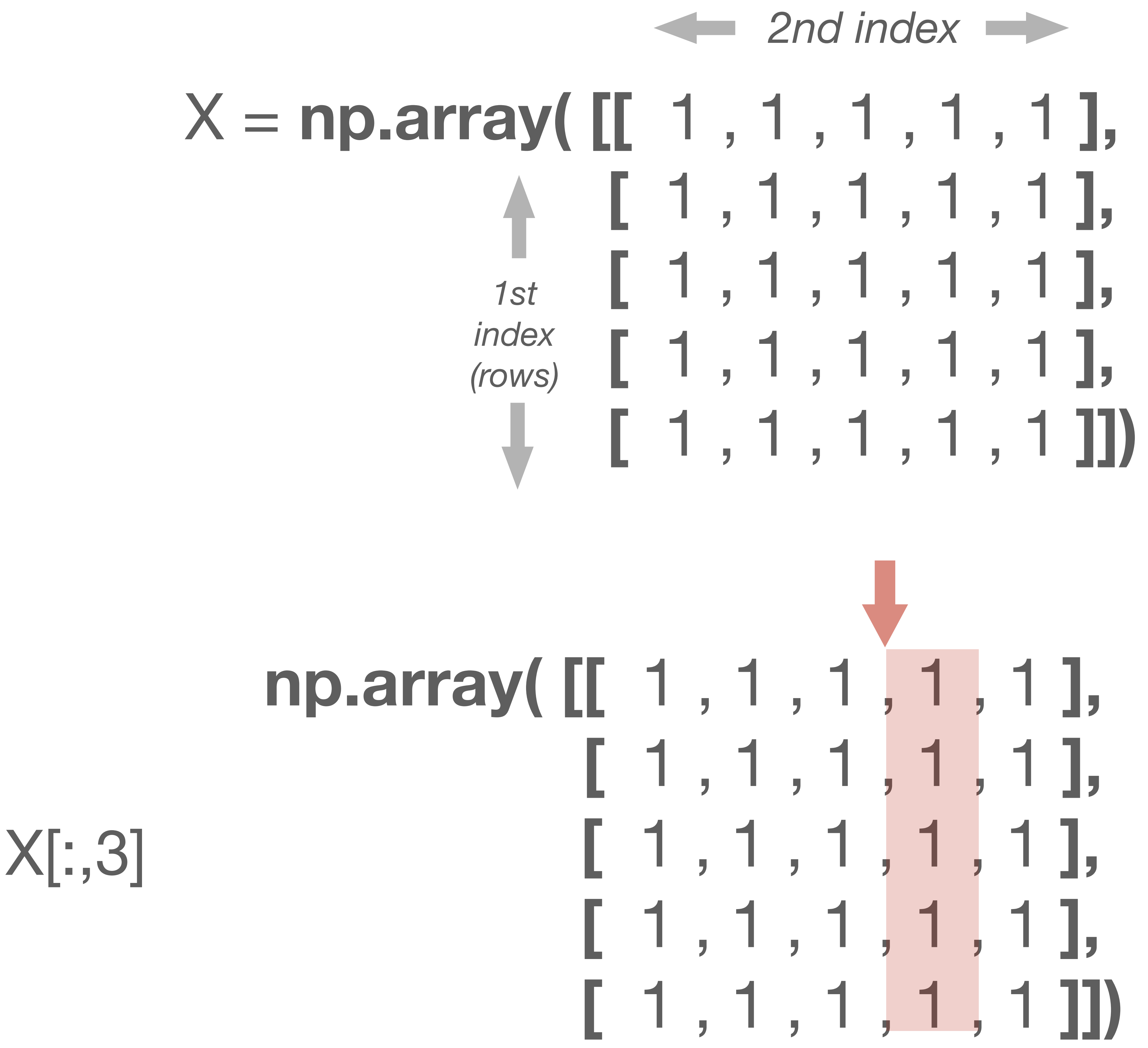
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```

X[:,3]

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

Python - Indexing

np.array: A = **np.array(** [[1, 2, 3],
[3, 2, 1]] **)**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

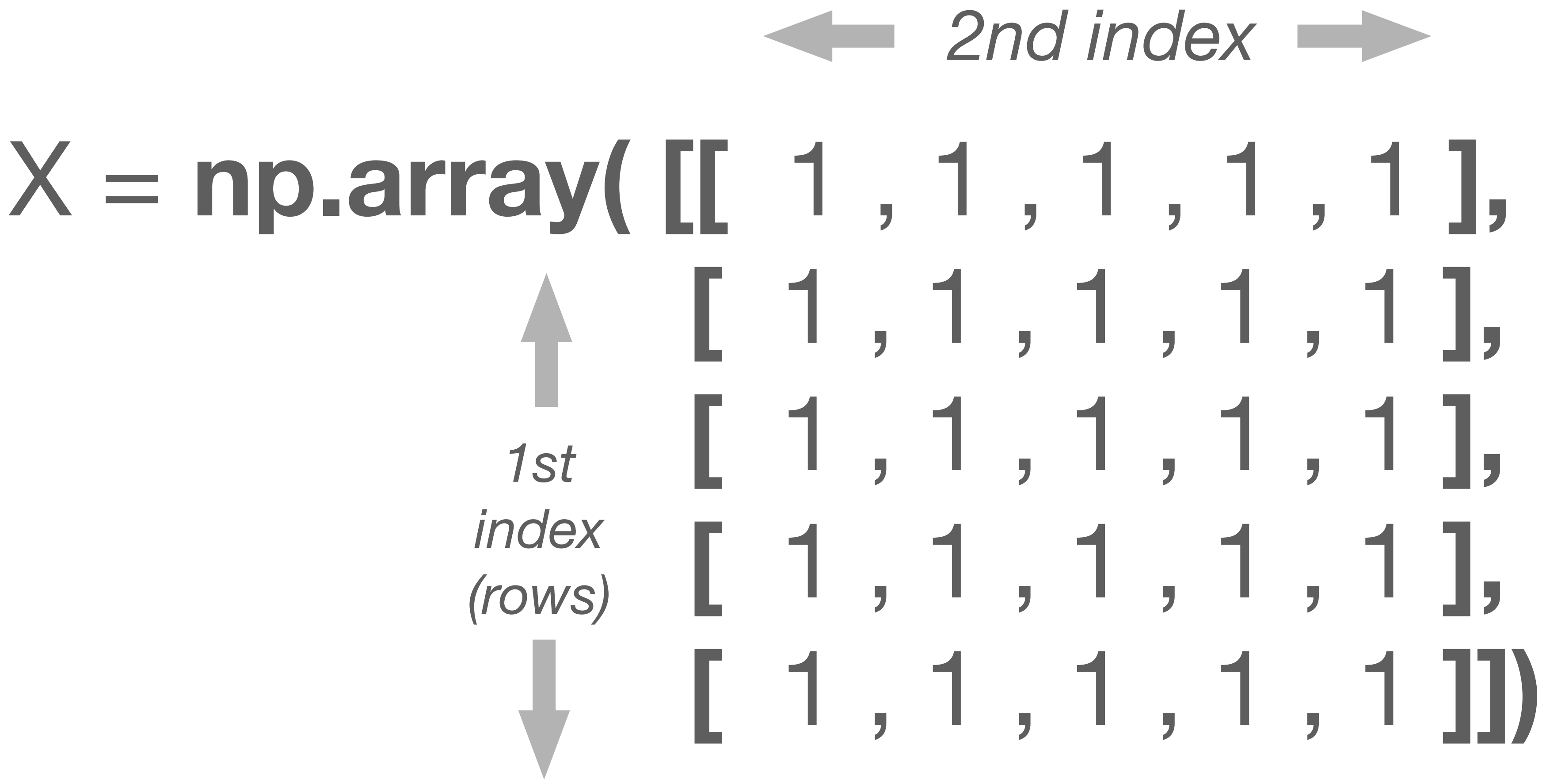
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

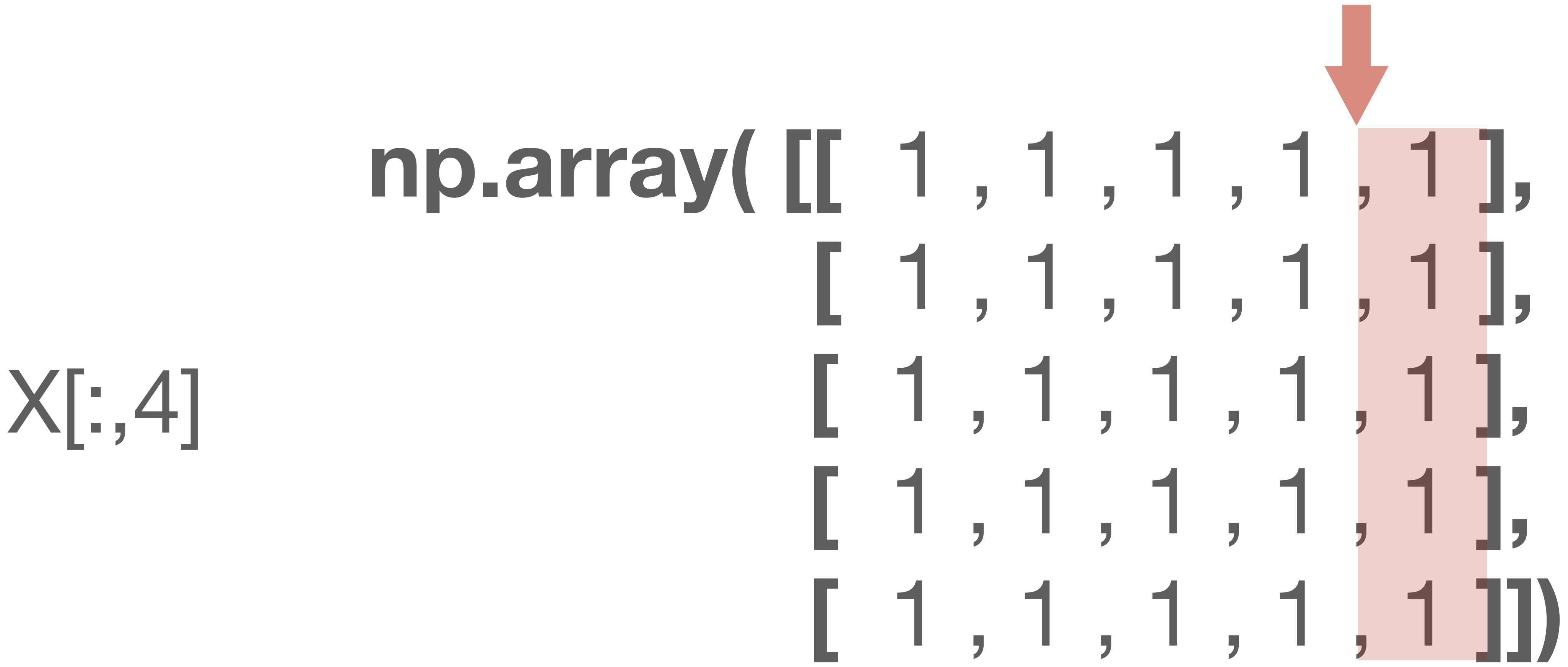
bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```



X[:,4]

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

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

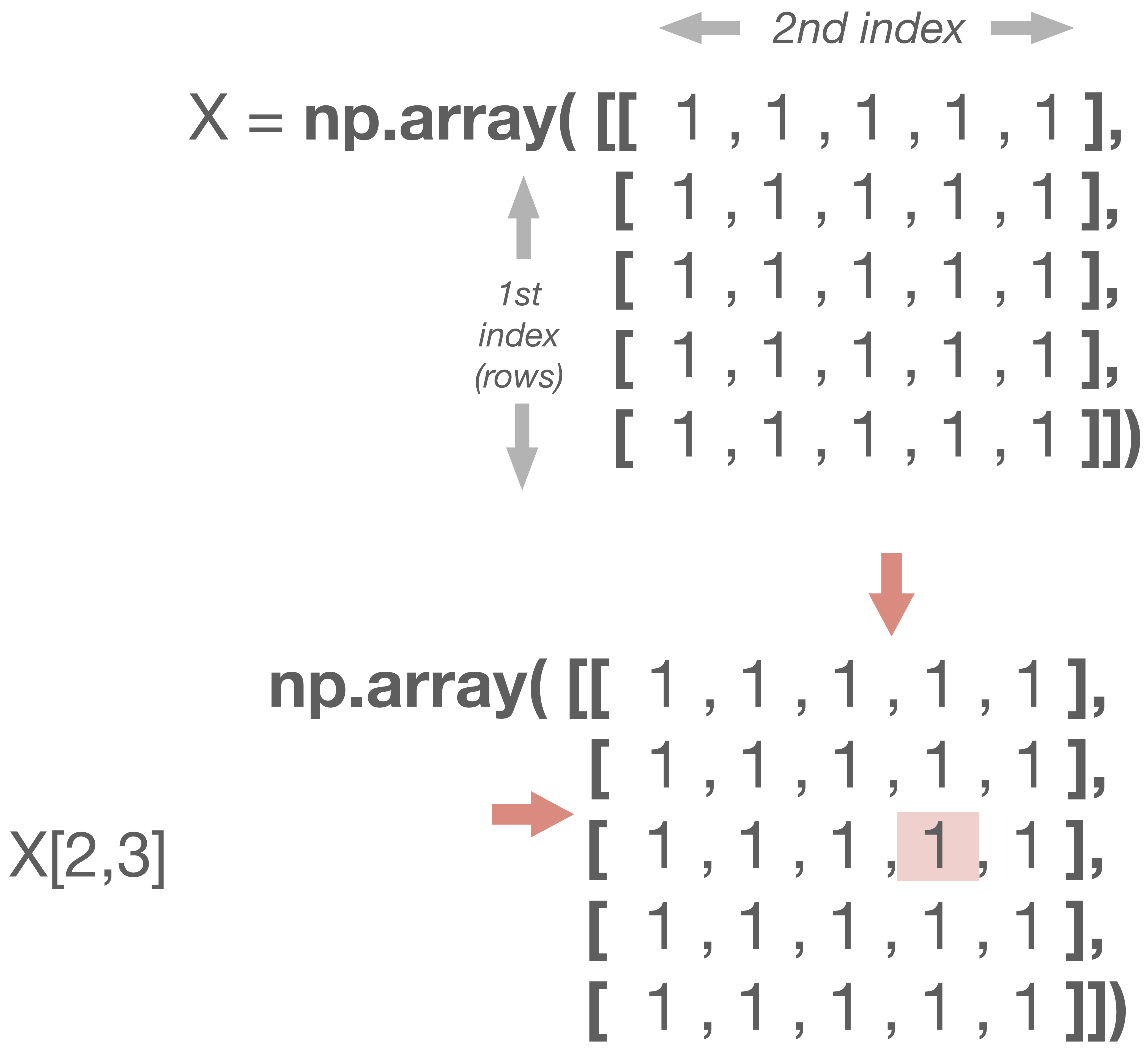
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```

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

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

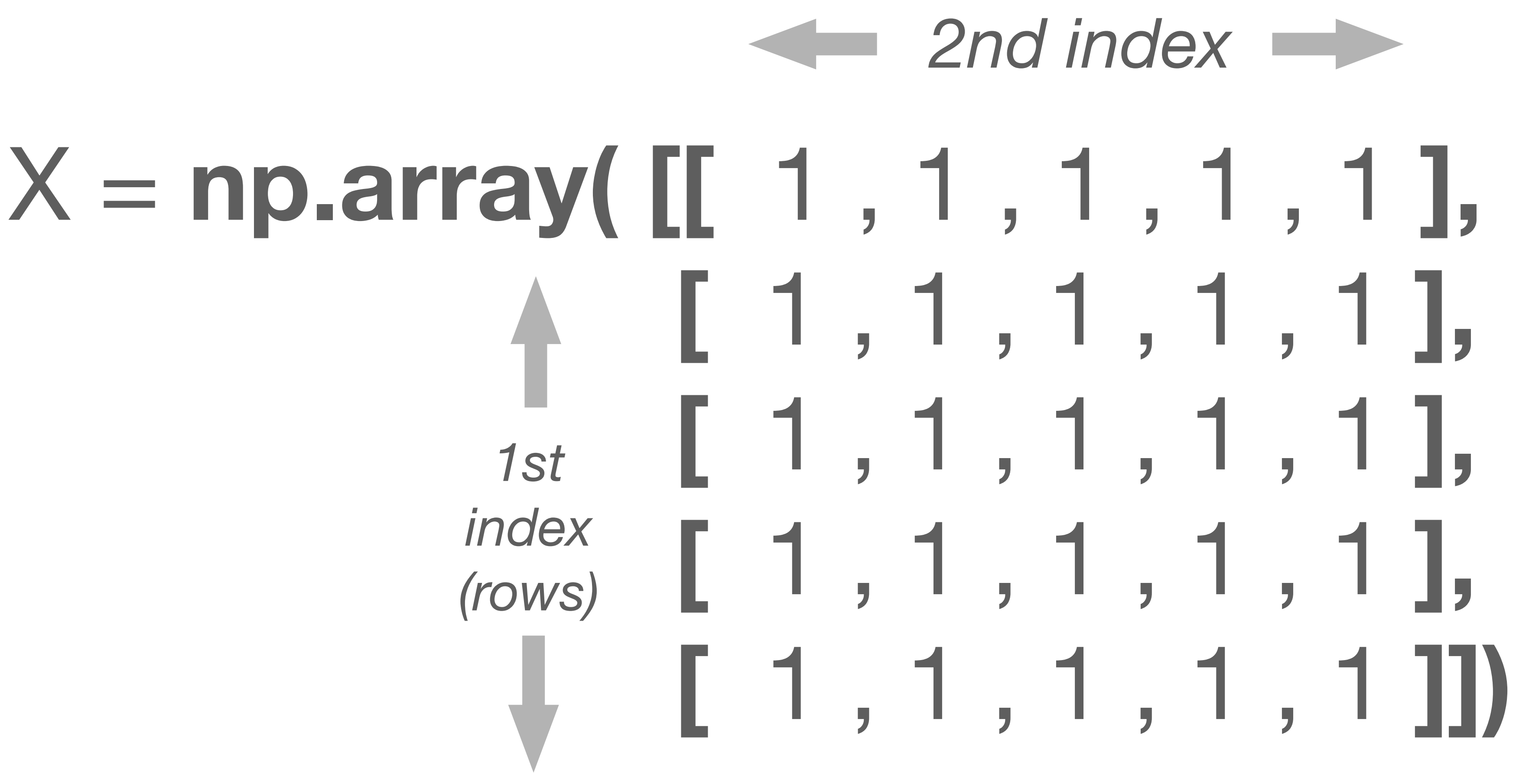
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

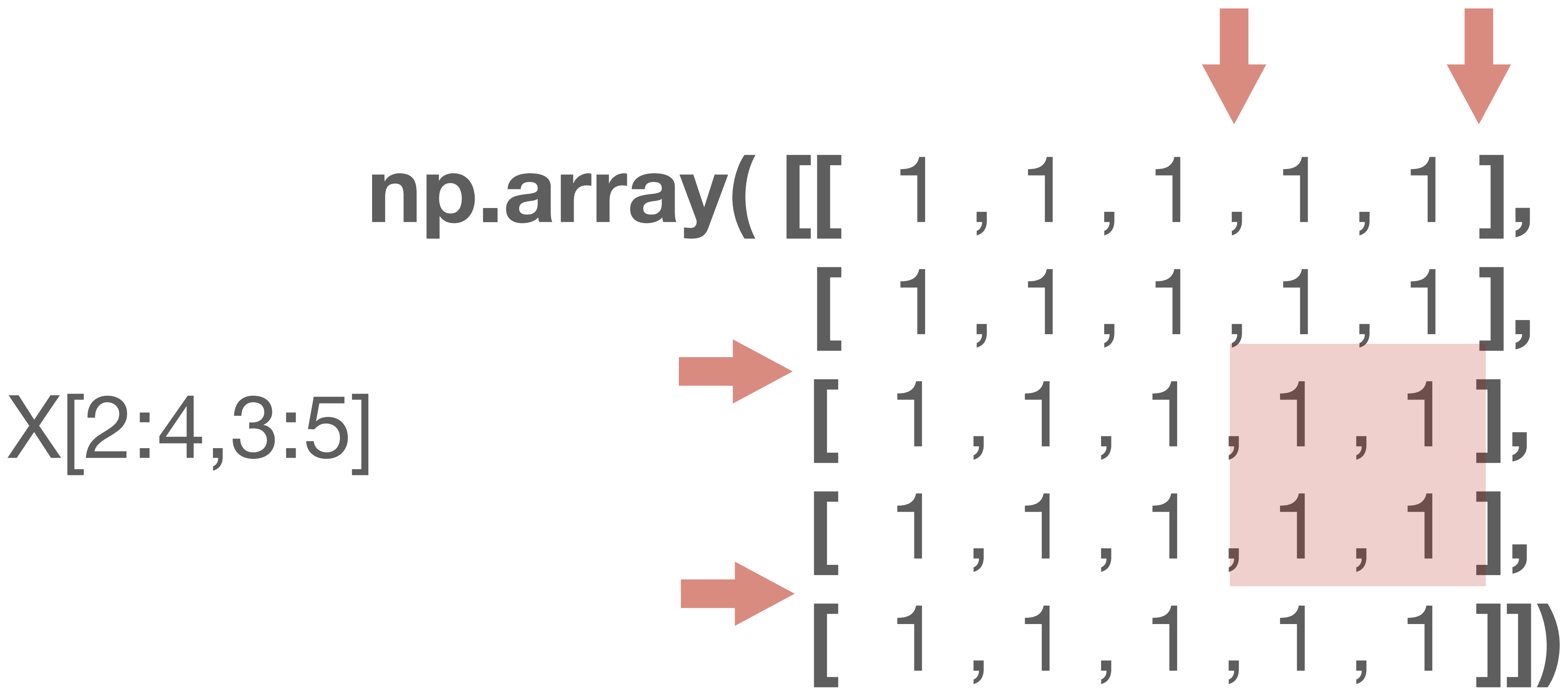
X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



2nd index

1st index (rows)

```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```



X[2:4,3:5]

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

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

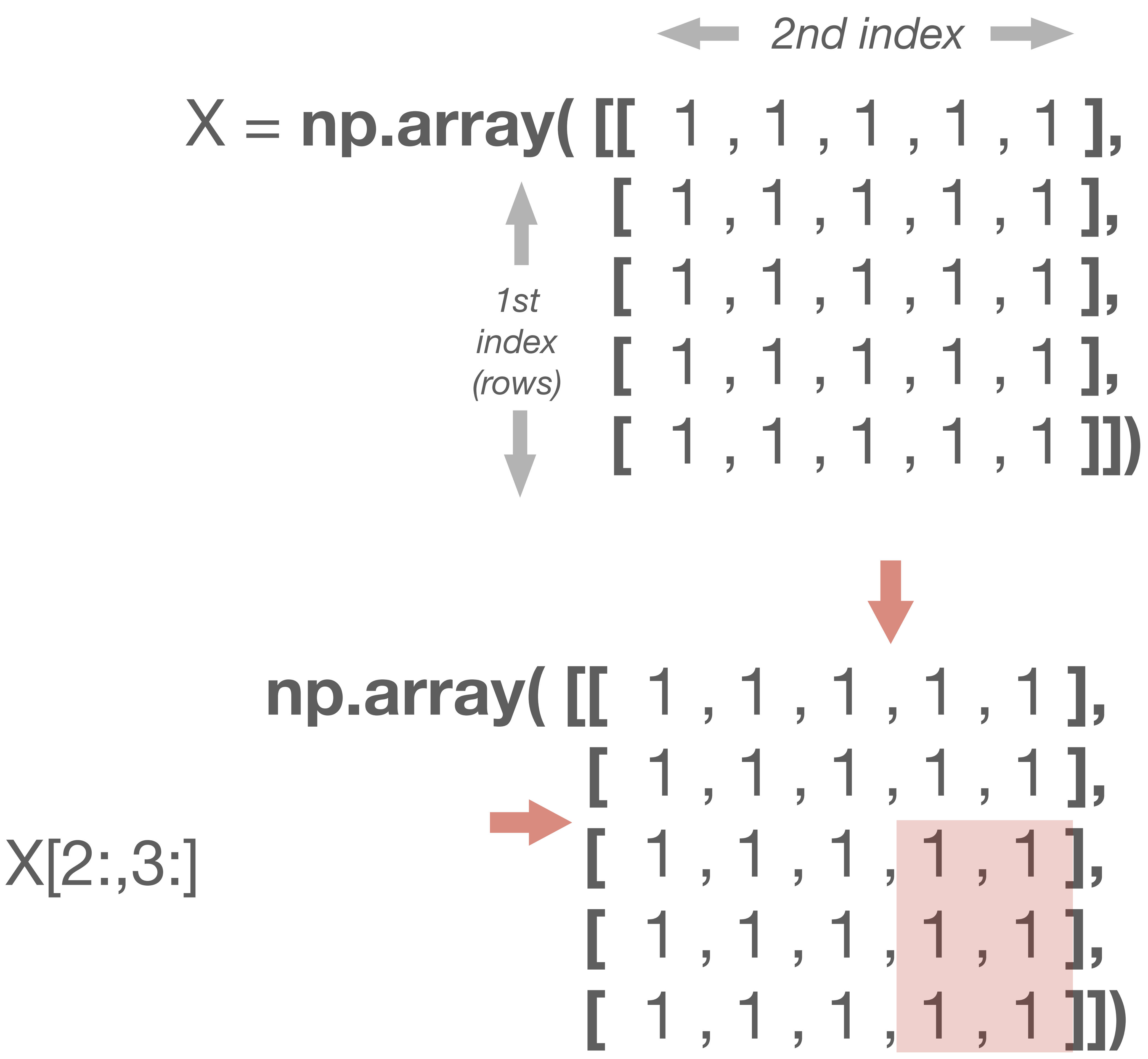
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]])
```

X[2:,3:]

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

Python - Indexing

np.array:

A = **np.array(** **[[**1, 2, 3**]**,
[3, 2, 1**]]** **)**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

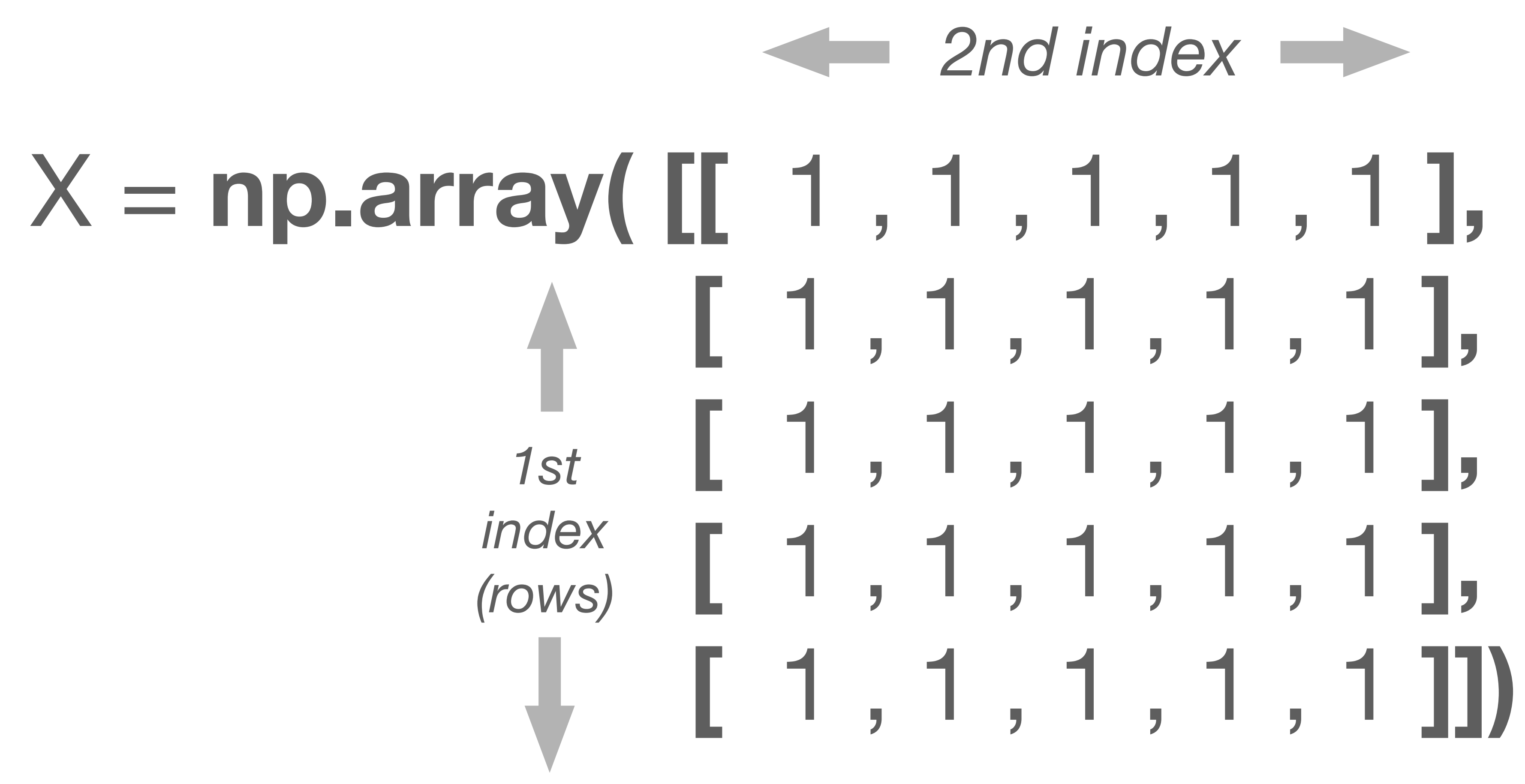
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block

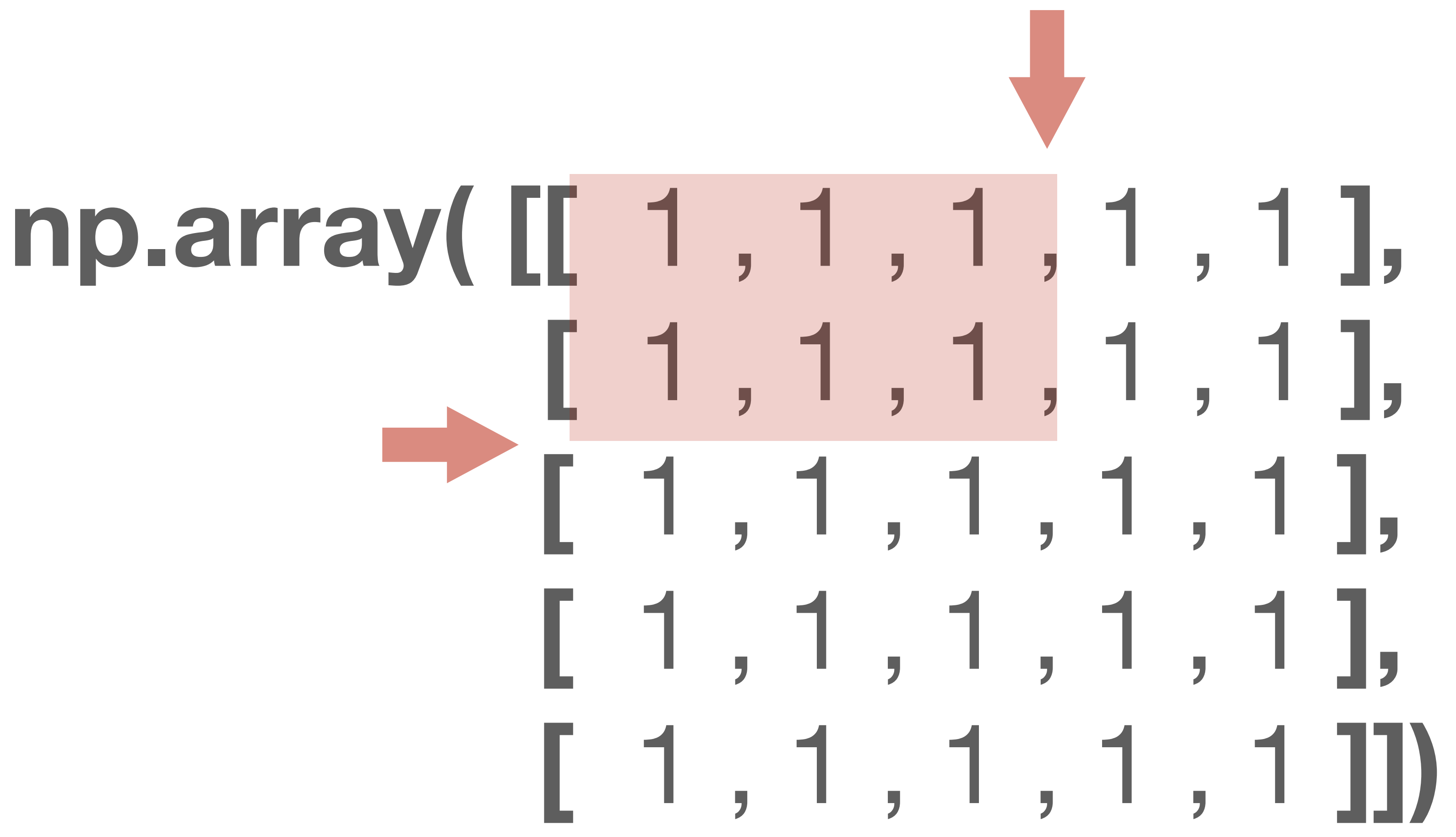


2nd index

1st index (rows)

```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```

X[:2,:3]



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

Python - Indexing

np.array: A = np.array([[1, 2, 3], [3, 2, 1]])

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing   start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind]   - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2]   - returns [0,0],[2,3], and [3,2] elements

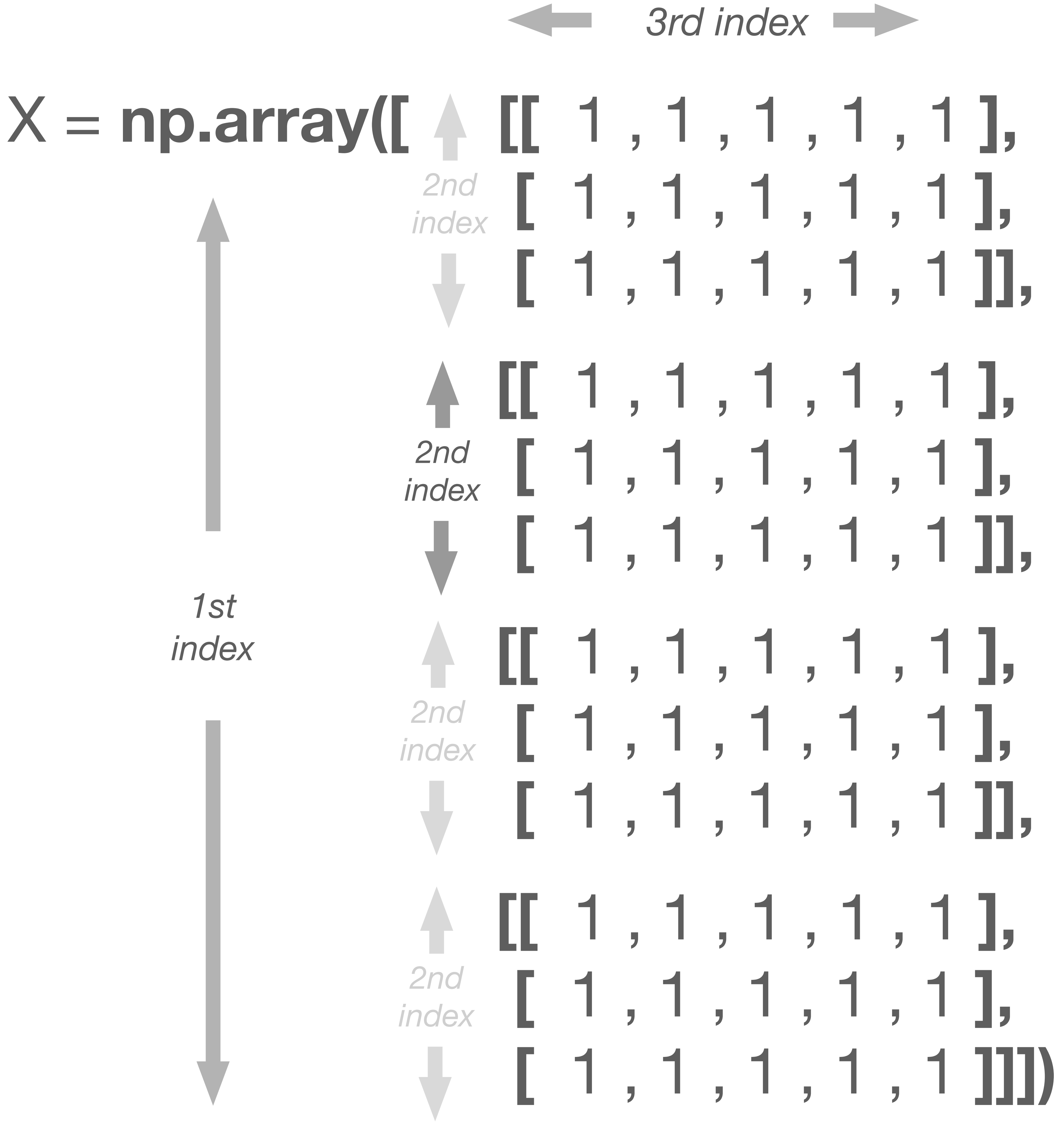
boolean indexing

bool = **[True, True, False, True];**   MUST BE ARRAY LENGTH
x[bool]   - returns 0,1, and 3 element.

X[bool,bool]   - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



3rd index

```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],

               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],

               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],

               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

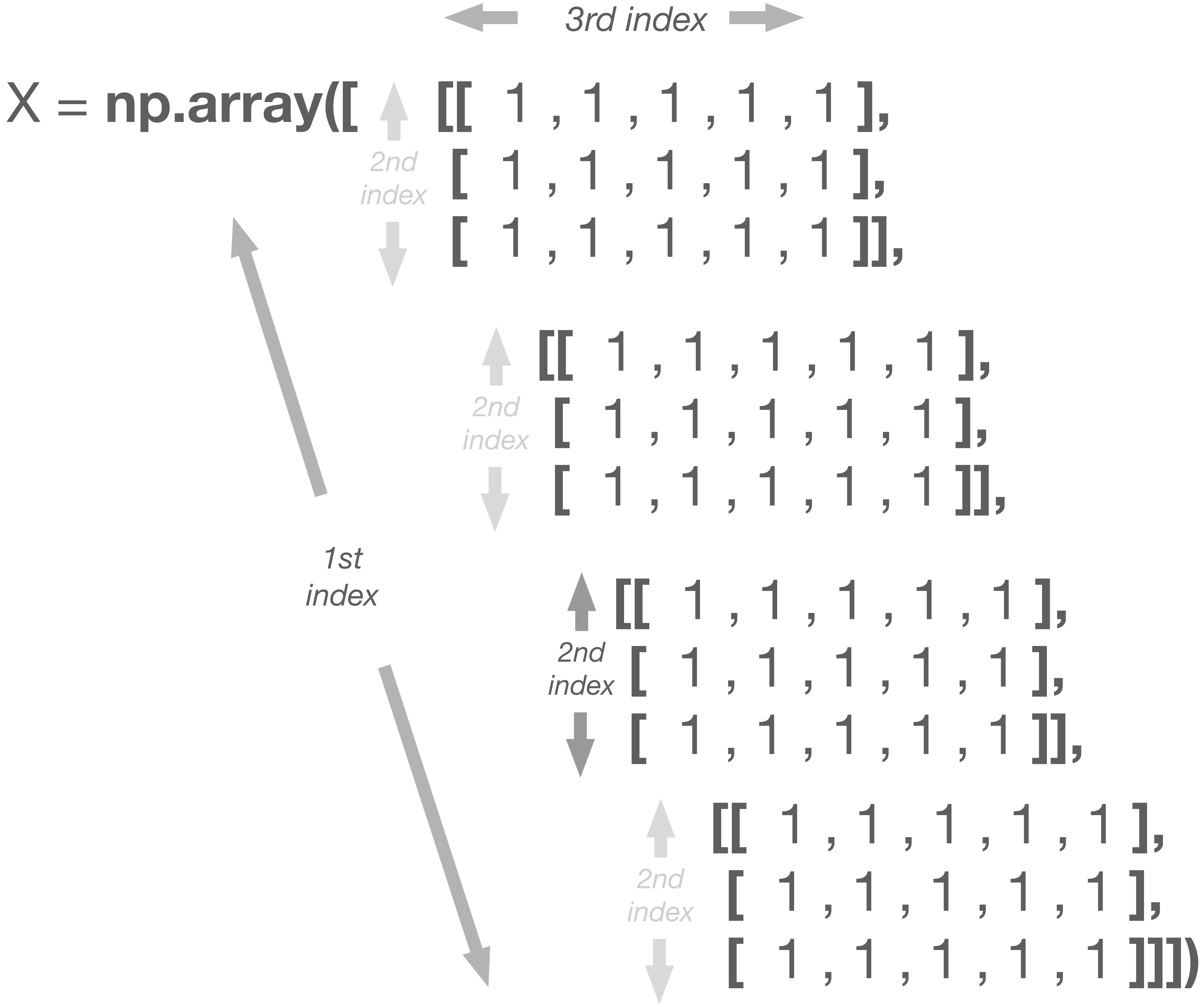
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ]],
                   [[ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ]],
                     [[ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind]   - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2]   - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];**   MUST BE ARRAY LENGTH
x[bool]   - returns 0,1, and 3 element.
X[bool,bool]   - returns the [0,0], [1,1], and [3,3]

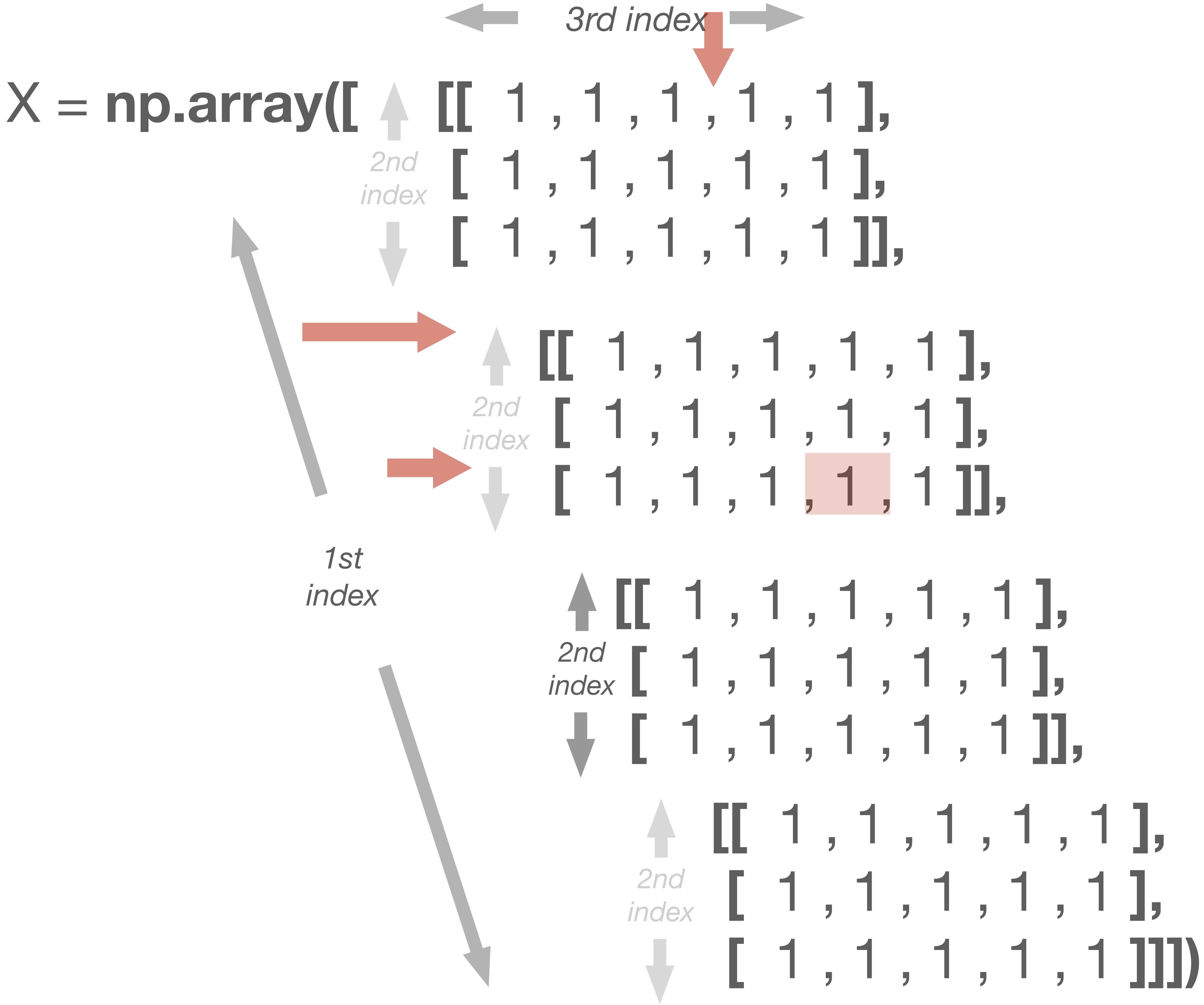
block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block

X[1,2,3]

or

X[1][2][3]



```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
                  [[ 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 ]],
                    [[ 1 , 1 , 1 , 1 , 1 ],
                     [ 1 , 1 , 1 , 1 , 1 ],
                     [ 1 , 1 , 1 , 1 , 1 ]],
                      [[ 1 , 1 , 1 , 1 , 1 ],
                       [ 1 , 1 , 1 , 1 , 1 ],
                       [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block

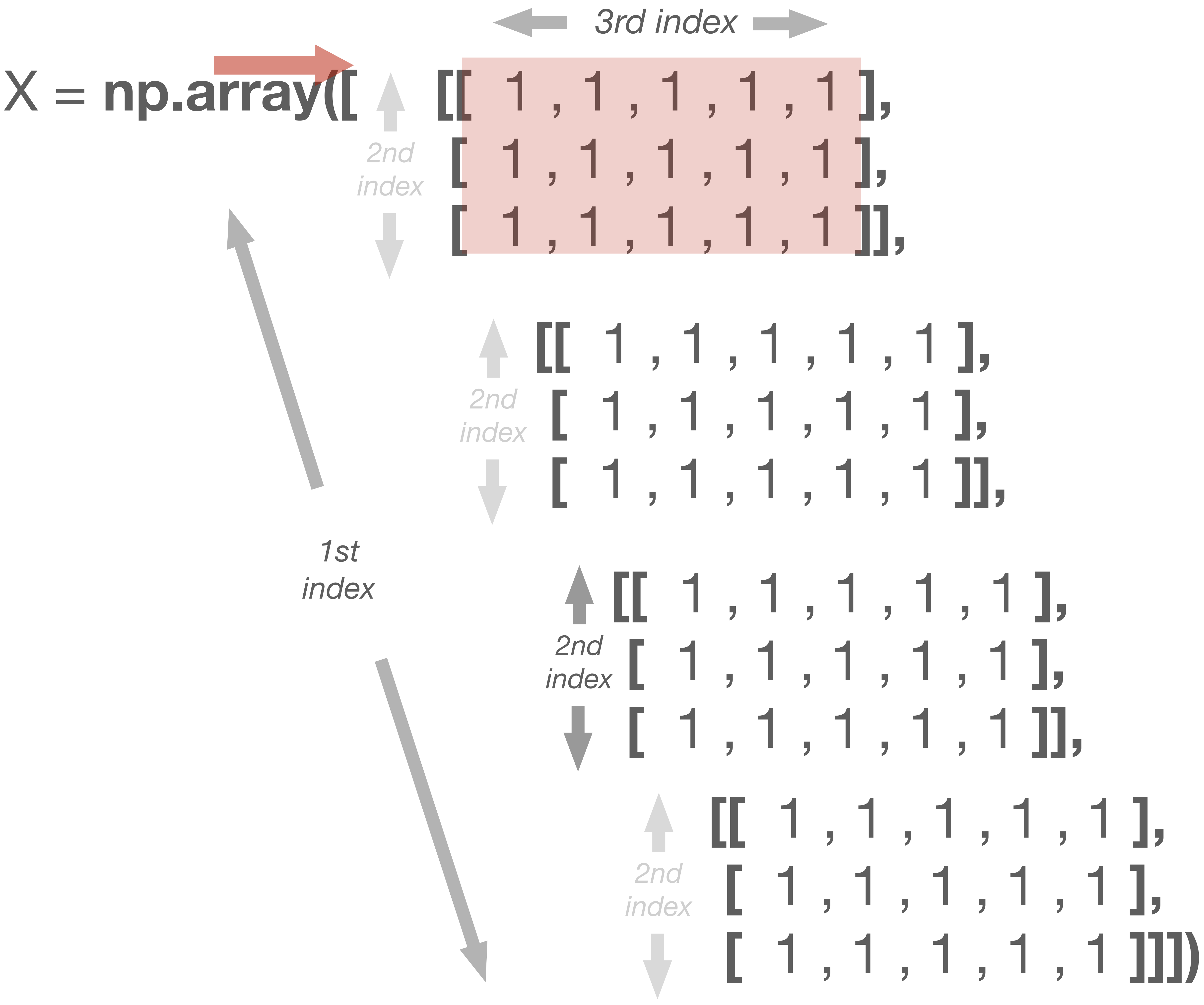
X[0]

or

X[0,:]

or

X[0,:,:]



```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ]],
                   [[ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ]],
                     [[ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

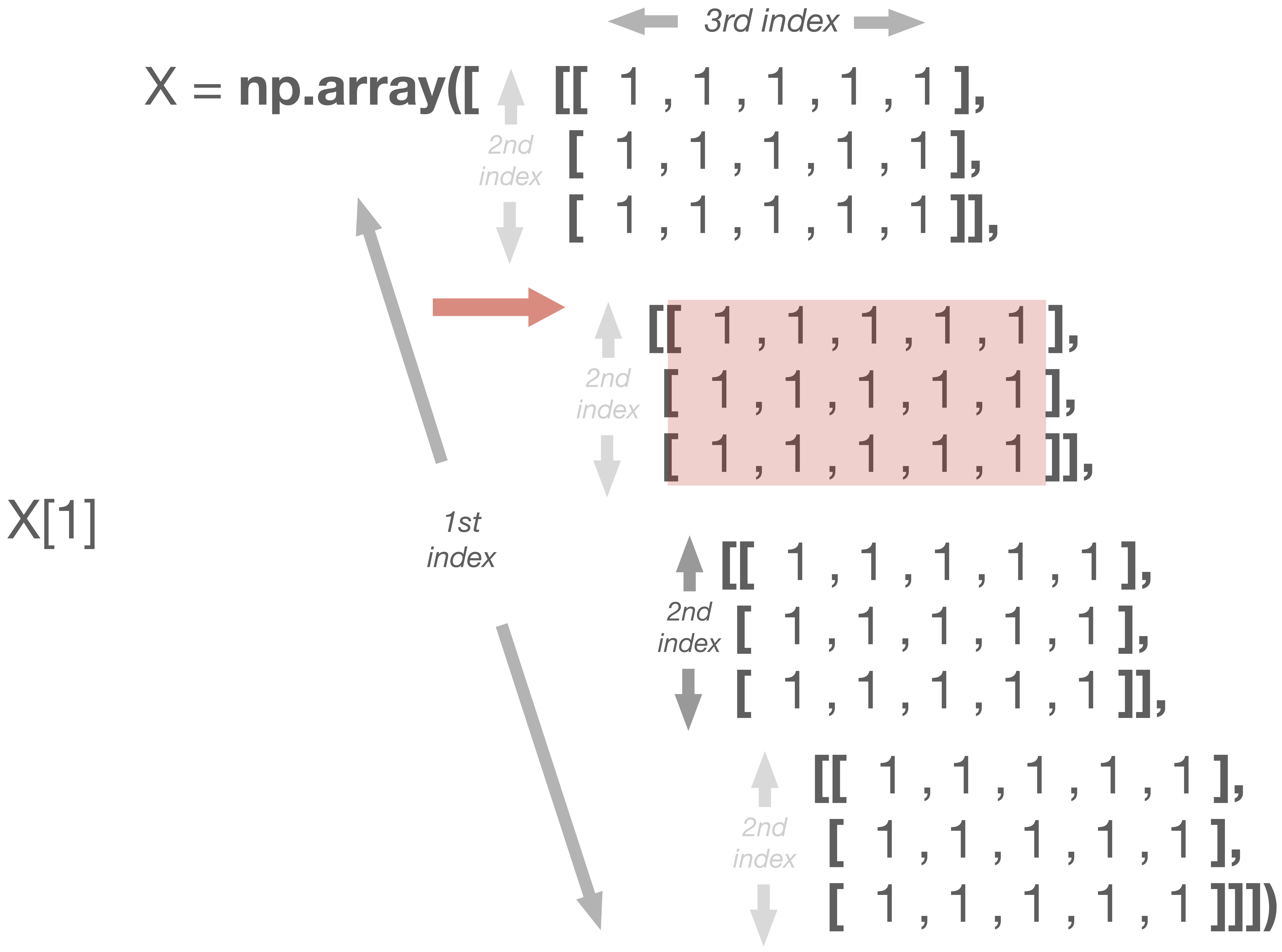
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

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

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

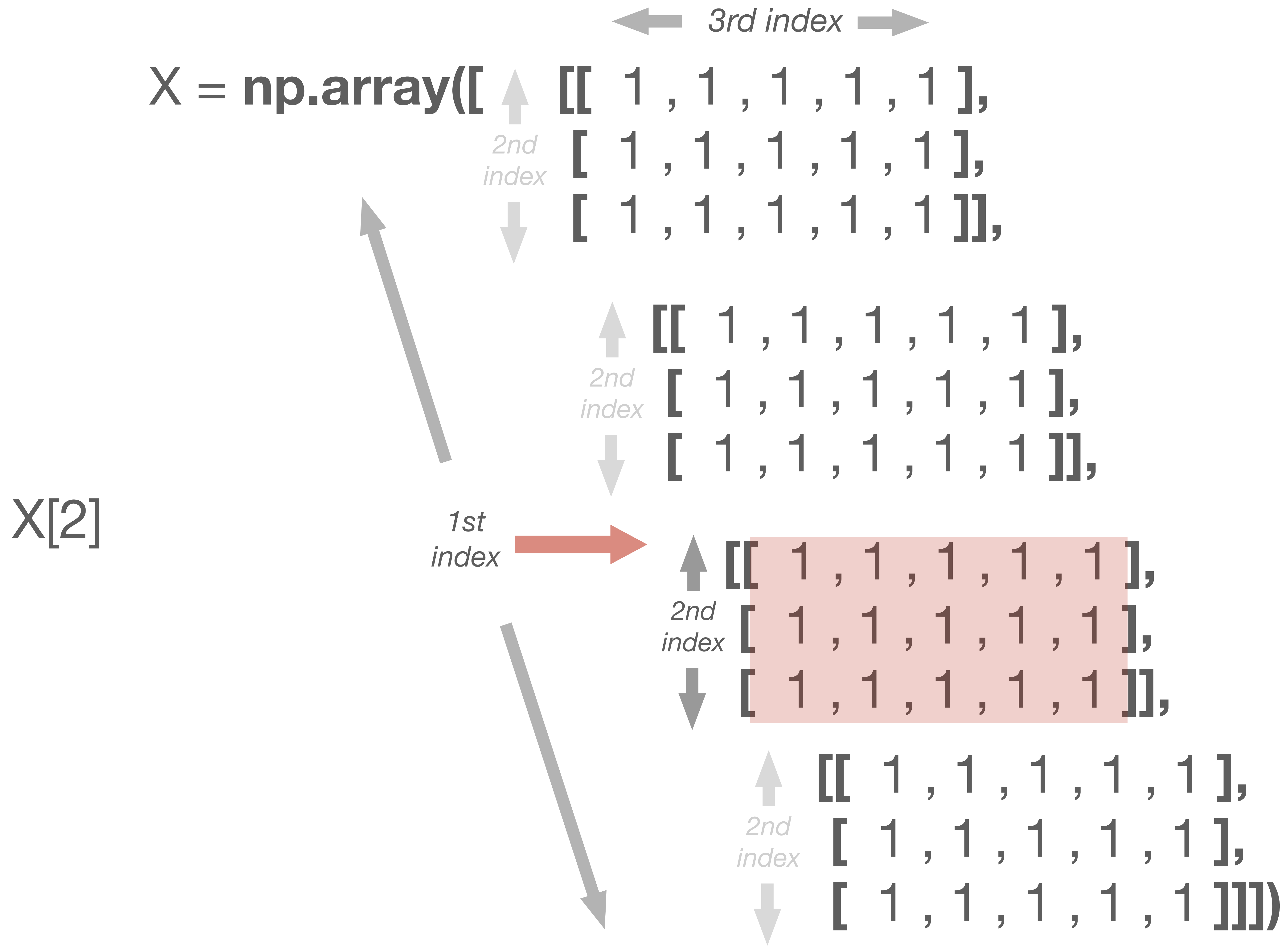
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ]],
                   [[ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ]],
                     [[ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array: A = **np.array([[**1, 2, 3**]**, **[**3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

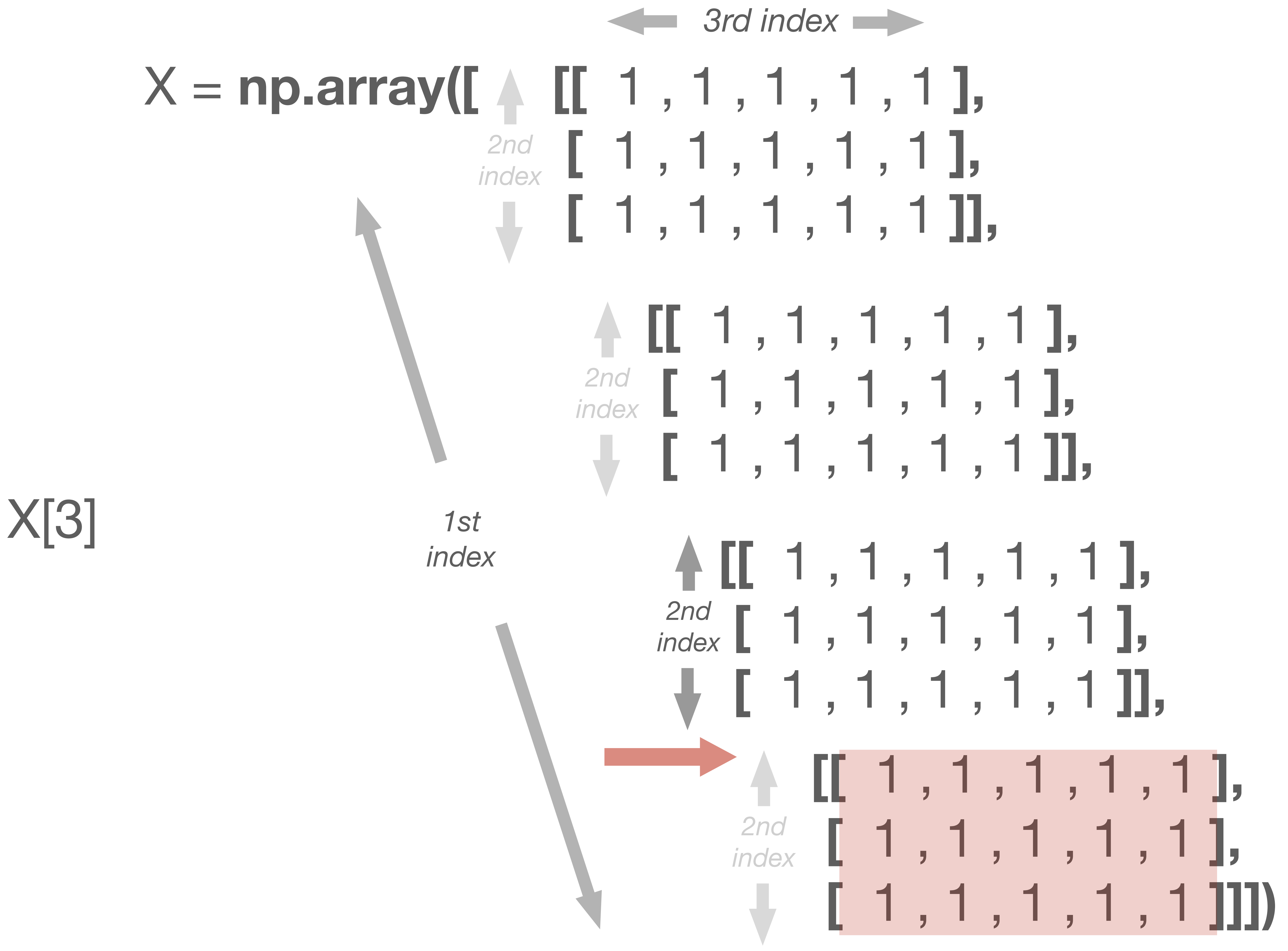
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

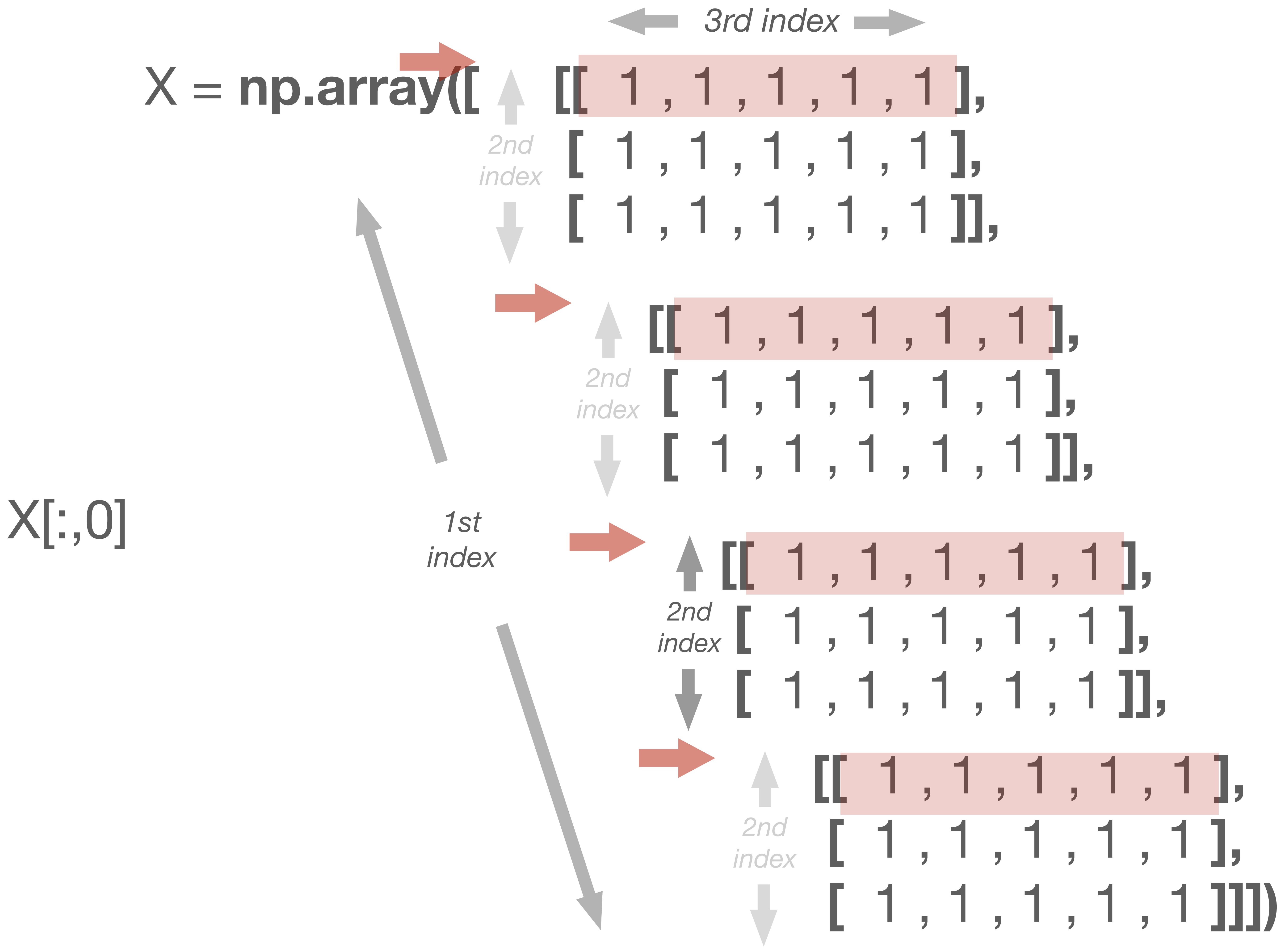
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



X[:,0]

```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

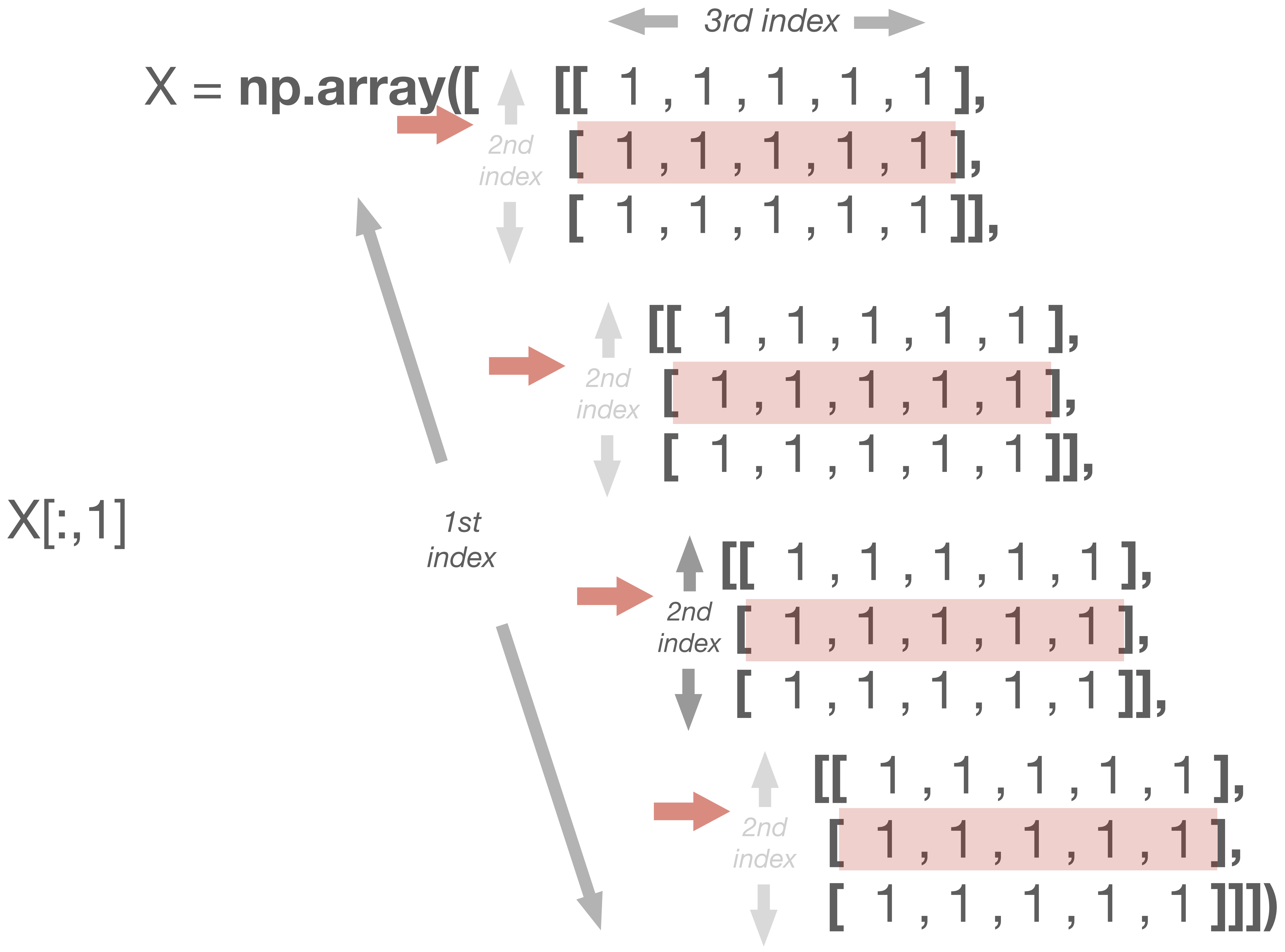
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



X[:,1]

```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array: A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing   **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind]   - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2]   - returns [0,0],[2,3], and [3,2] elements

boolean indexing

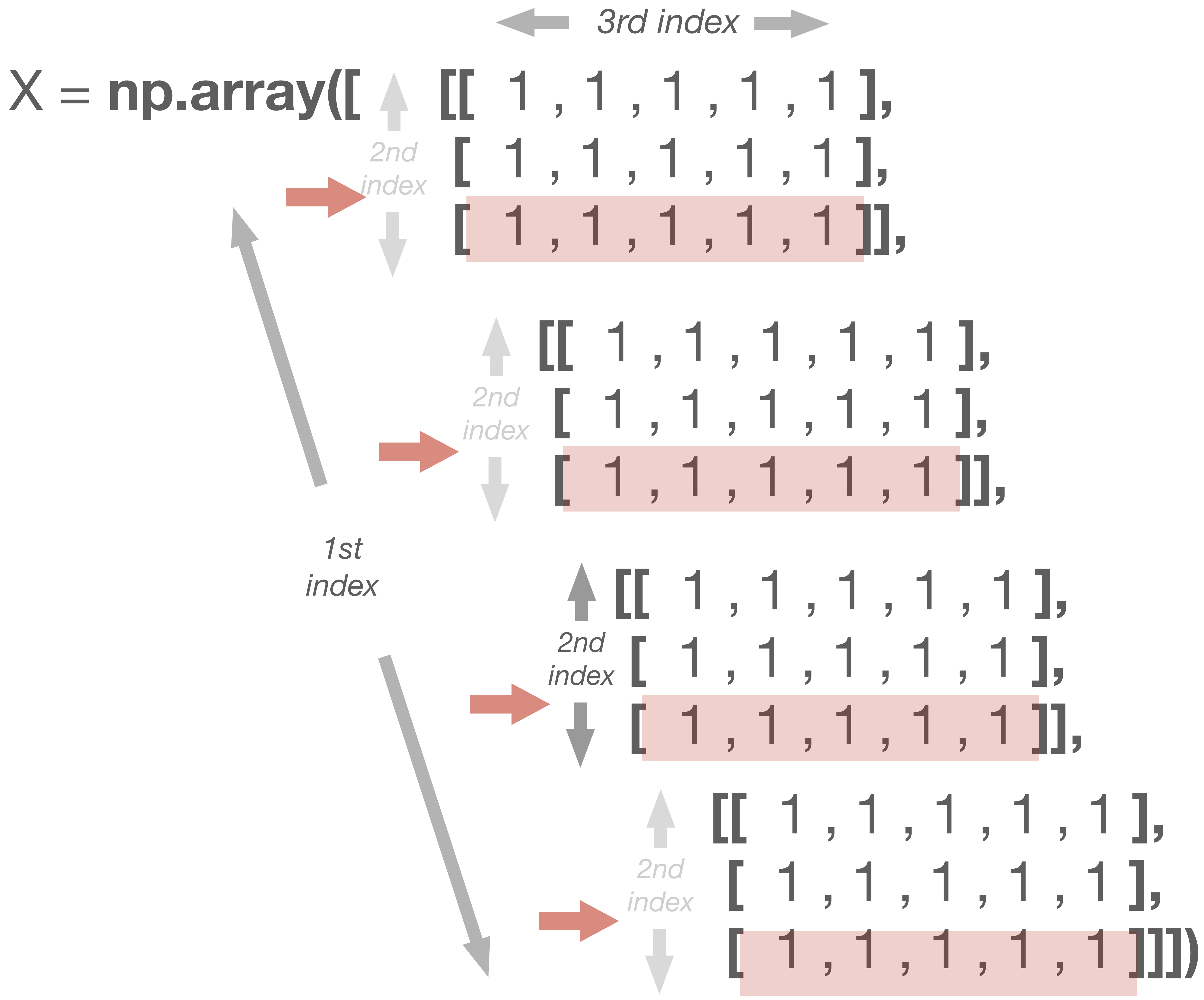
bool = **[True, True, False, True];**   MUST BE ARRAY LENGTH
x[bool]   - returns 0,1, and 3 element.

X[bool,bool]   - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block

X[:,2]



```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array: A = np.array([[1, 2, 3],
                                      [3, 2, 1]])

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing   **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

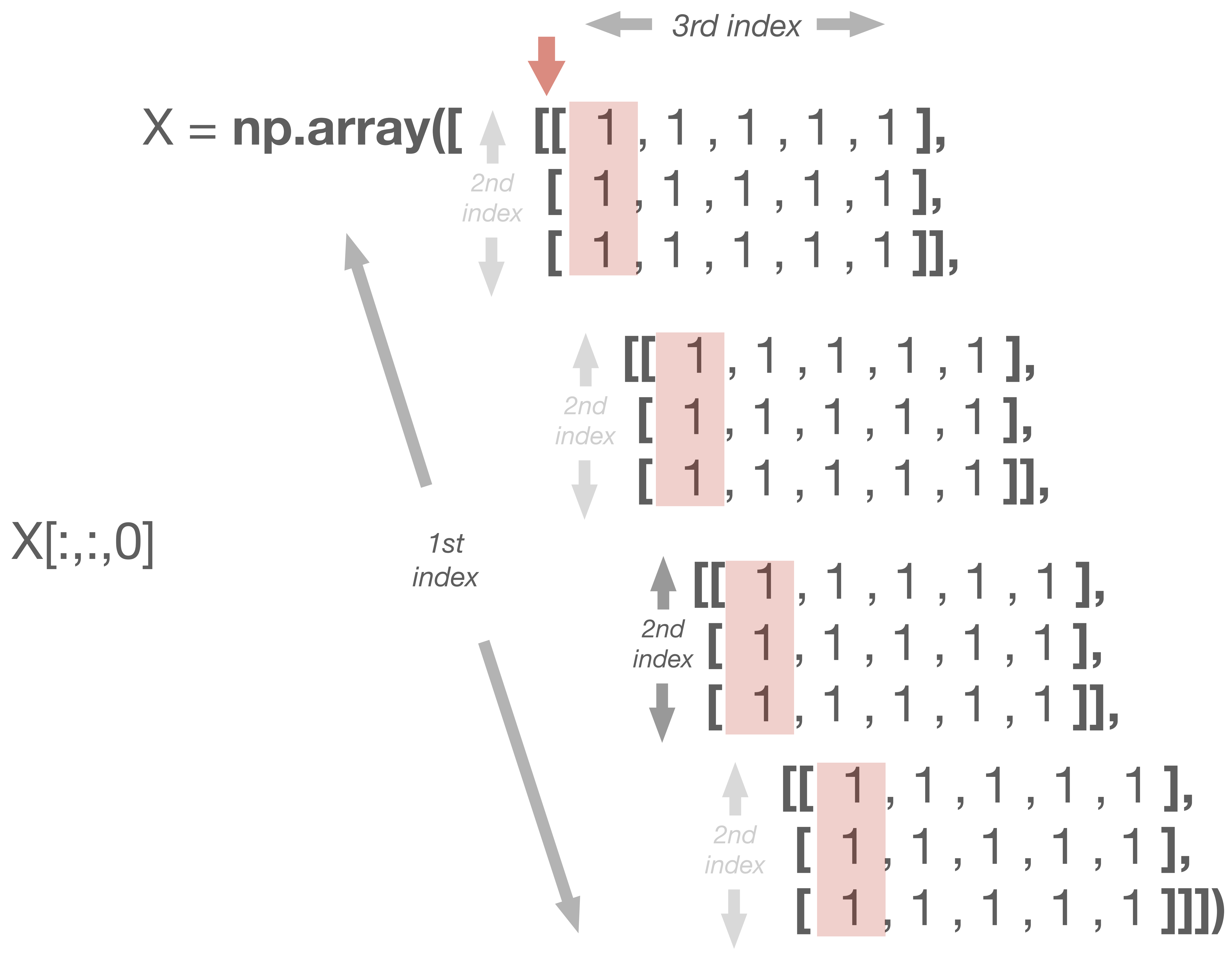
ind = [0, 2, 3];
x[ind]   - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2]   - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];**   MUST BE ARRAY LENGTH
x[bool]   - returns 0,1, and 3 element.
X[bool,bool]   - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



X[:,:,0]

```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ]],
                   [[ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ]],
                     [[ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

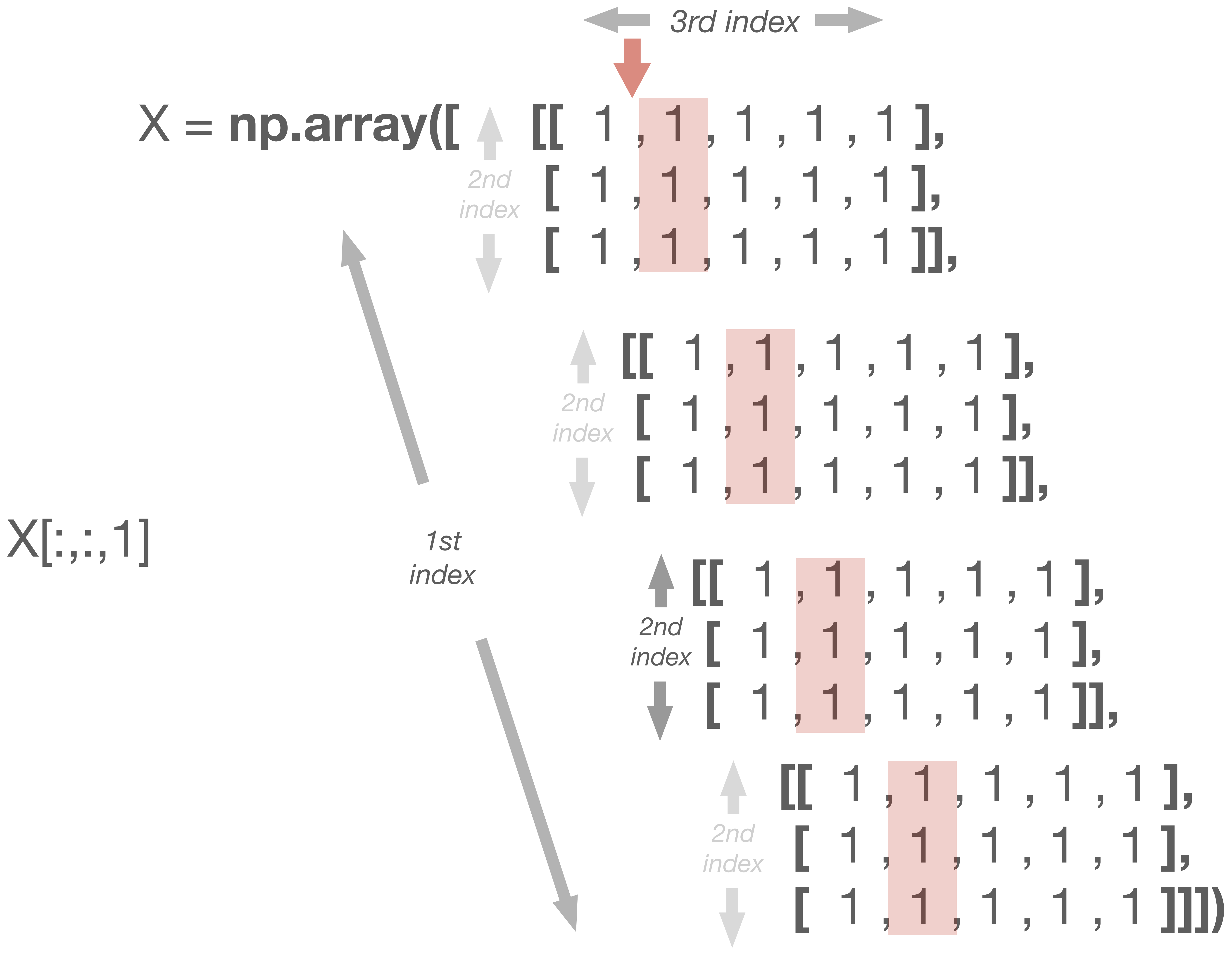
ind = [0, 2, 3];
x[ind]   - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2]   - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];**   MUST BE ARRAY LENGTH
x[bool]   - returns 0,1, and 3 element.
X[bool,bool]   - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ]],
                   [[ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ]],
                     [[ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

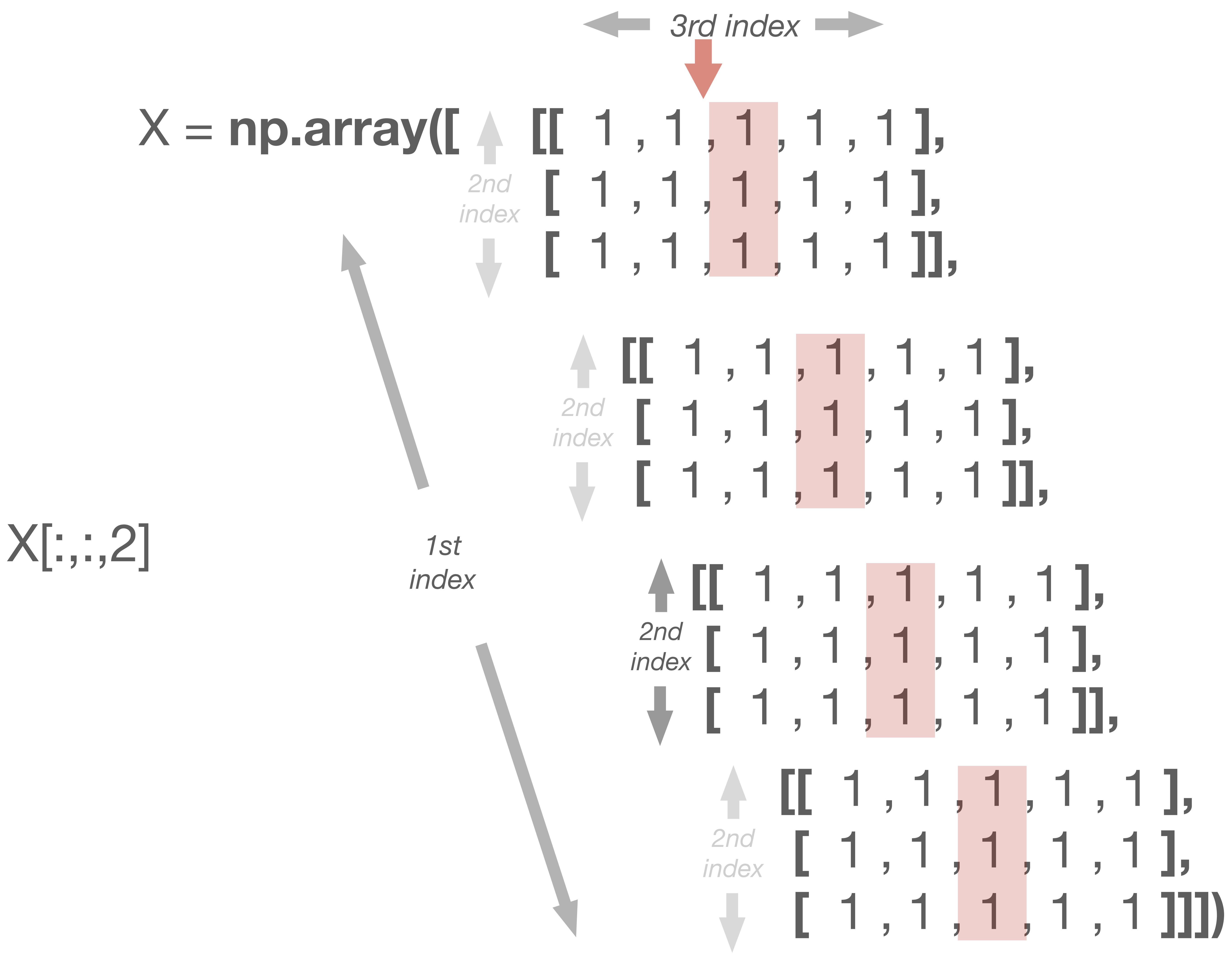
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



X[:,:,2]

```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

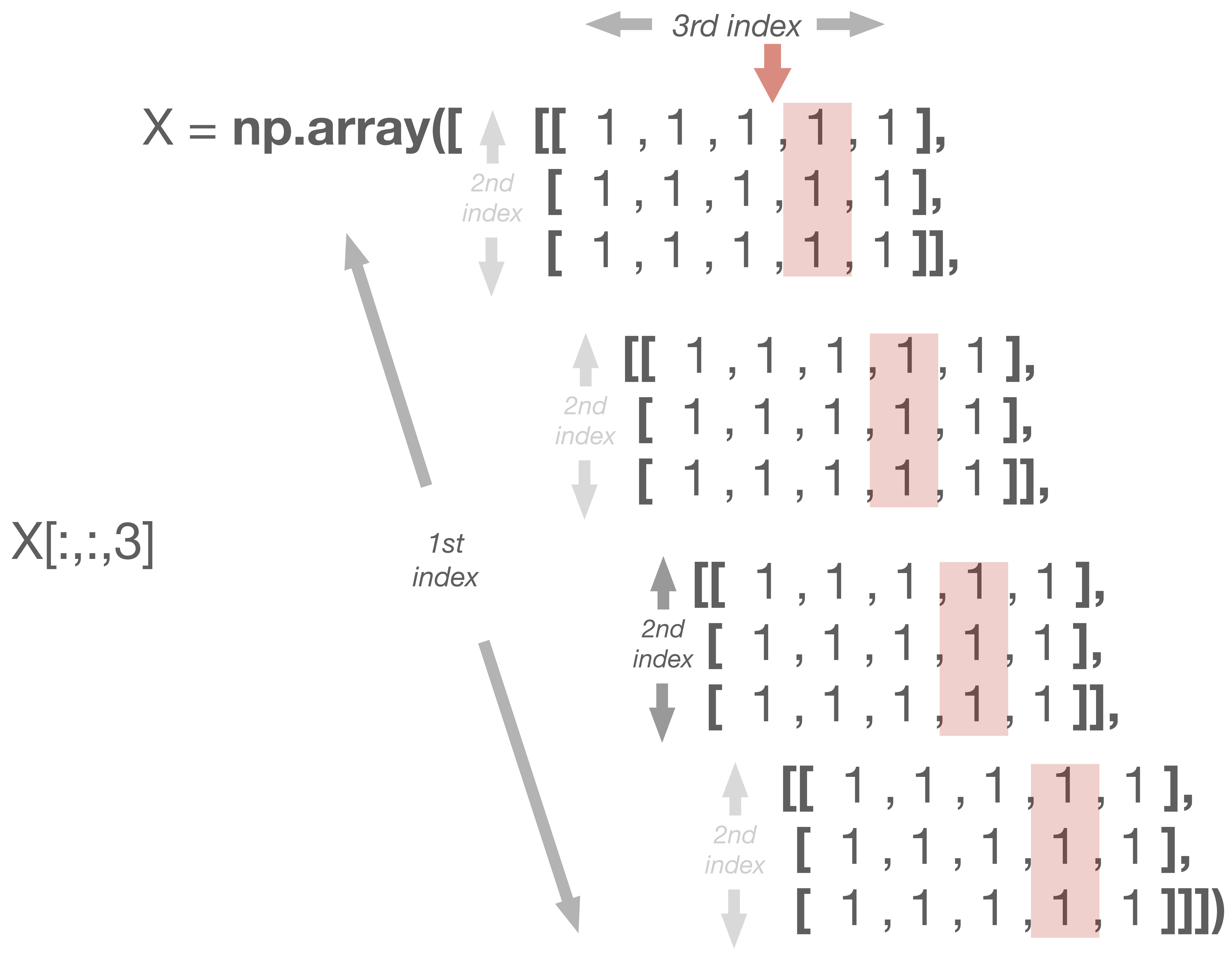
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



X[:,:,3]

```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ]],
                   [[ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ]],
                     [[ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array: A = **np.array(** **[[**1, 2, 3**]**,
[3, 2, 1**]]** **)**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

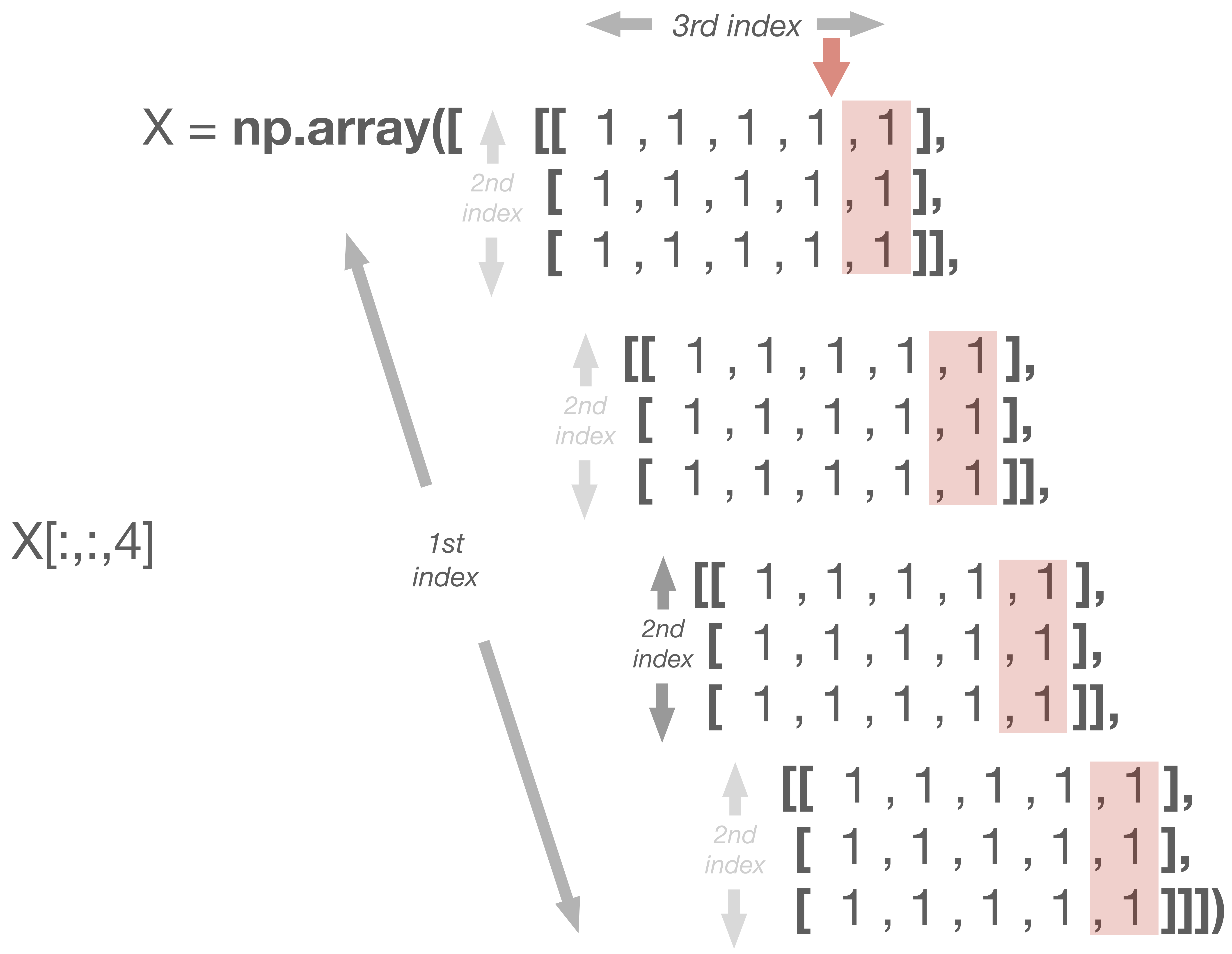
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



X[:,:,4]

```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ]],
                   [[ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ]],
                     [[ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array: A = **np.array(** **[[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

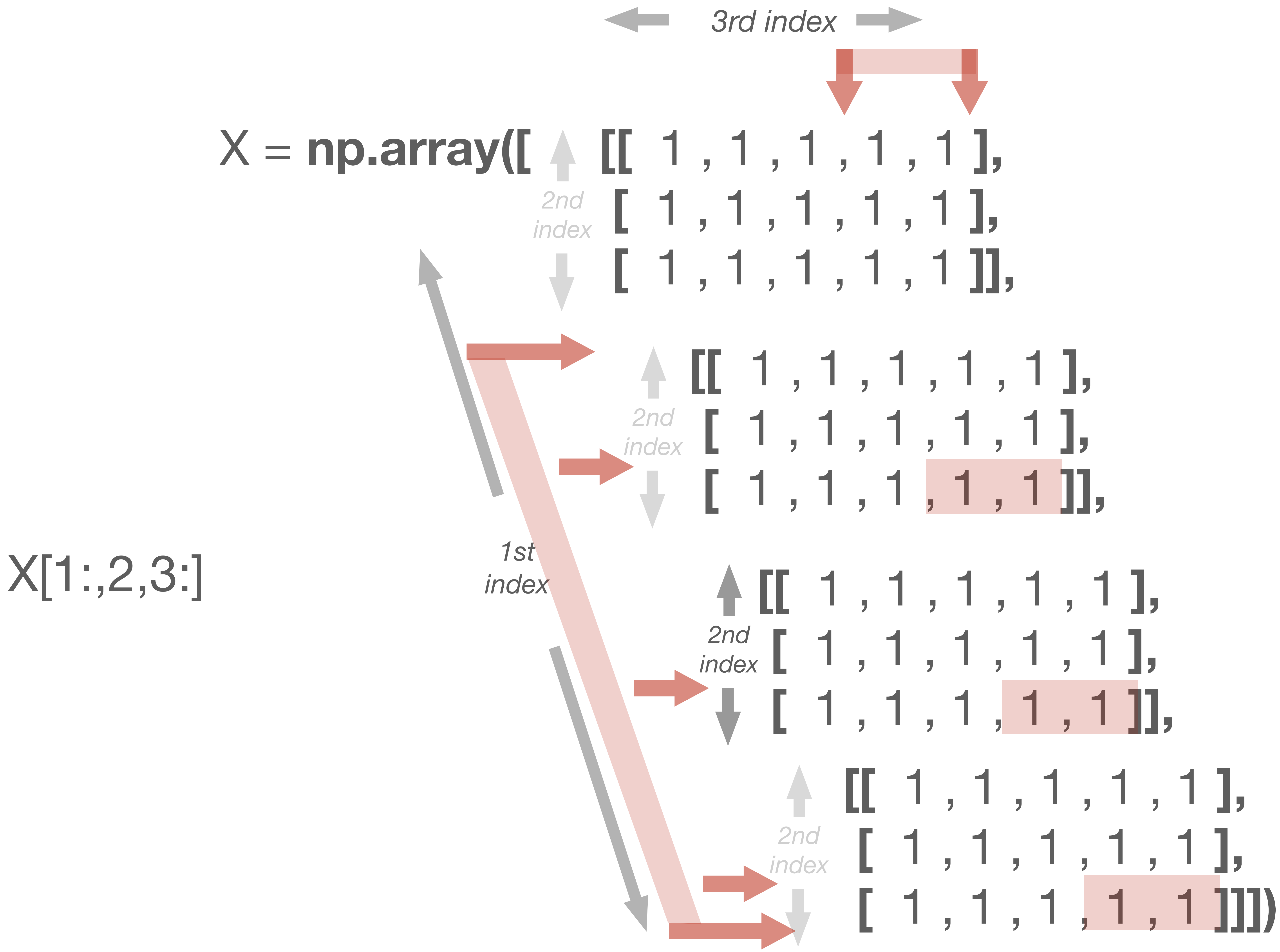
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



X[1:,2,3:]

```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

A = **np.array(** **[[**1, 2, 3**]**,
[3, 2, 1**]]** **)**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

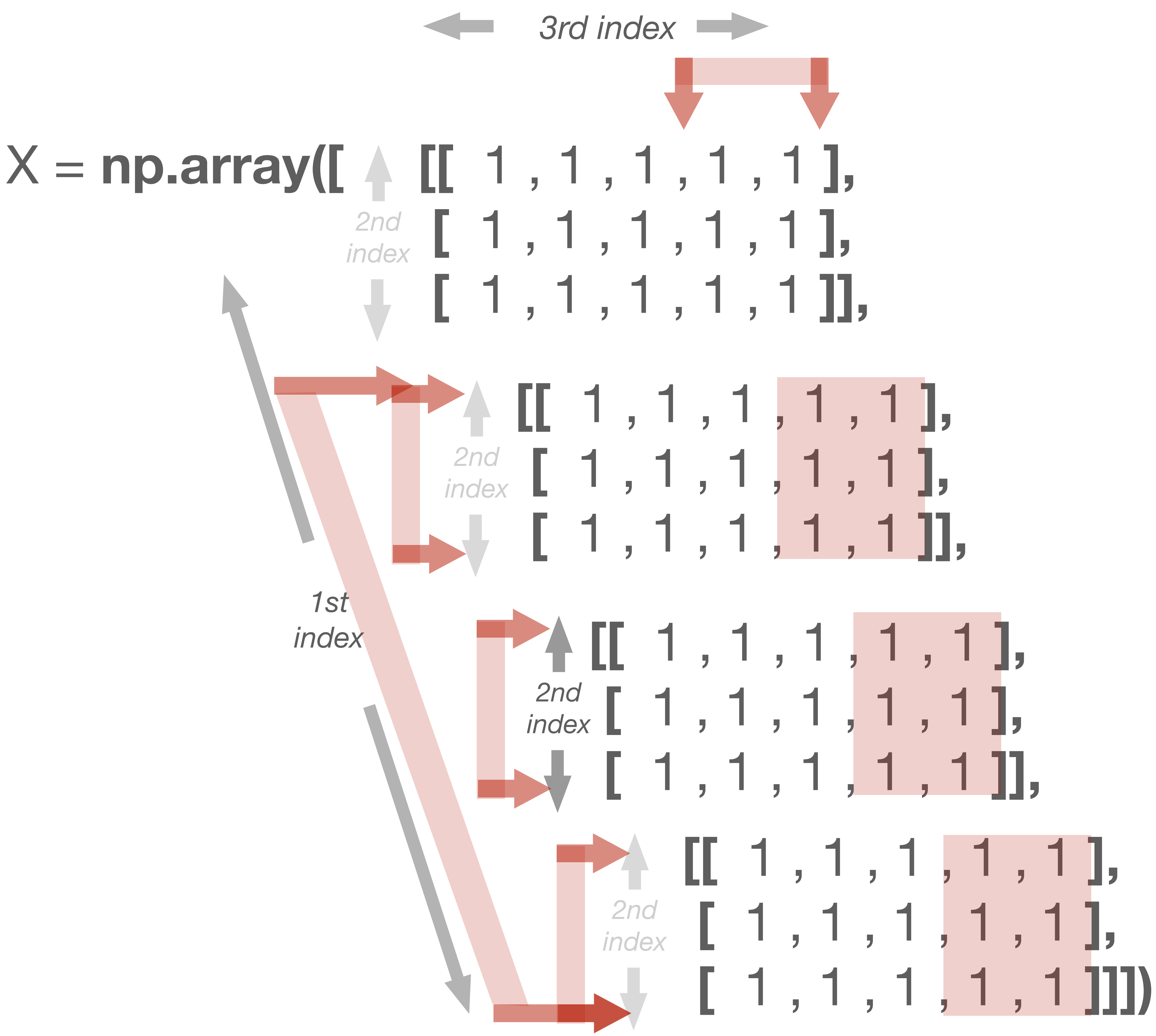
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block

X[1:,:,3:]



```
X = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]]])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing   start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

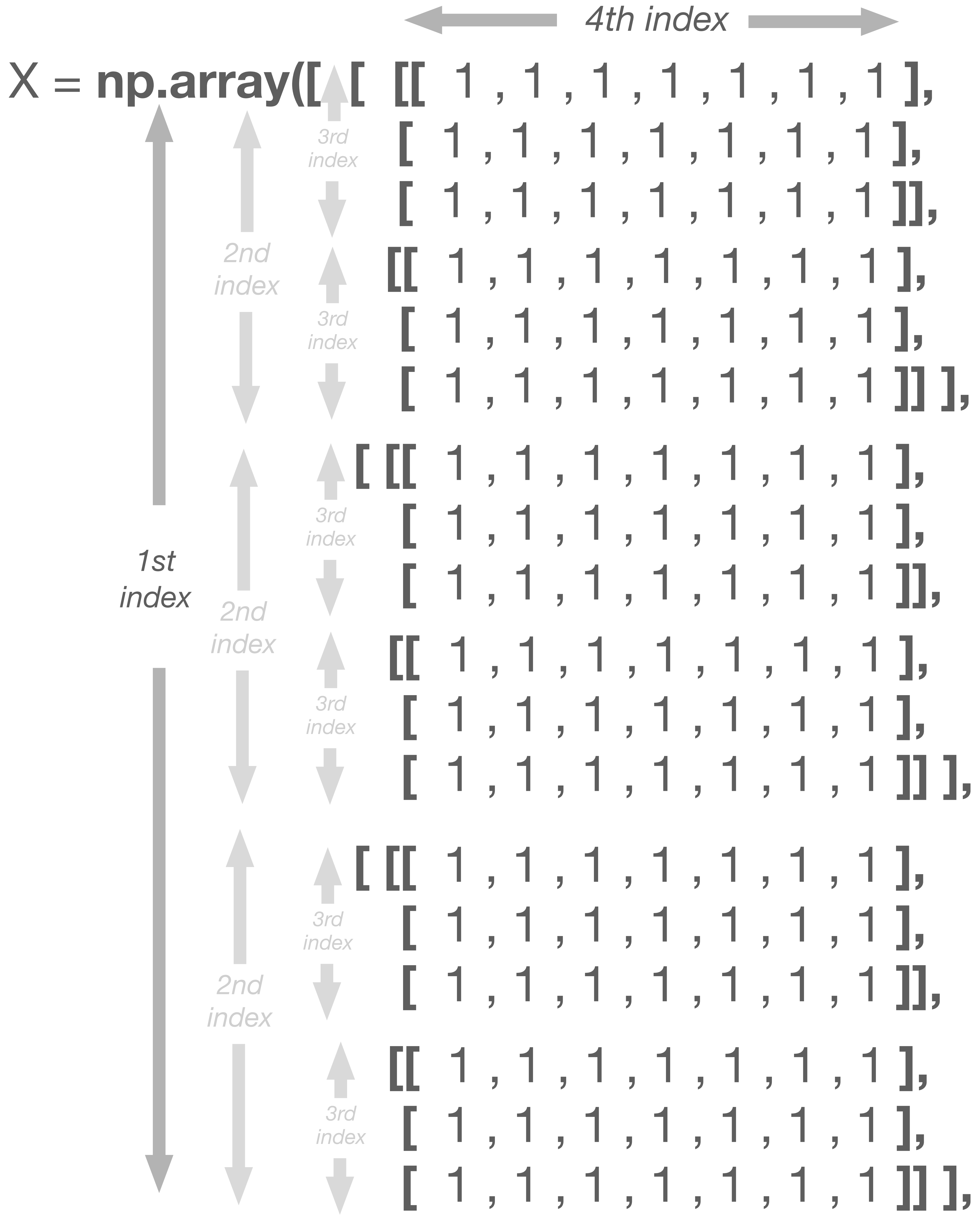
ind = [0, 2, 3];
x[ind]   - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2]   - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];**   MUST BE ARRAY LENGTH
x[bool]   - returns 0,1, and 3 element.
X[bool,bool]   - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [  [[  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [  1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

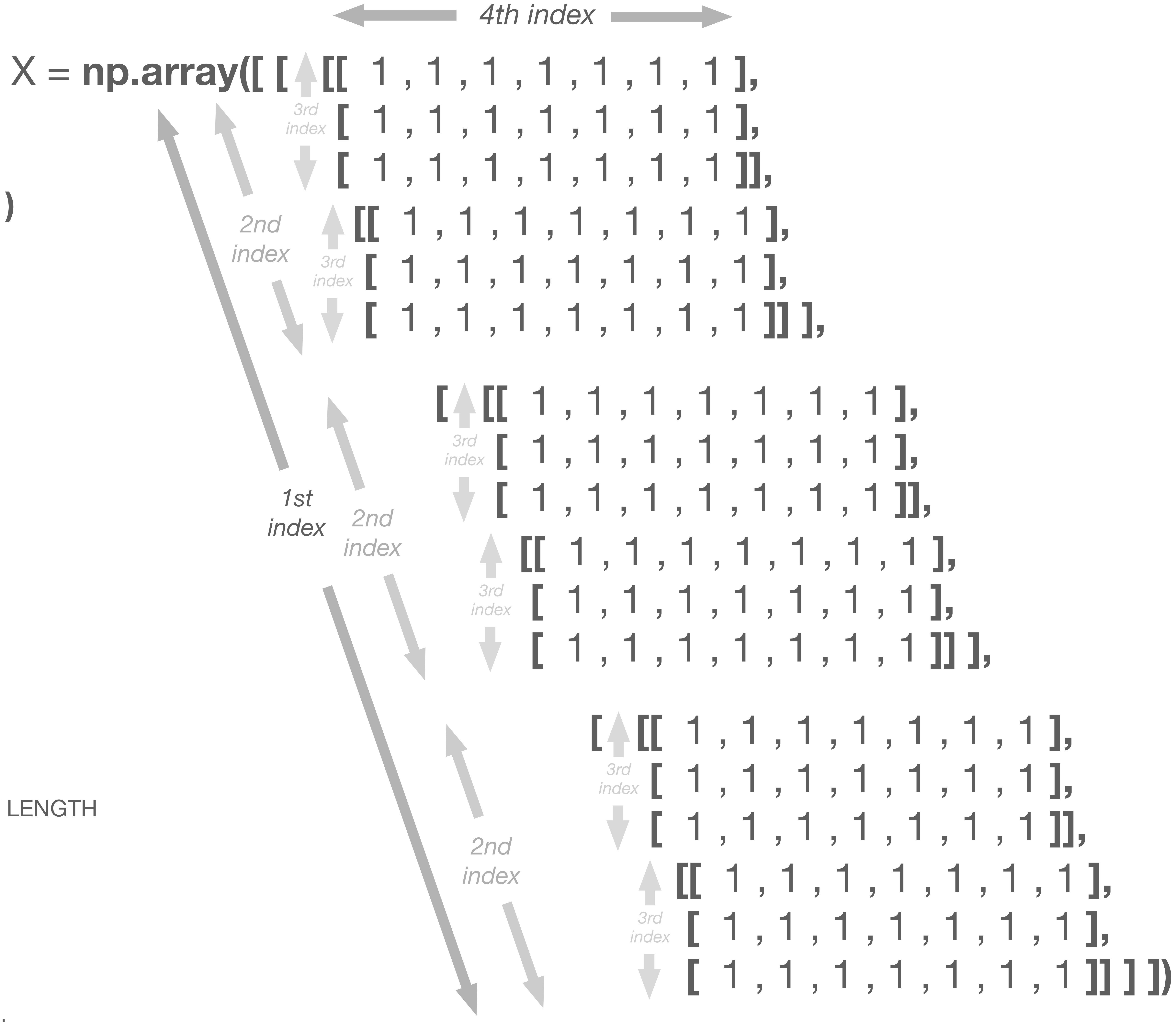
bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block

X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

X[0]

array indexing

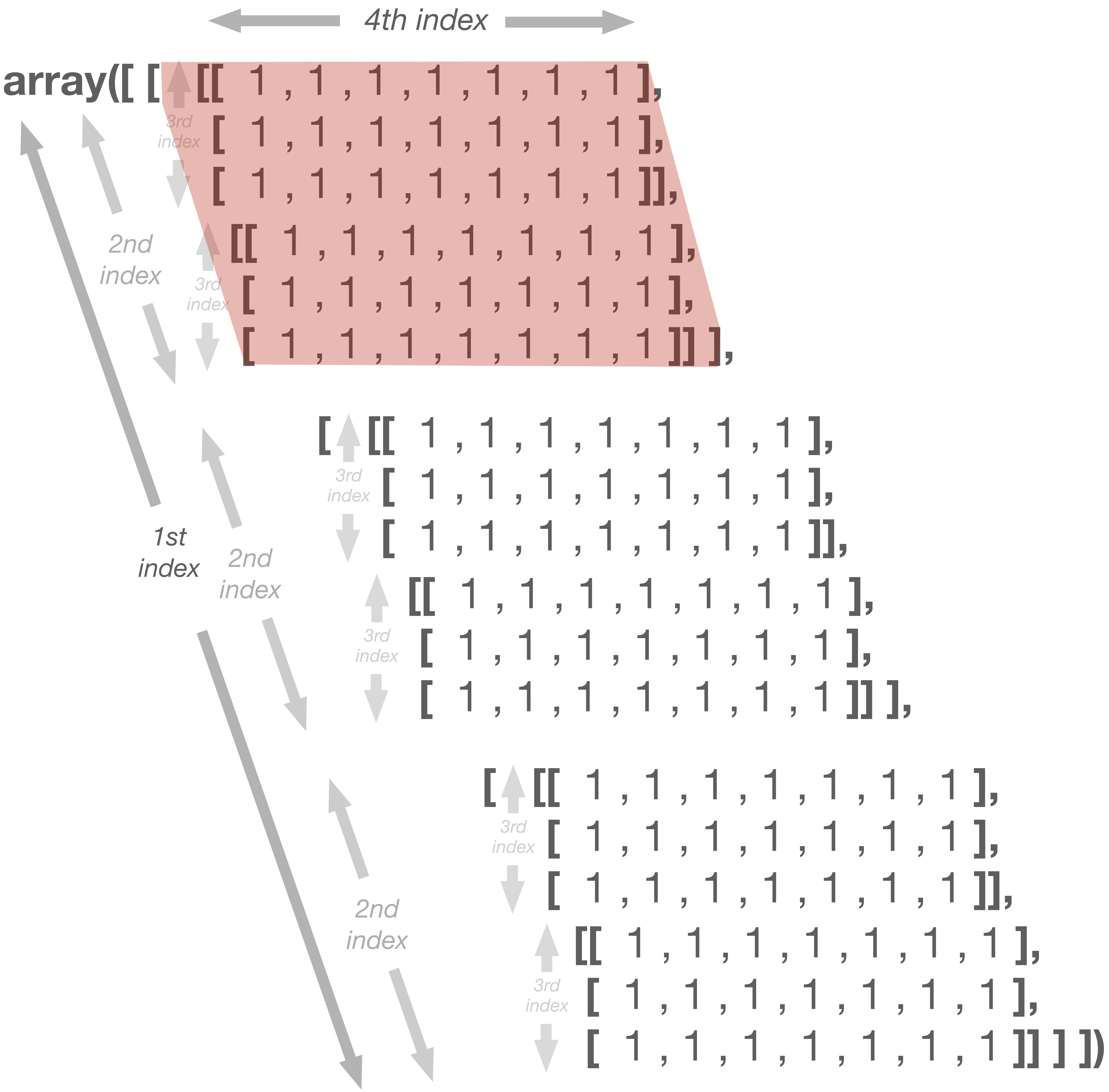
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array(** [[1, 2, 3],
[3, 2, 1]])

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

X[1]

array indexing

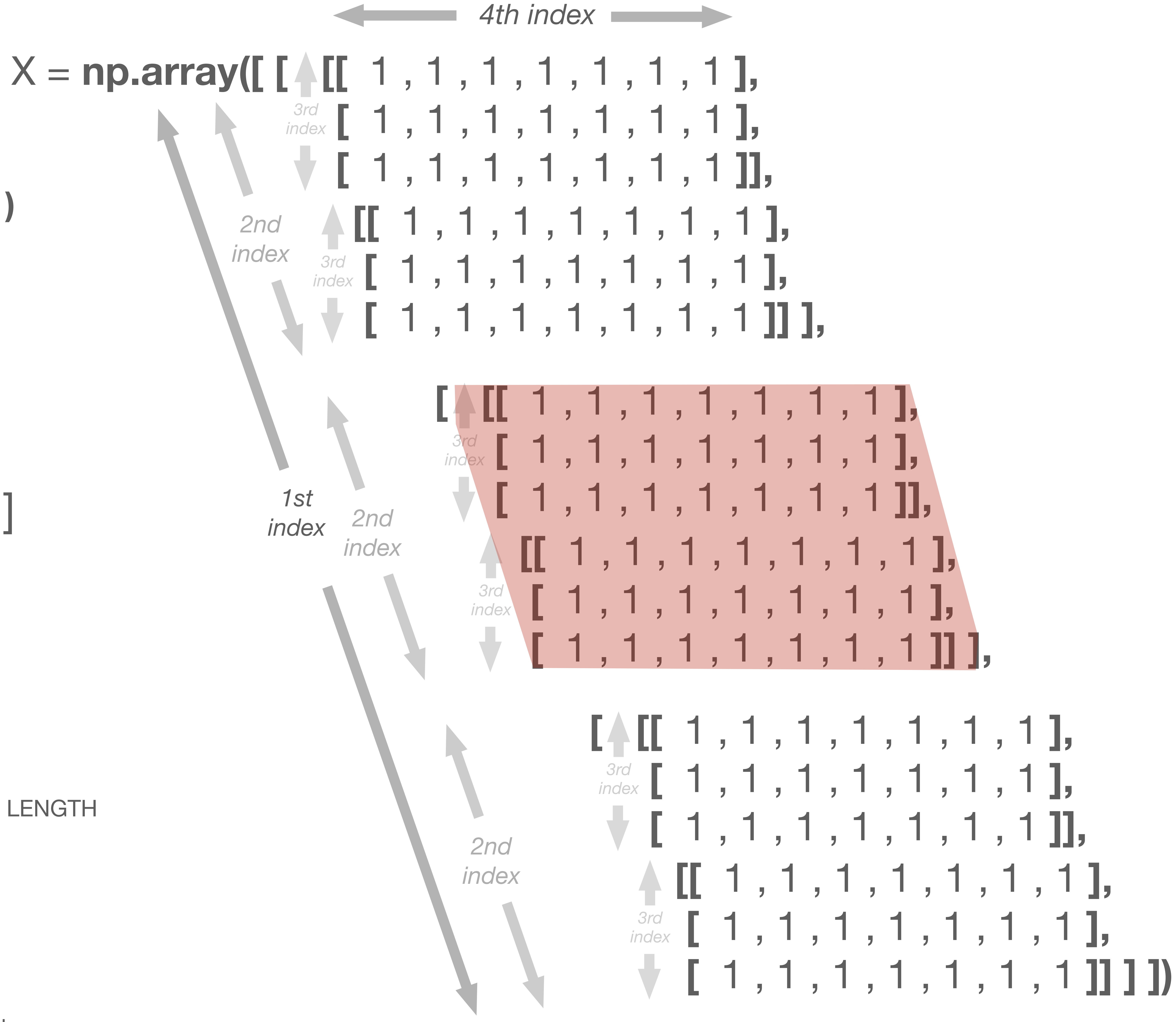
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

X[2]

array indexing

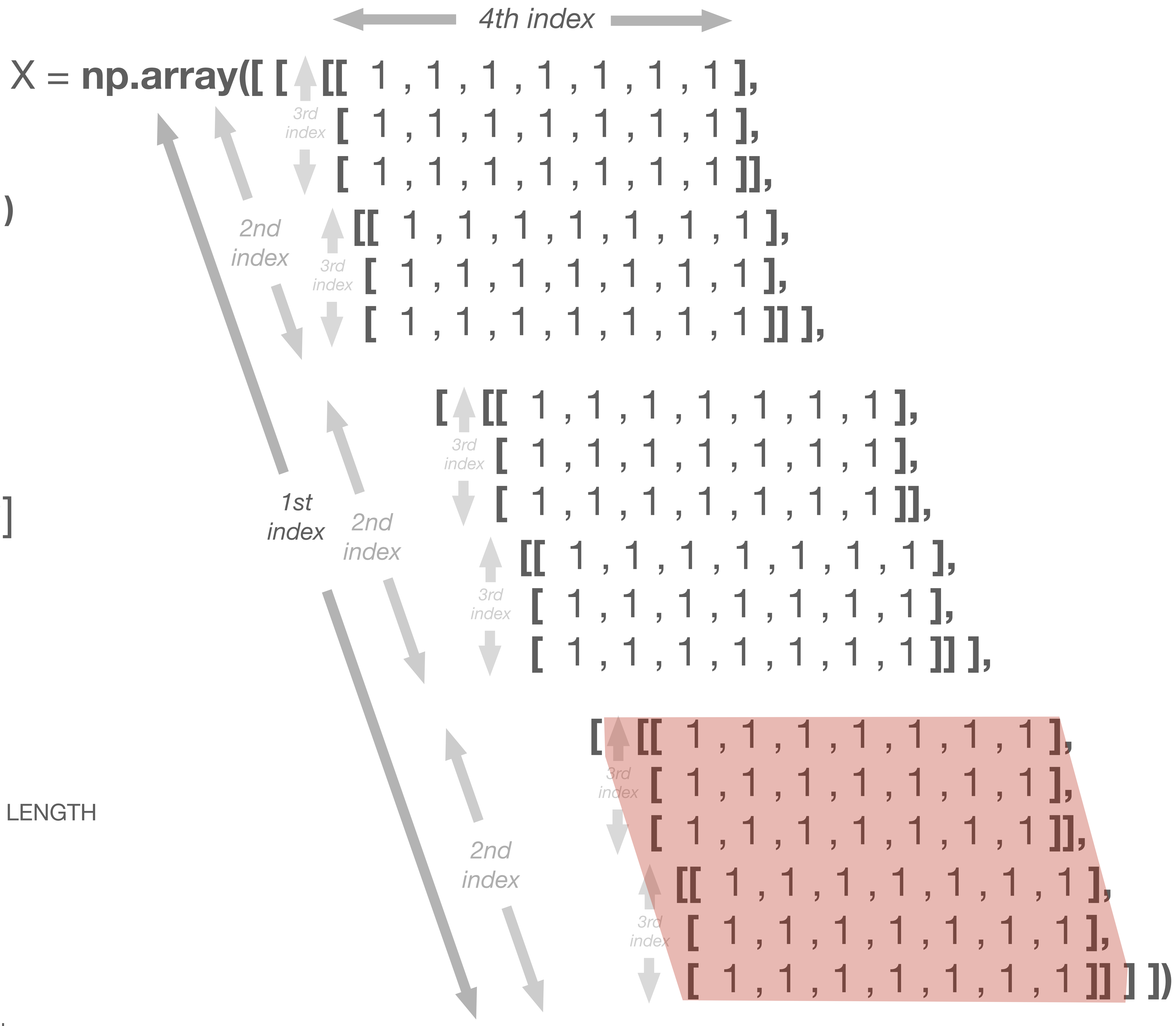
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,0]

array indexing

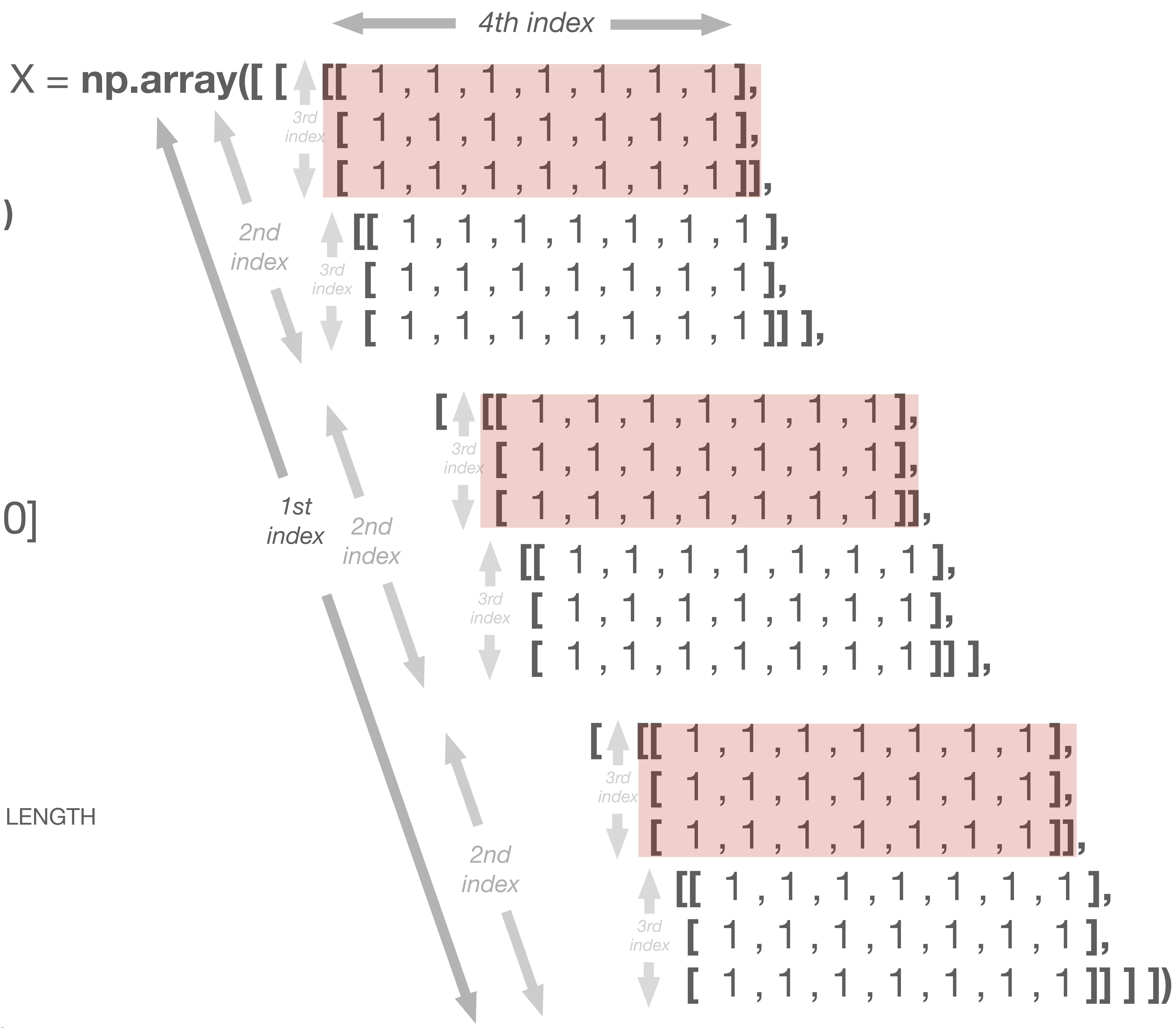
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array: A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,1]

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

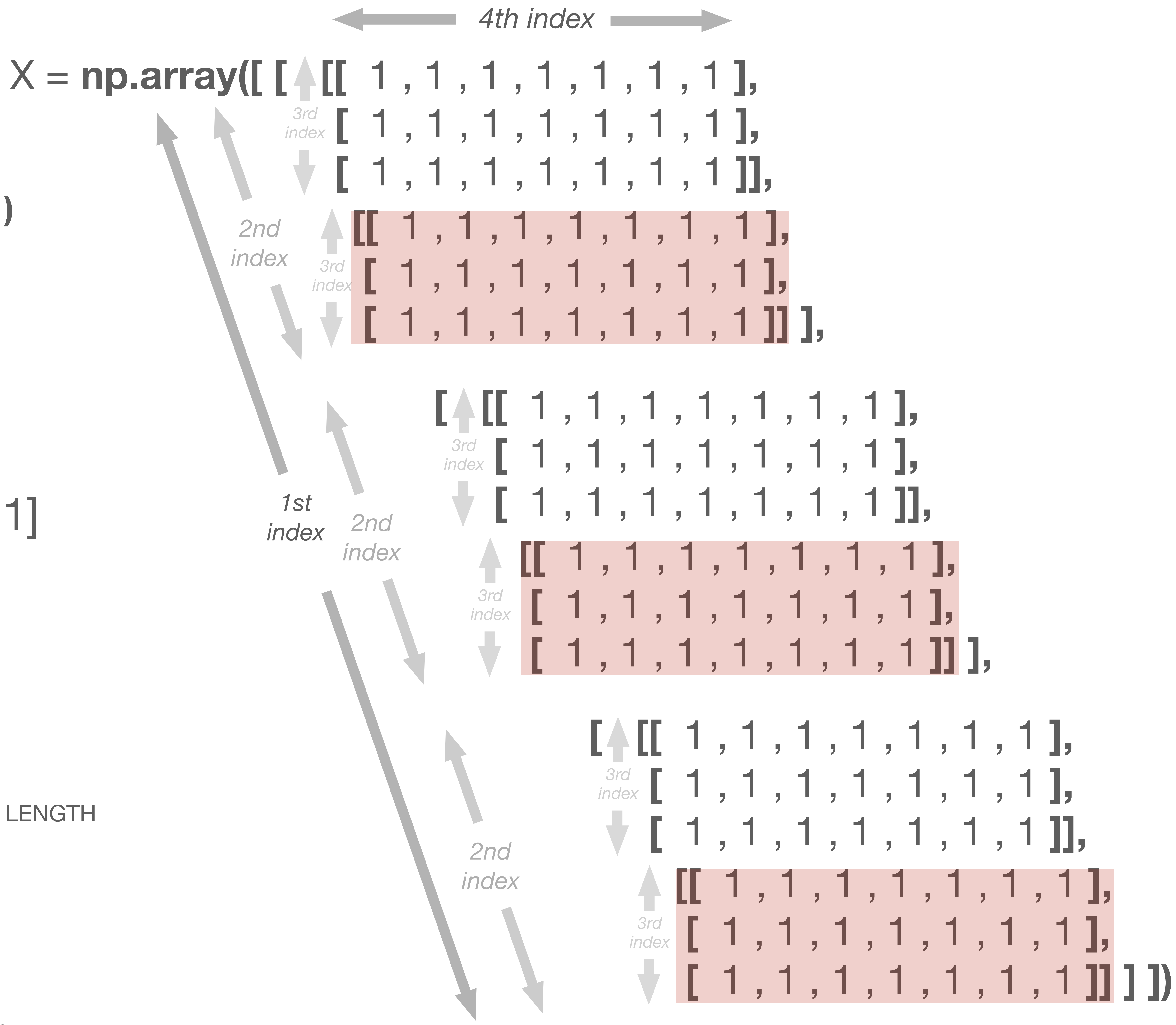
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,:,0]

array indexing

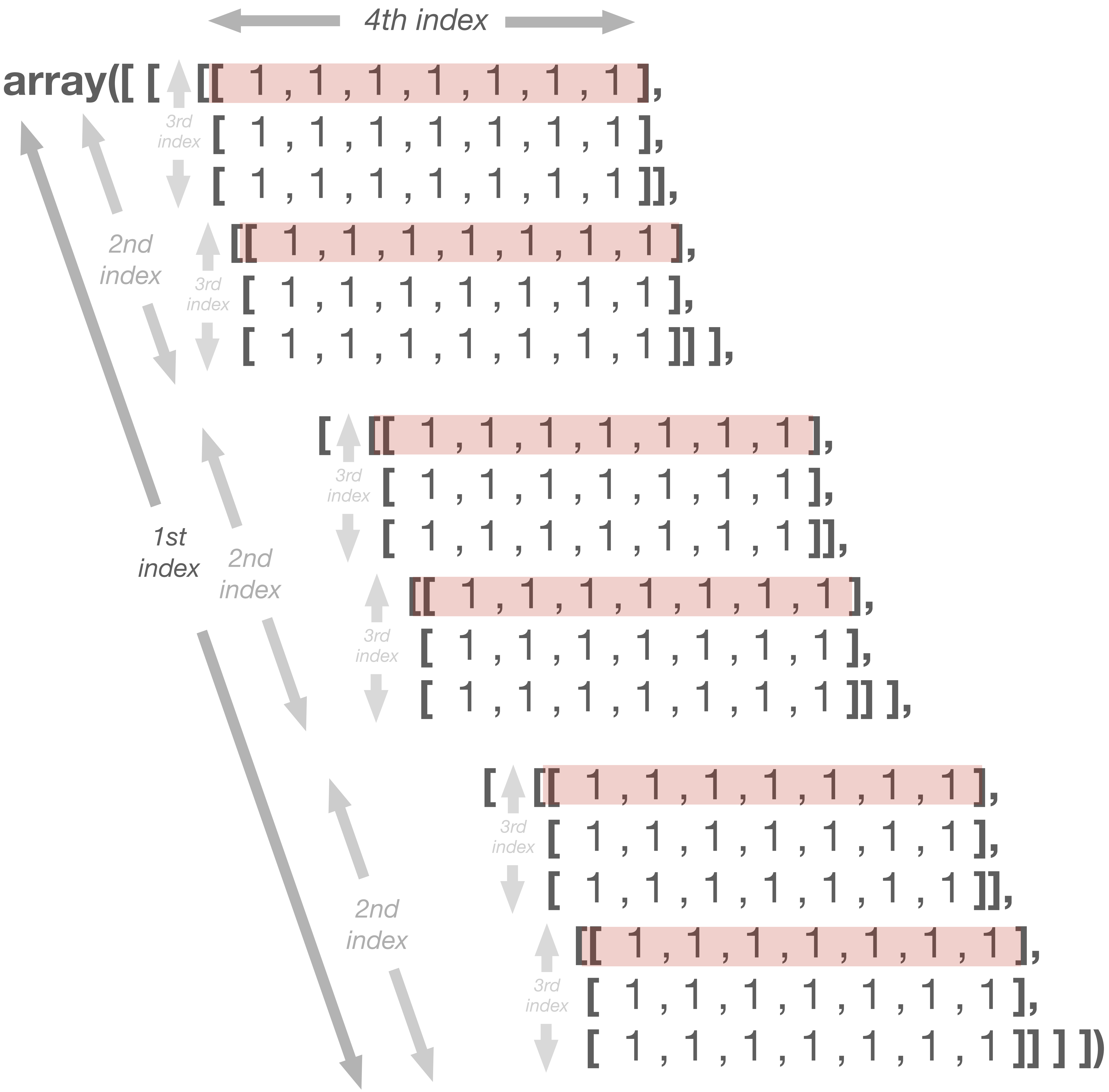
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
                 [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                    [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                     [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                     [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
                   [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                      [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                       [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                       [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,:,1]

array indexing

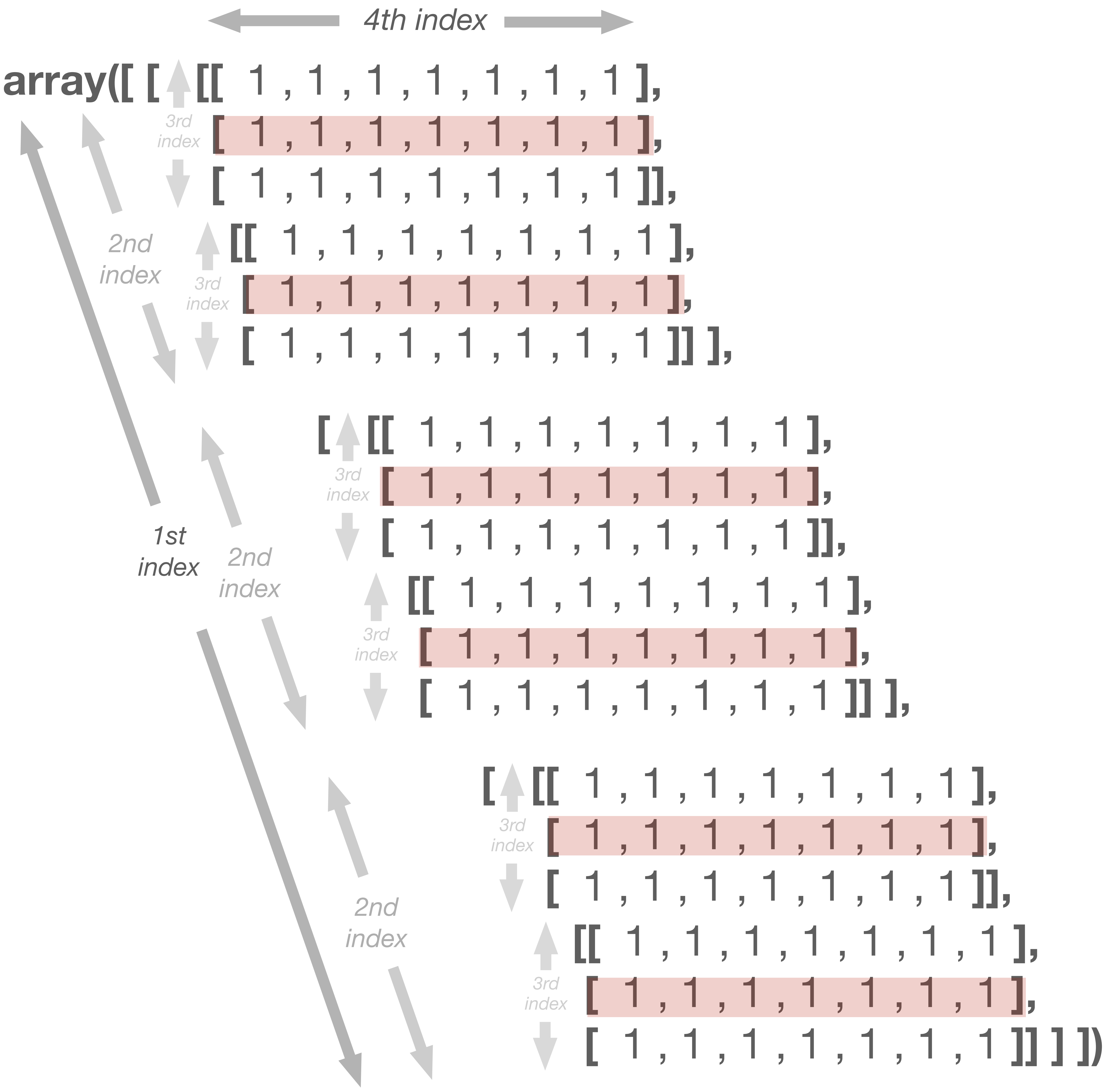
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,:,2]

array indexing

ind = [0, 2, 3];
x[ind]   - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2]   - returns [0,0],[2,3], and [3,2] elements

boolean indexing

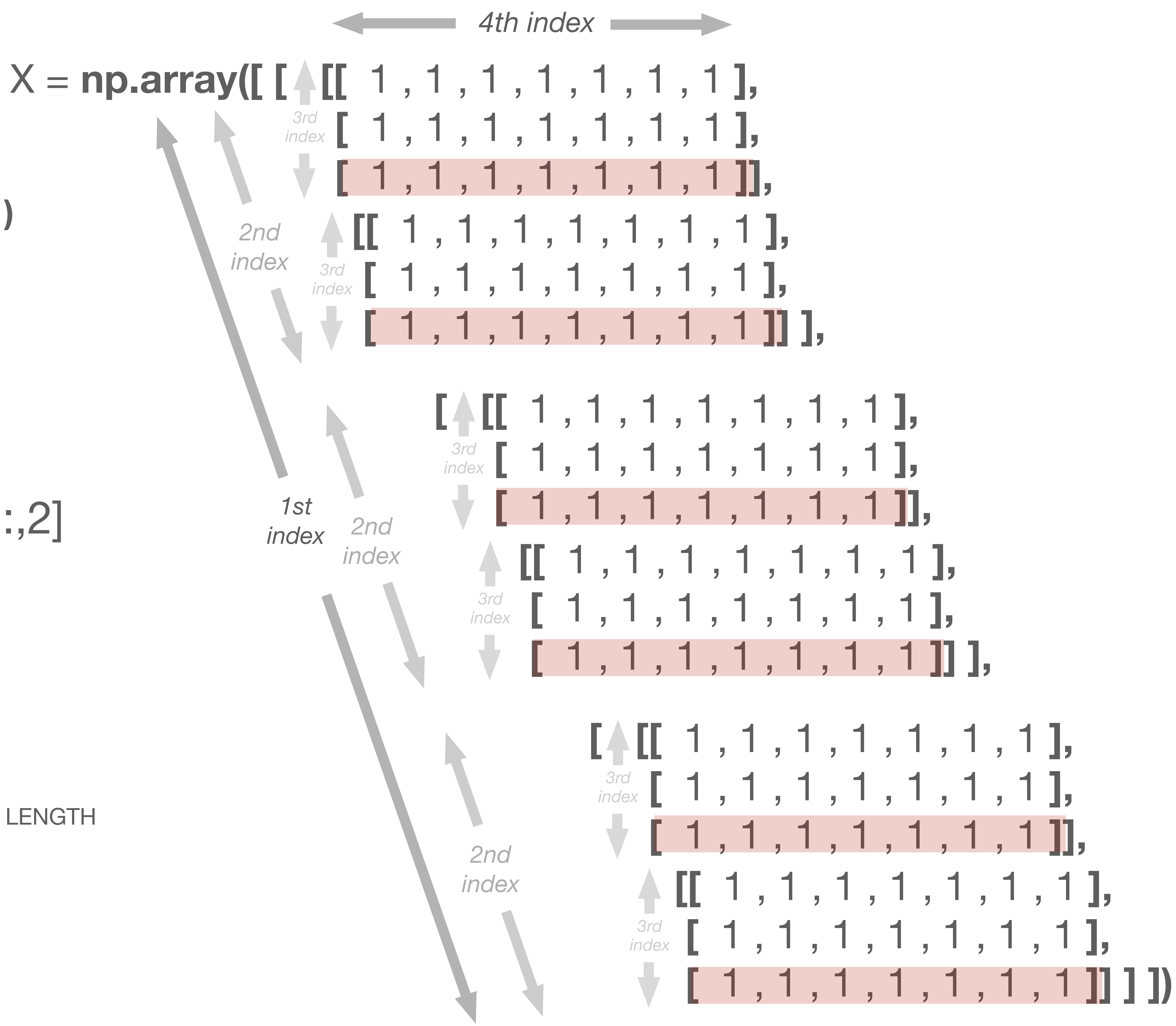
bool = **[True, True, False, True];**   MUST BE ARRAY LENGTH
x[bool]   - returns 0,1, and 3 element.

X[bool,bool]   - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block

X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array(** [[1, 2, 3],
[3, 2, 1]])

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,:,:,0]

array indexing

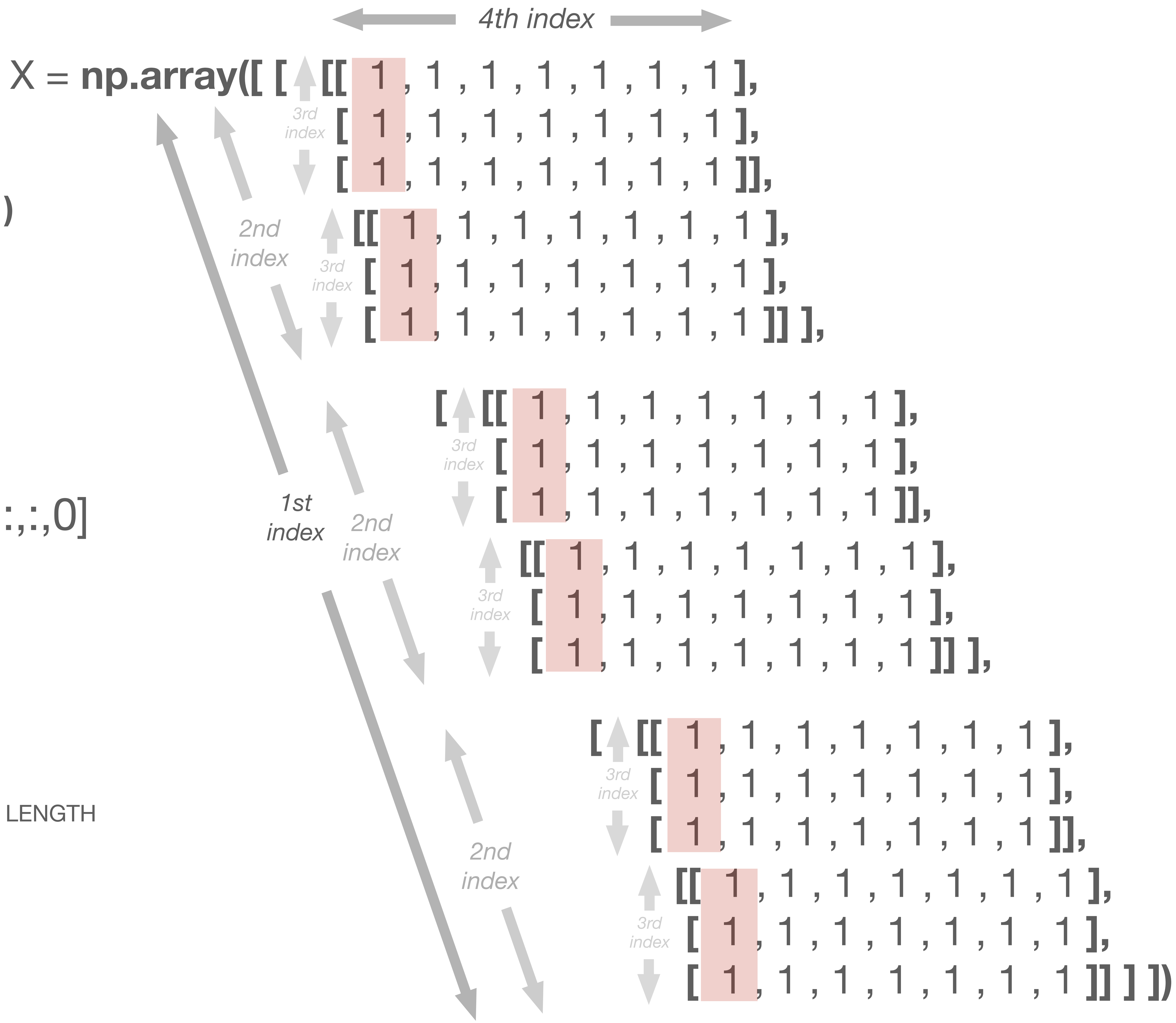
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1, 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,:,:,1]

array indexing

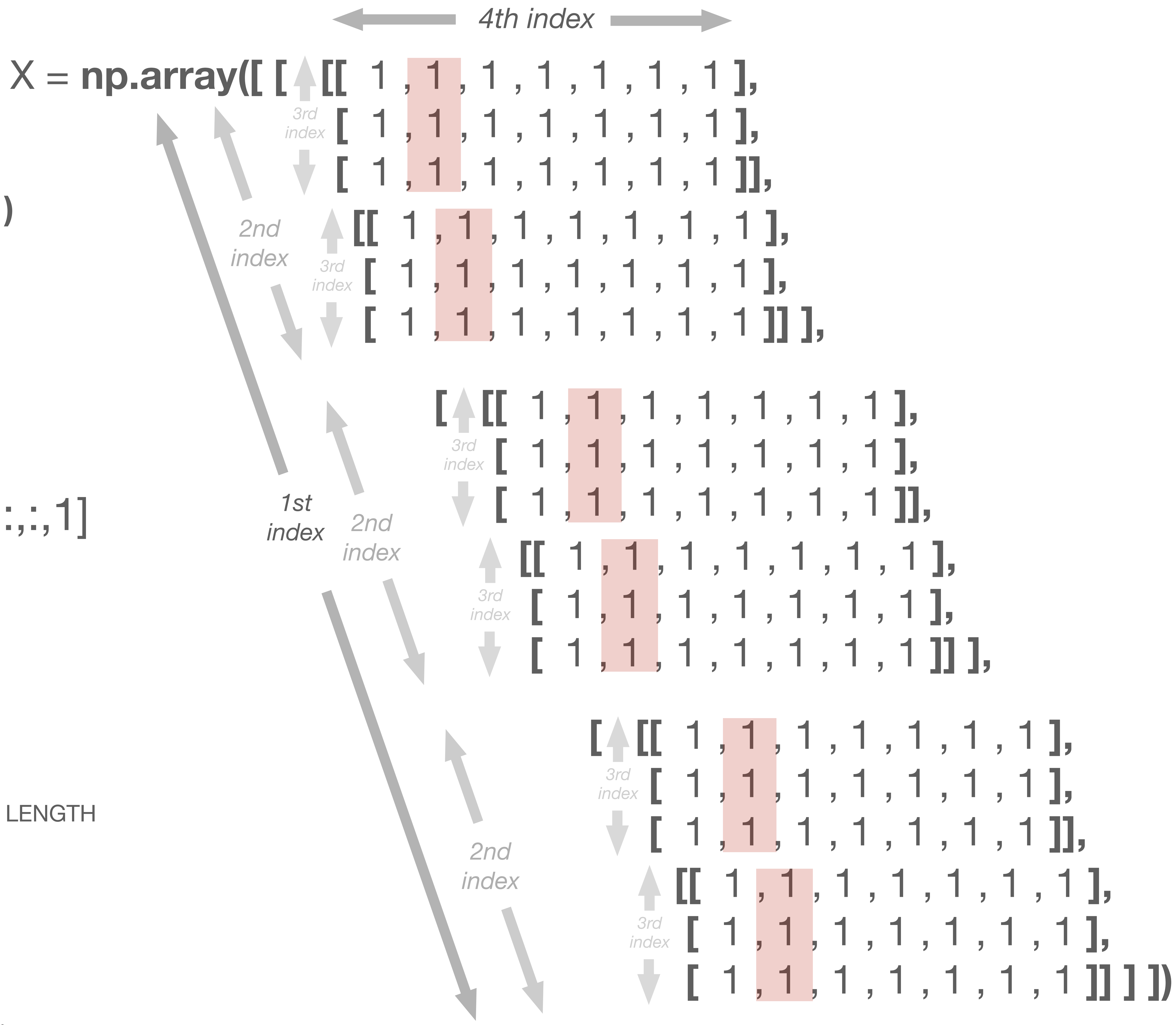
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,:,:,2]

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

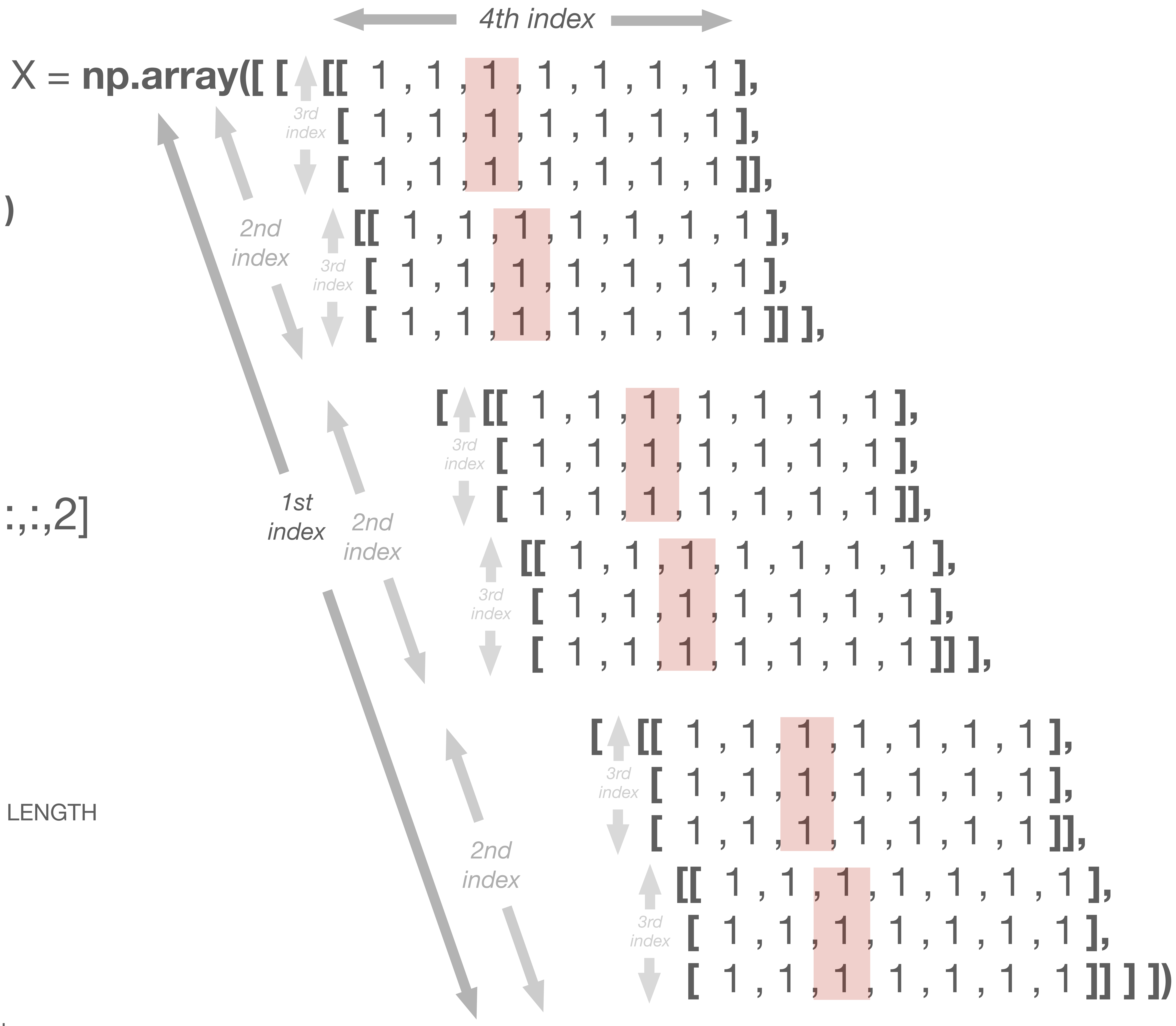
bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block

X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

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

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing   start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,:,:,3]

array indexing

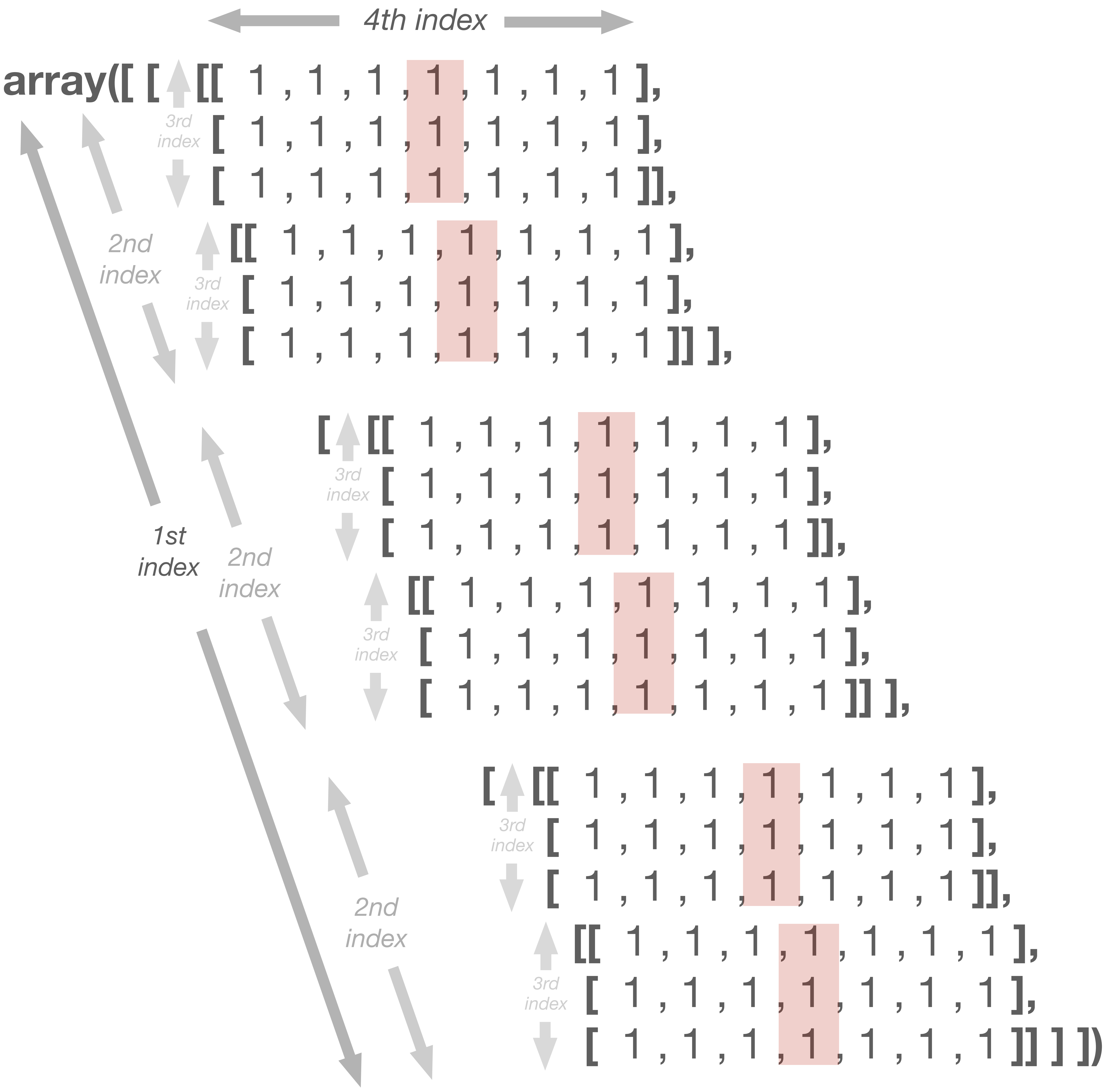
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,:,:,4]

array indexing

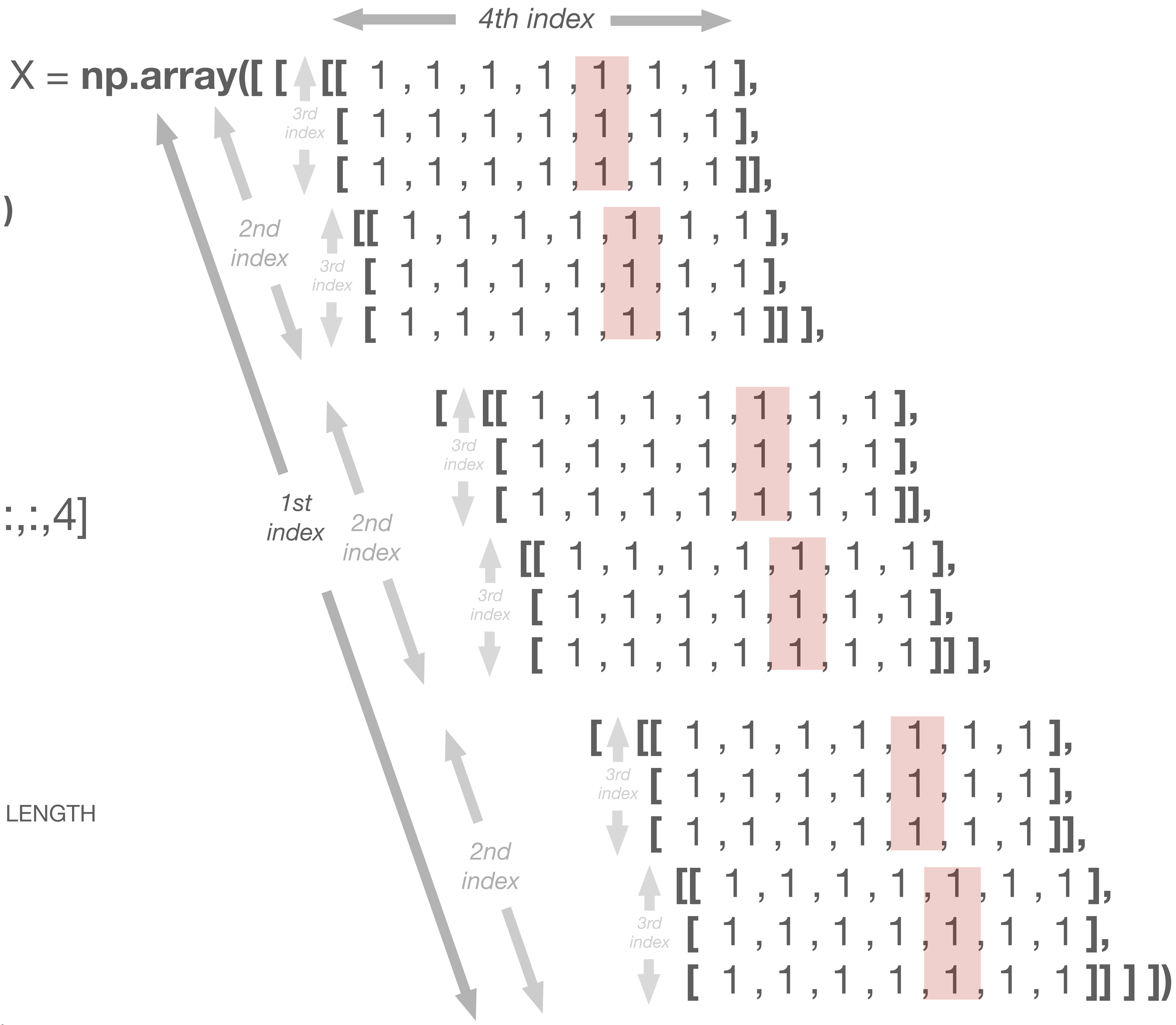
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
                   [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                      [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                       [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                       [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
                       [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                          [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                          [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                          [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                           [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                           [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array:

A = **np.array(** [[1, 2, 3],
[3, 2, 1]])

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,:,:,5]

array indexing

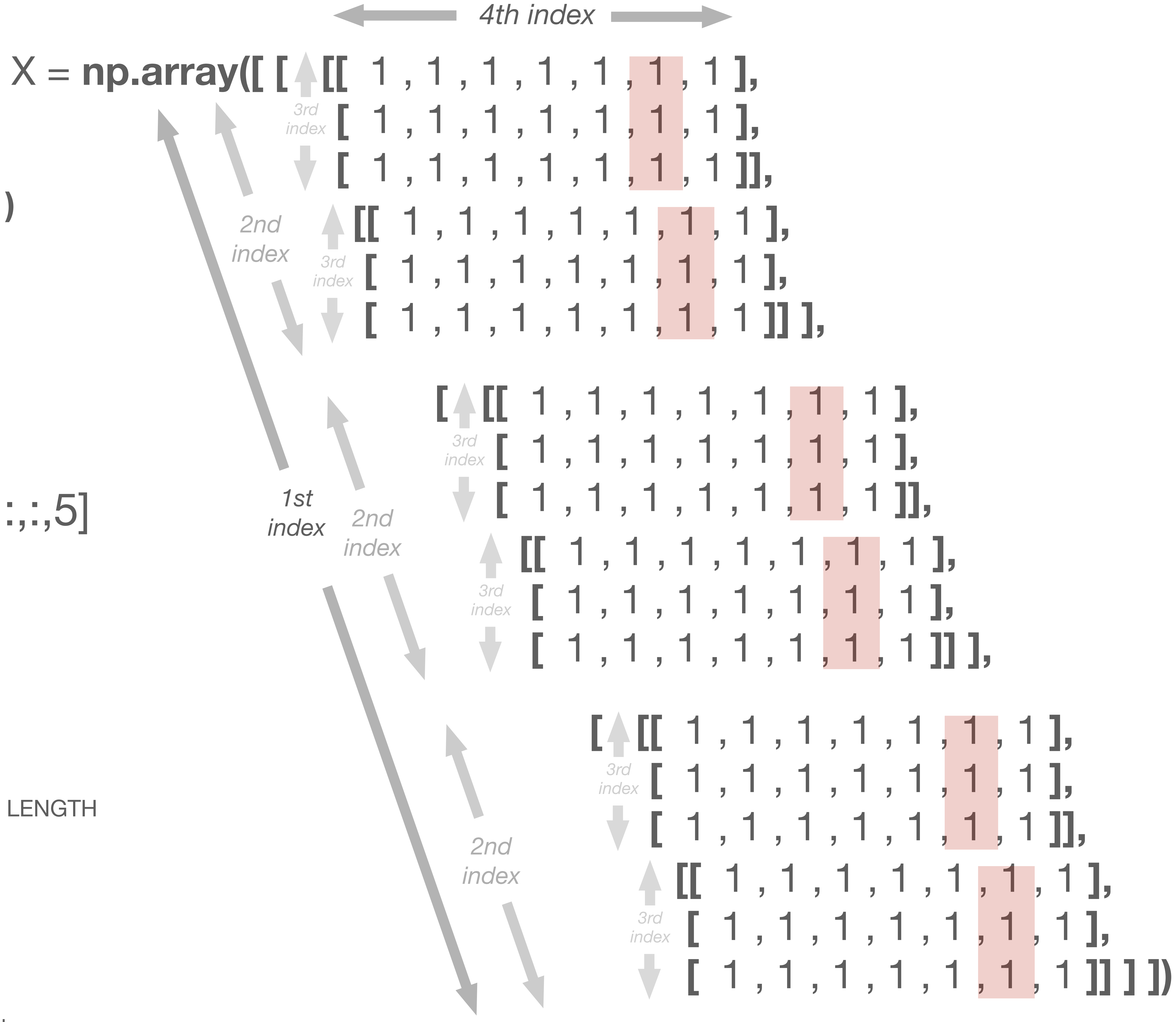
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
               [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array: A = **np.array(** [[1, 2, 3],
[3, 2, 1]])

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

X[:,:,:,6]

array indexing

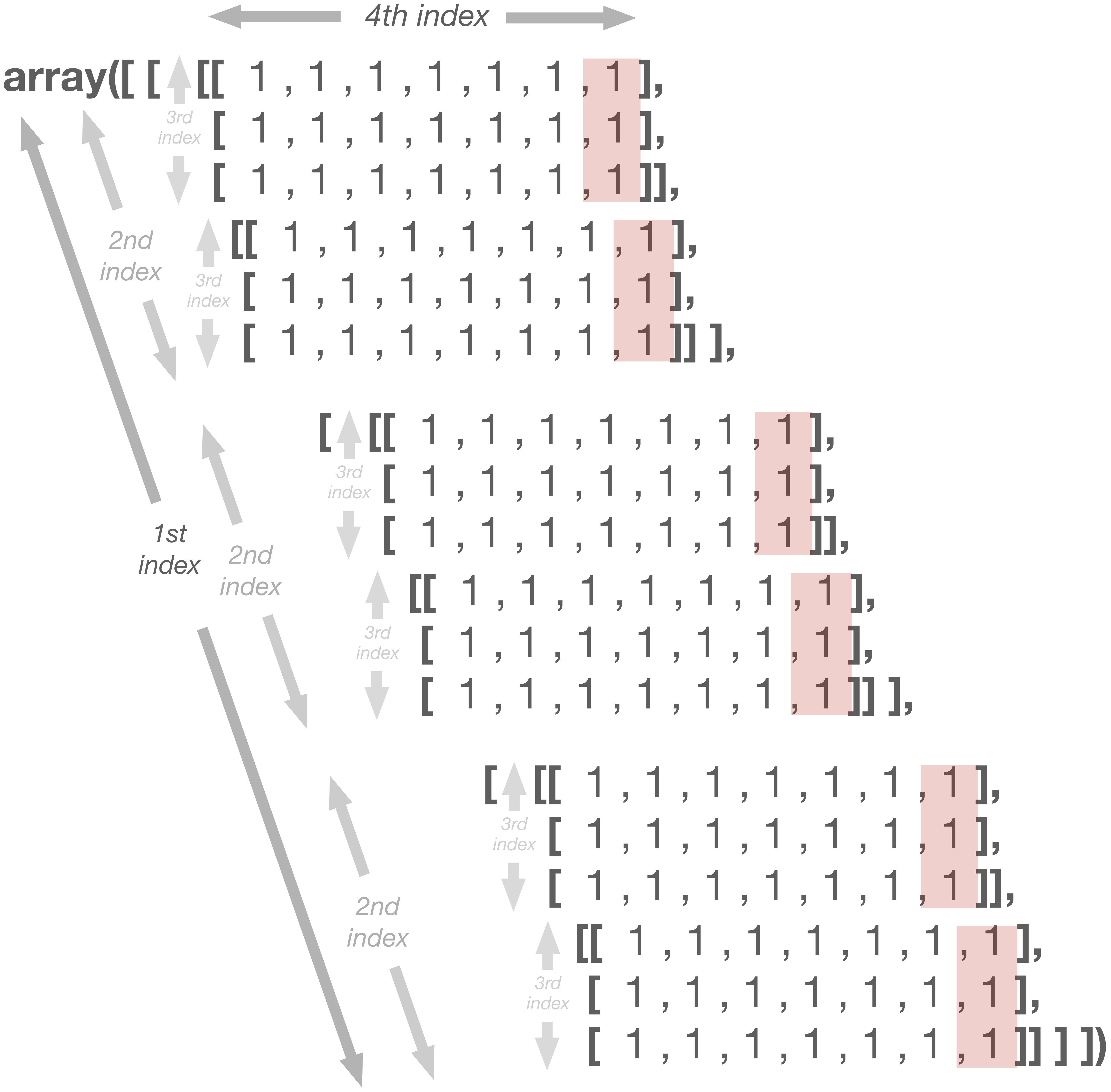
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array([ [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                  [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                   [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
                 [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                   [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ],
                  [ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                     [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                     [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
                    [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                     [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ],
                     [ 1 , 1 , 1 , 1 , 1 , 1 , 1 ]] ] ])
```

Python - Indexing

np.array: A = **np.array([[**1, 2, 3**],**
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

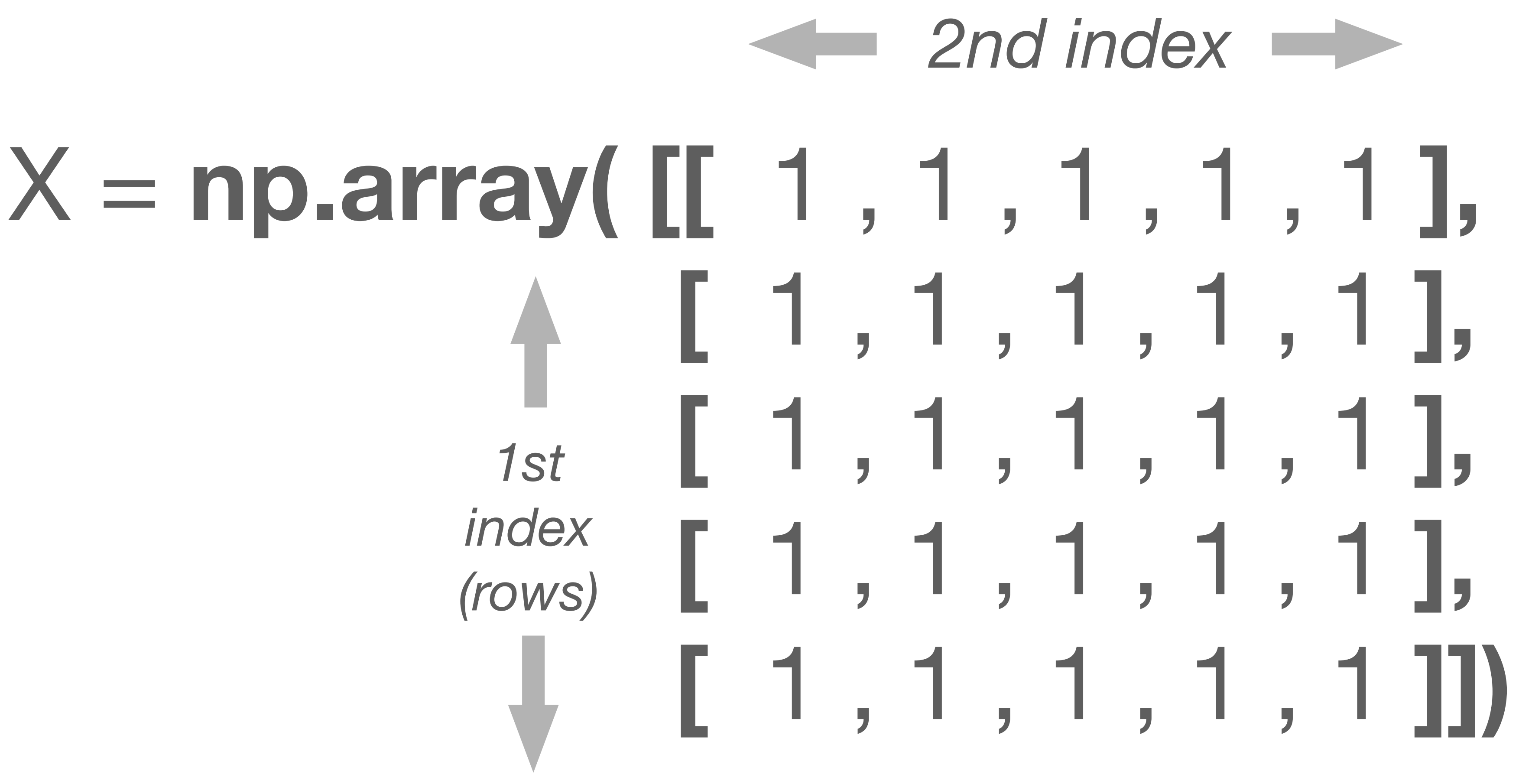
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

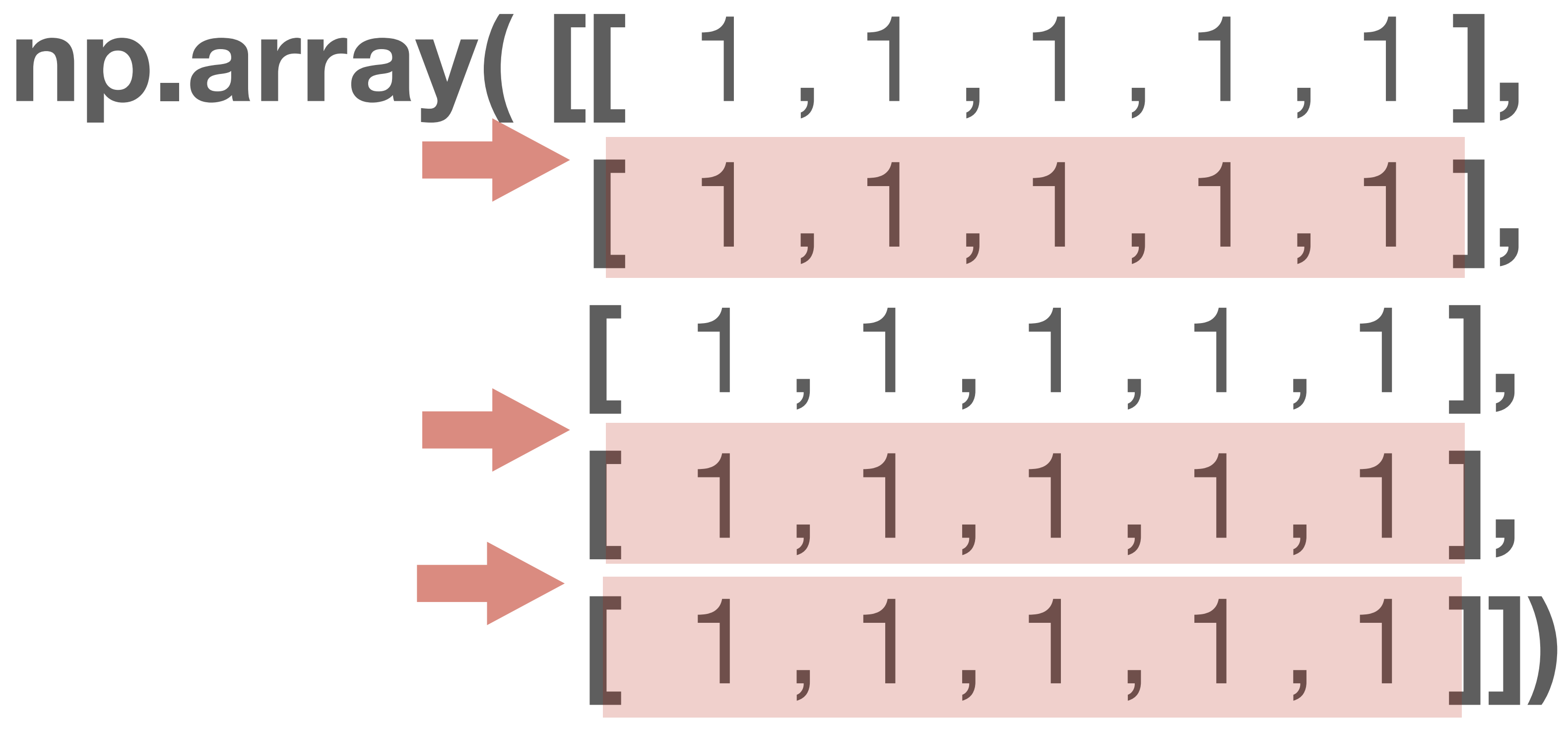
block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]])
```

X[[1,3,4]]



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

Python - Indexing

np.array:

A = **np.array(** **[[**1, 2, 3**]**,
[3, 2, 1**]]** **)**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing

start : end : step

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

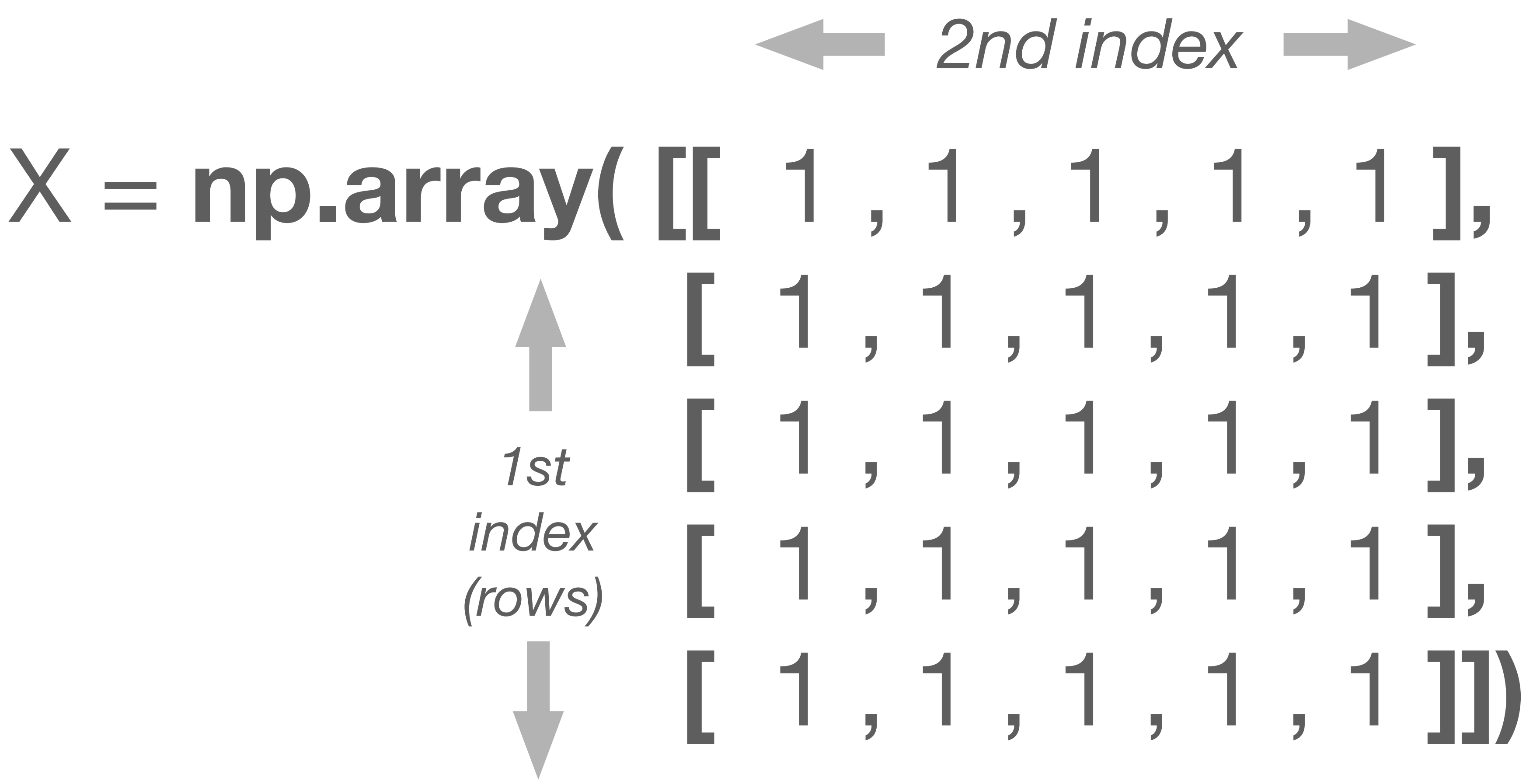
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

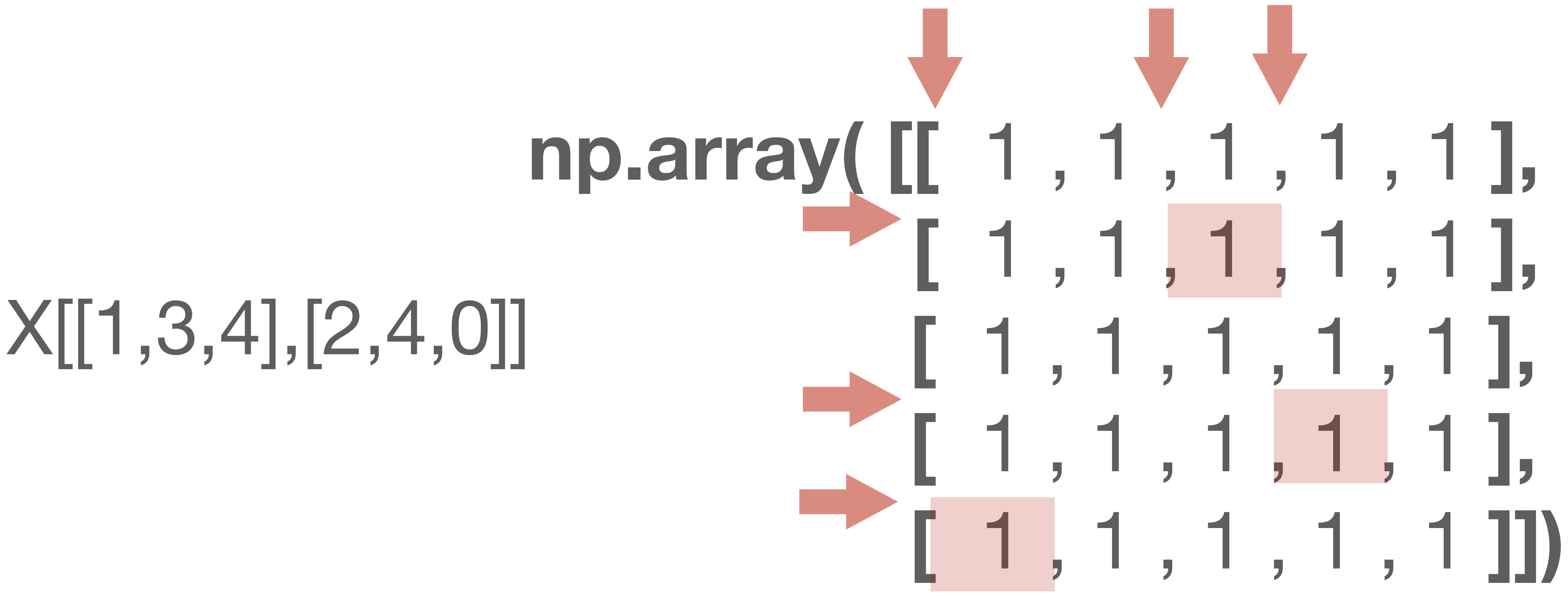
X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



2nd index

X = **np.array(** **[[** 1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **]])**

1st index (rows)



X[[1,3,4],[2,4,0]]

np.array(**[[** 1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **]])**

Python - Indexing

np.array: A = **np.array(** **[[**1, 2, 3**]**,
[3, 2, 1**]]** **)**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

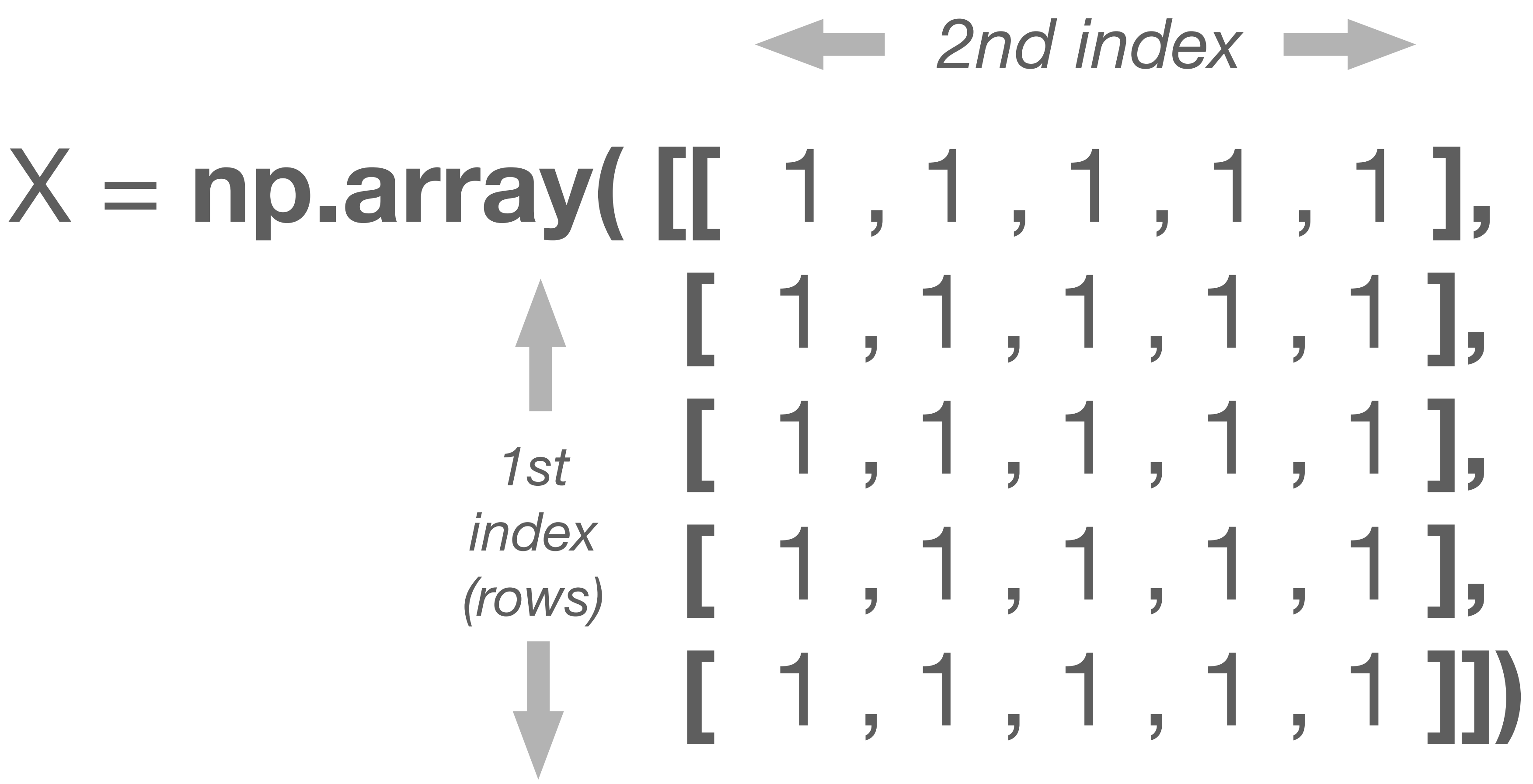
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

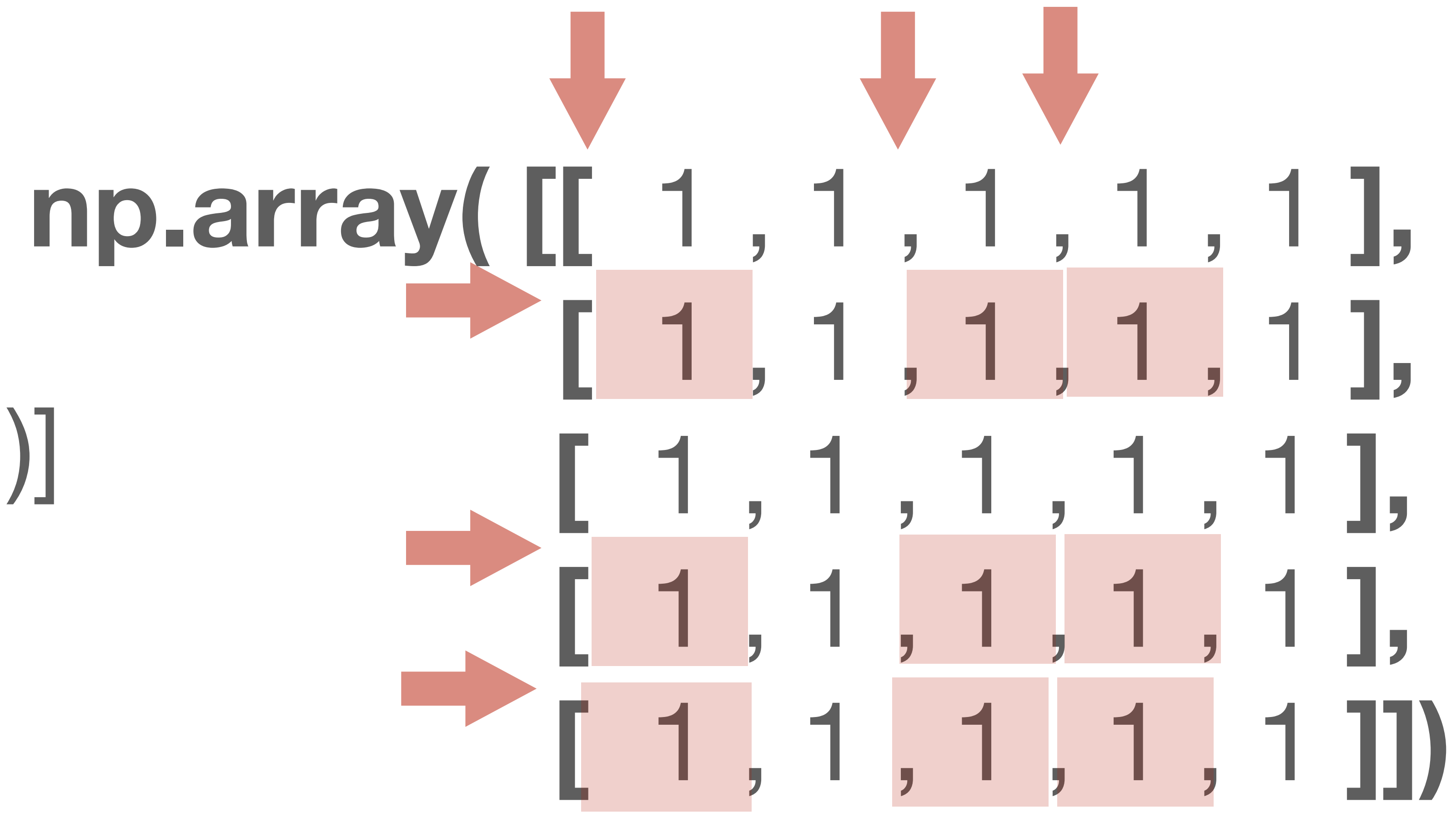
block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```

X[np.ix_([1,3,4],[2,4,0])]



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

Python - Indexing

np.array: A = **np.array(** [[1, 2, 3],
[3, 2, 1]])

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

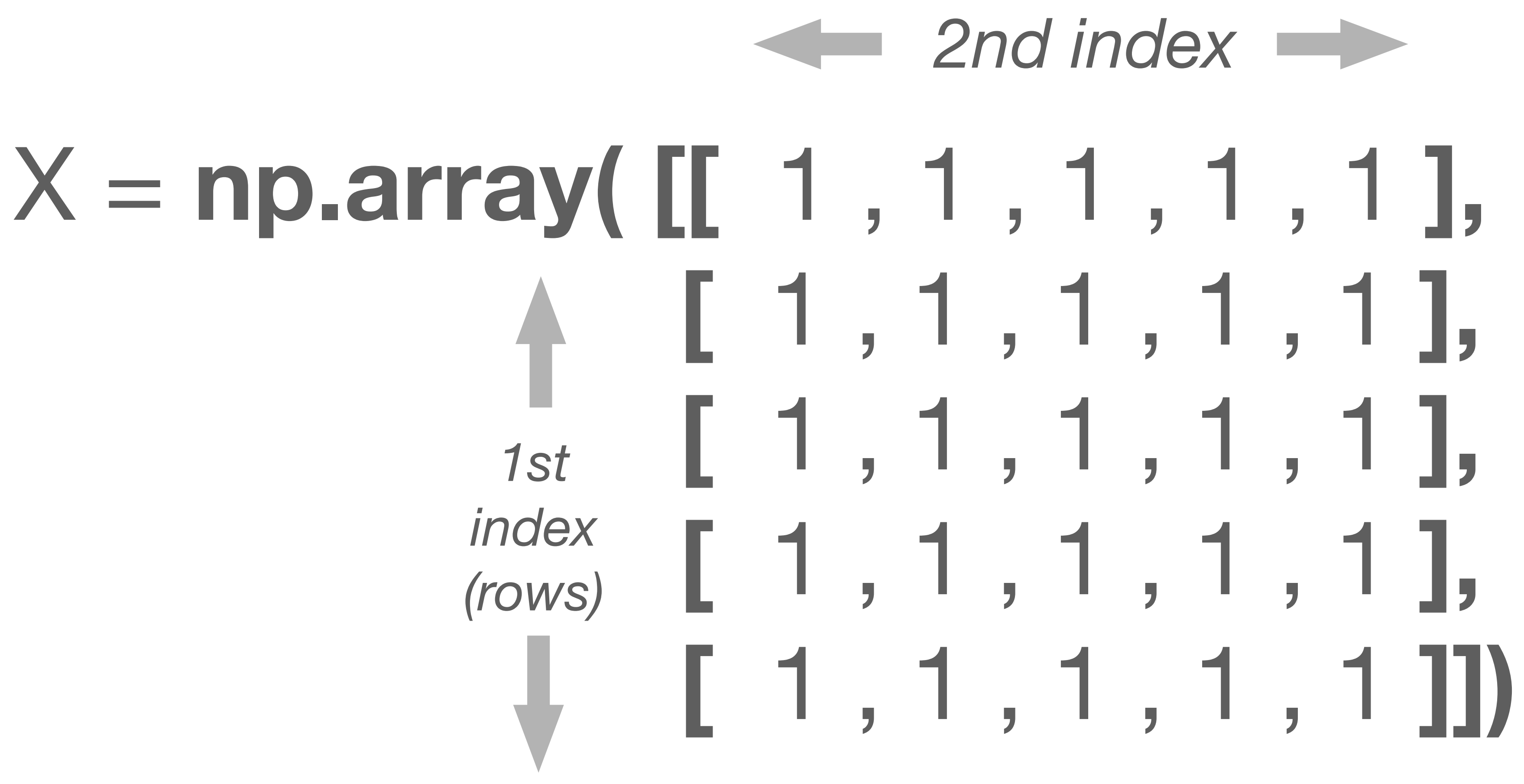
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



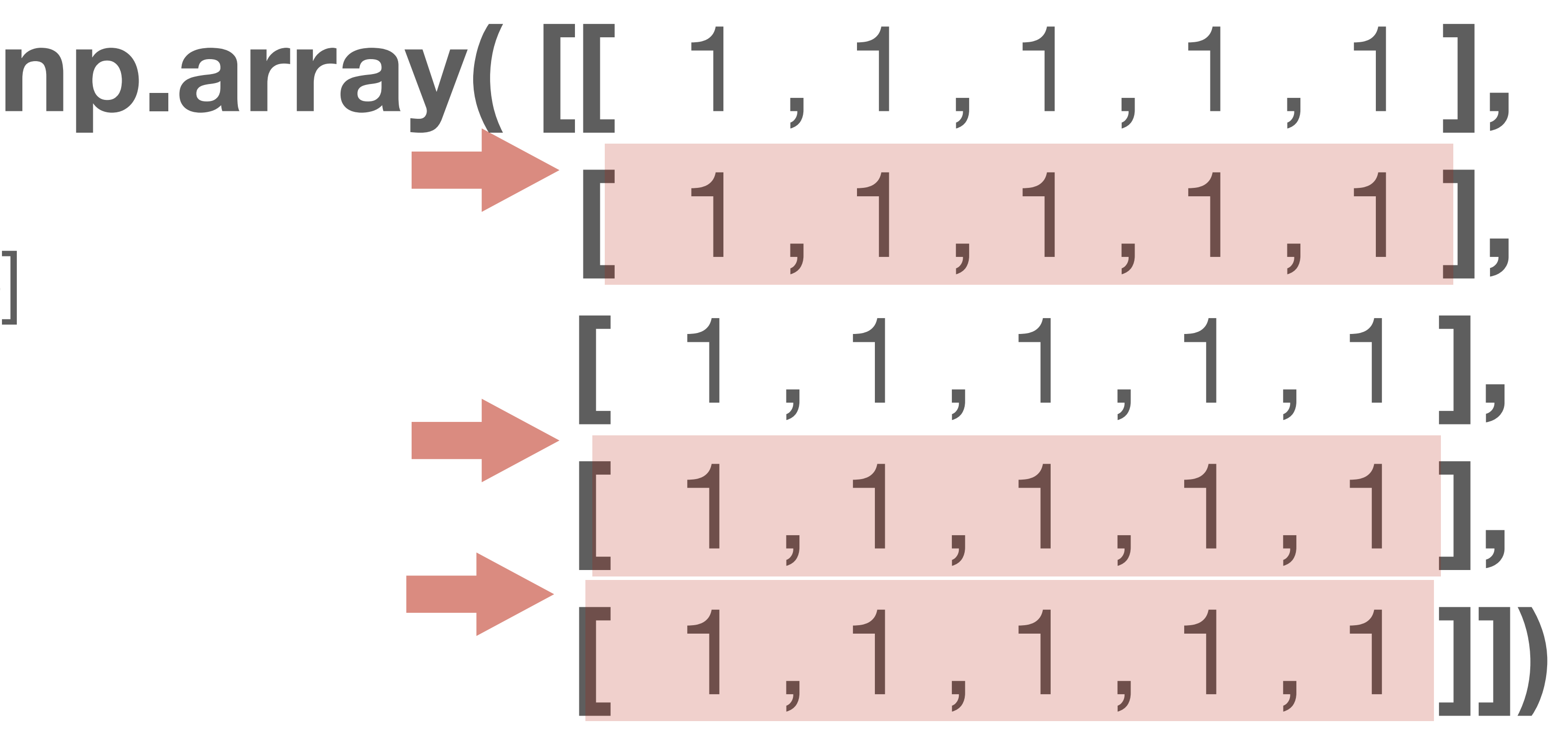
2nd index

X = **np.array(** [[1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1]])

1st index (rows)

bools = [False,True,False,True,True]

X[bools]



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

Python - Indexing

np.array: A = **np.array(** **[[**1, 2, 3**]**,
[3, 2, 1**]]** **)**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

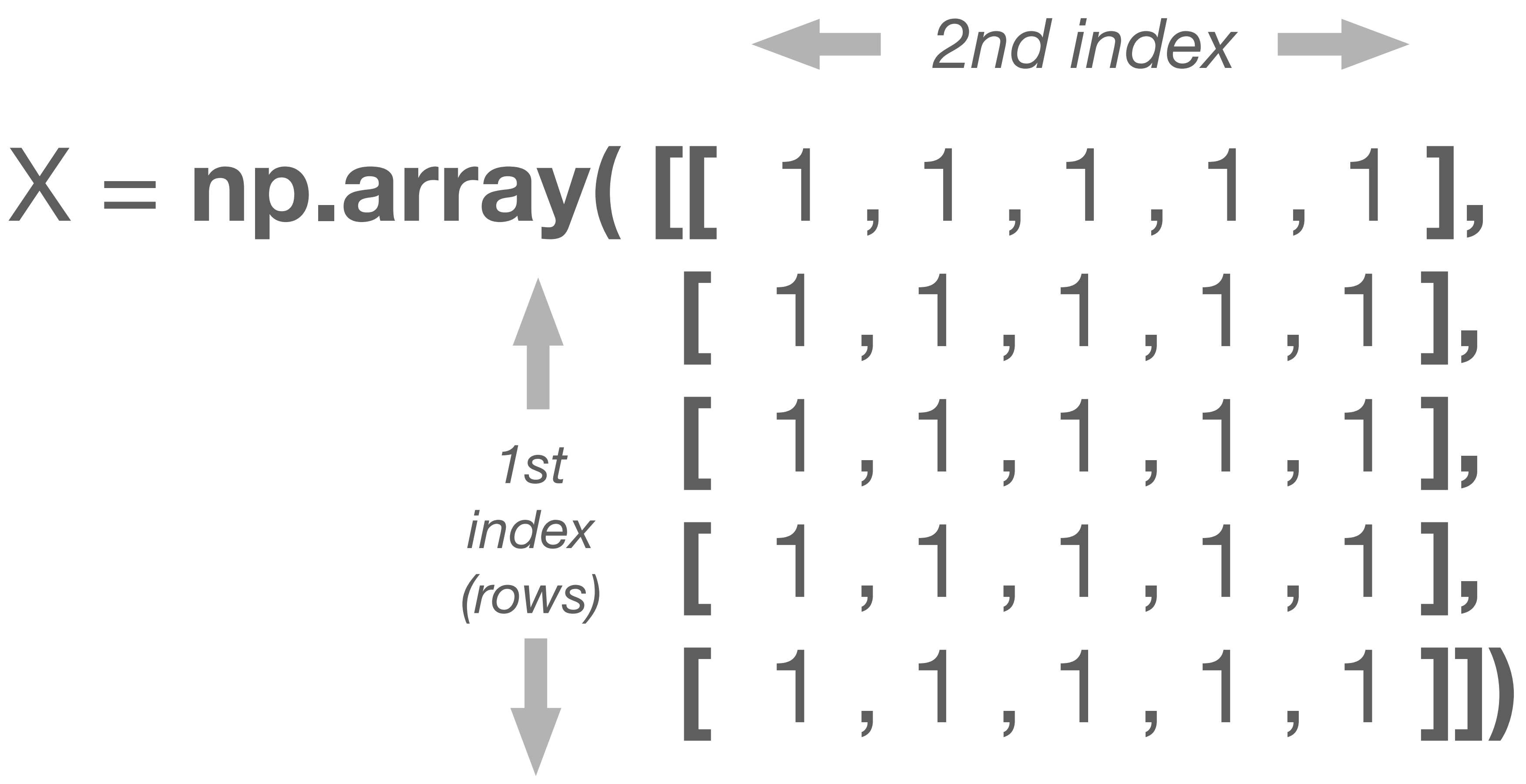
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



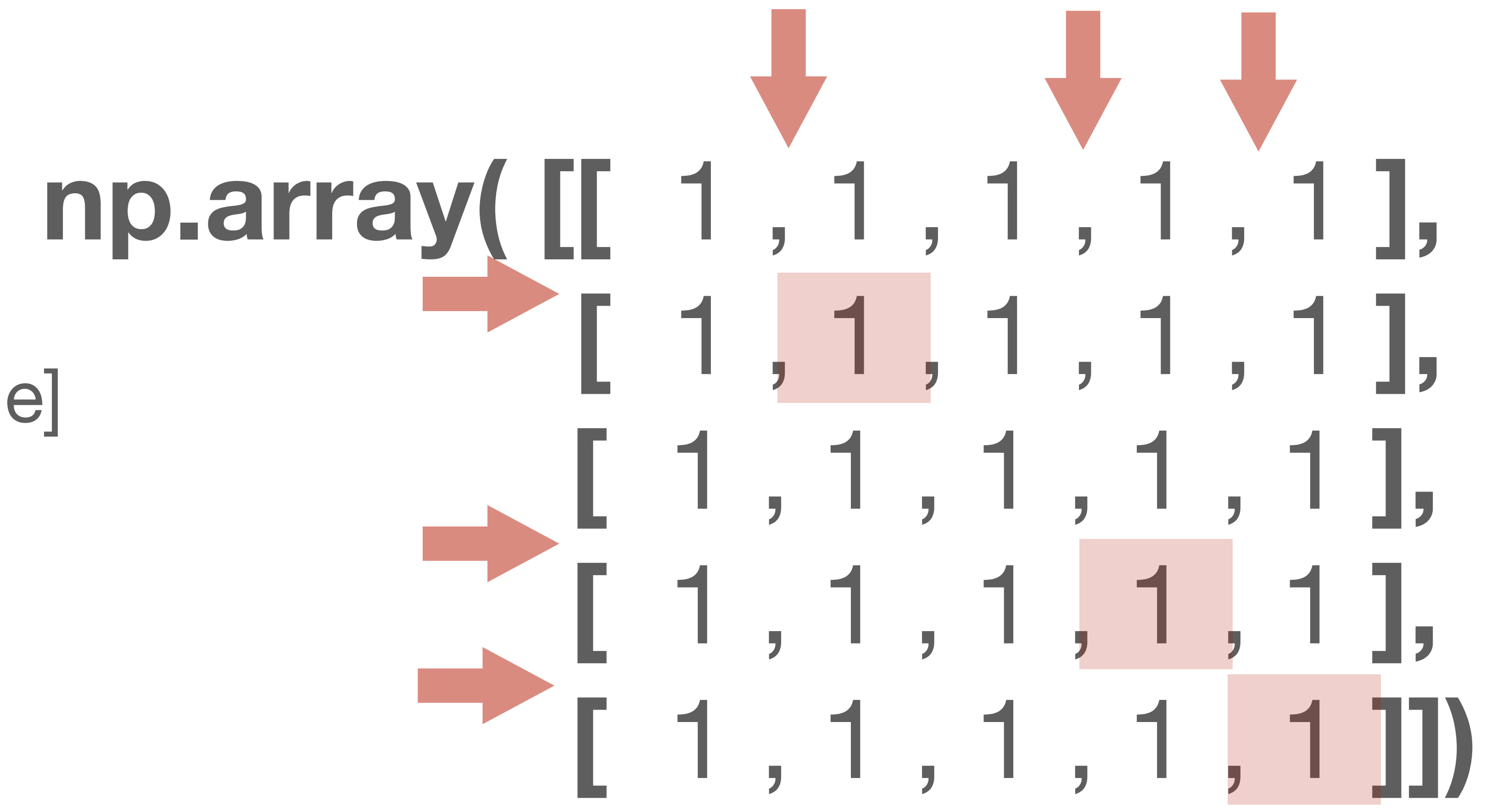
2nd index

1st index (rows)

X = **np.array([[** 1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **]])**

bools = [False,True,False,True,True]

X[bools,bools]



np.array([[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **],**
[1 , 1 , 1 , 1 , 1 **]])**

Python - Indexing

np.array: A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

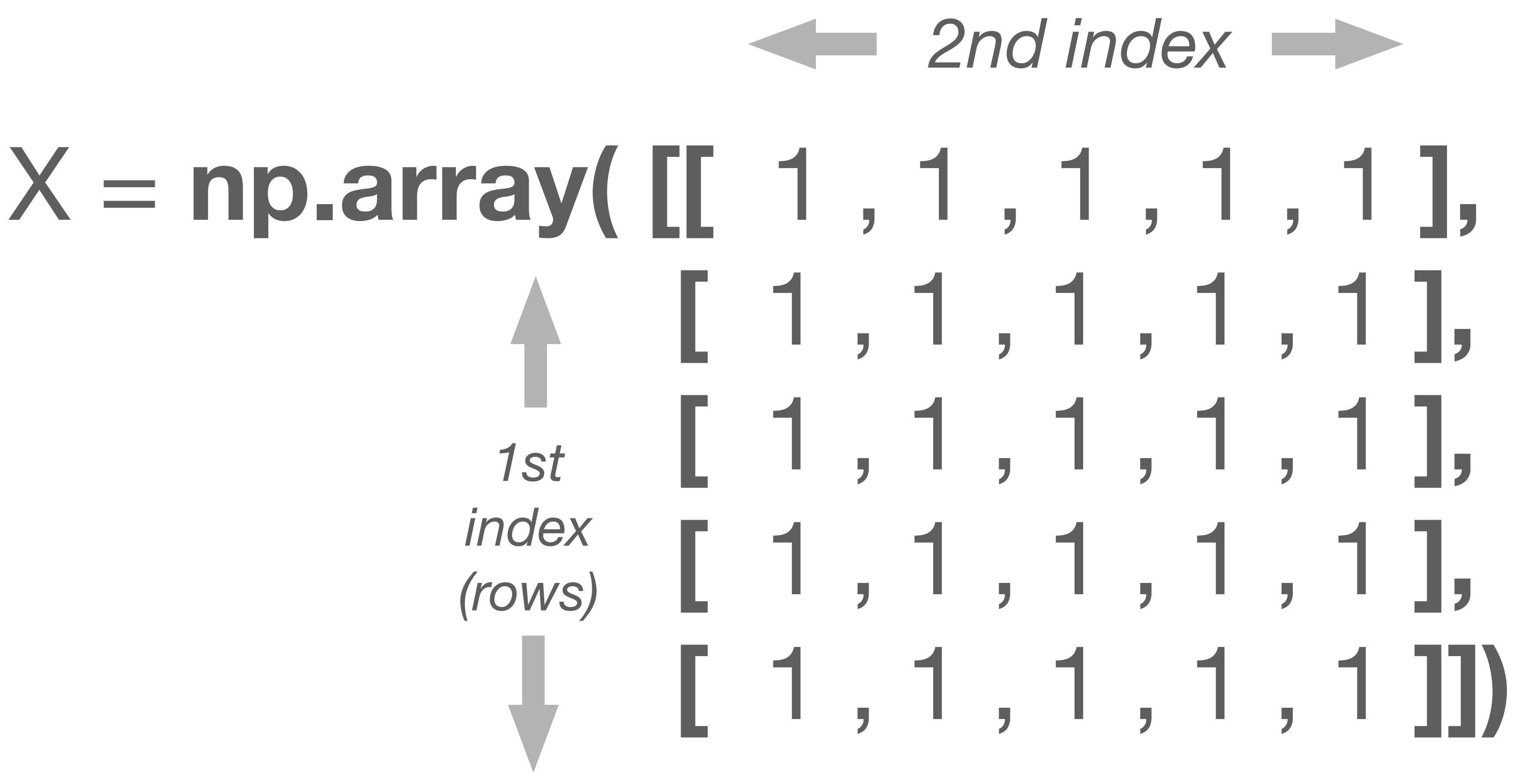
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

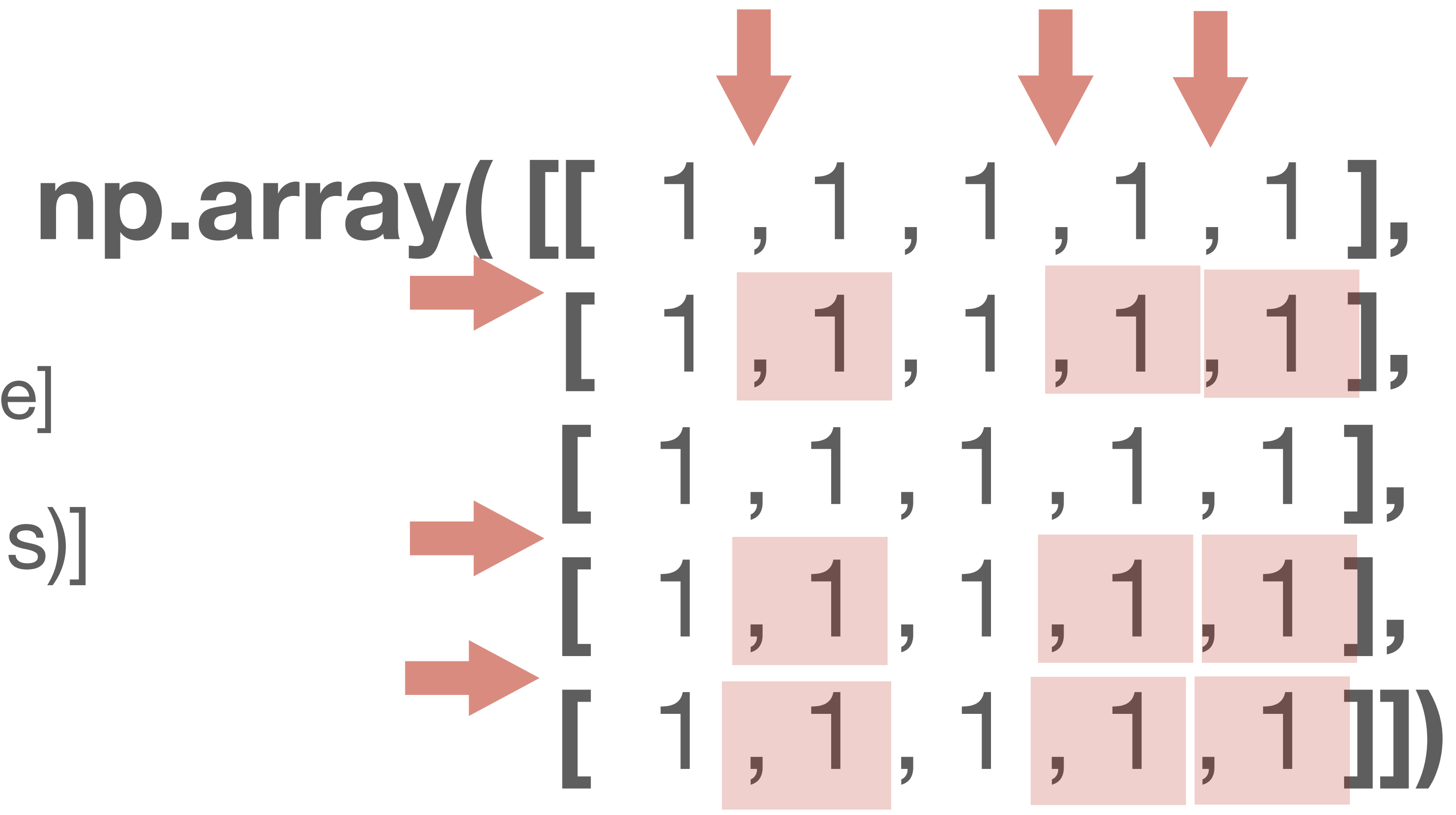
X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block

bools = [False,True,False,True,True]

X[np.ix_(bools,bools)]



```
X = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```



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

Python - Indexing

np.array:

A = **np.array([[**1, 2, 3**]**,
[3, 2, 1**]])**

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements

ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

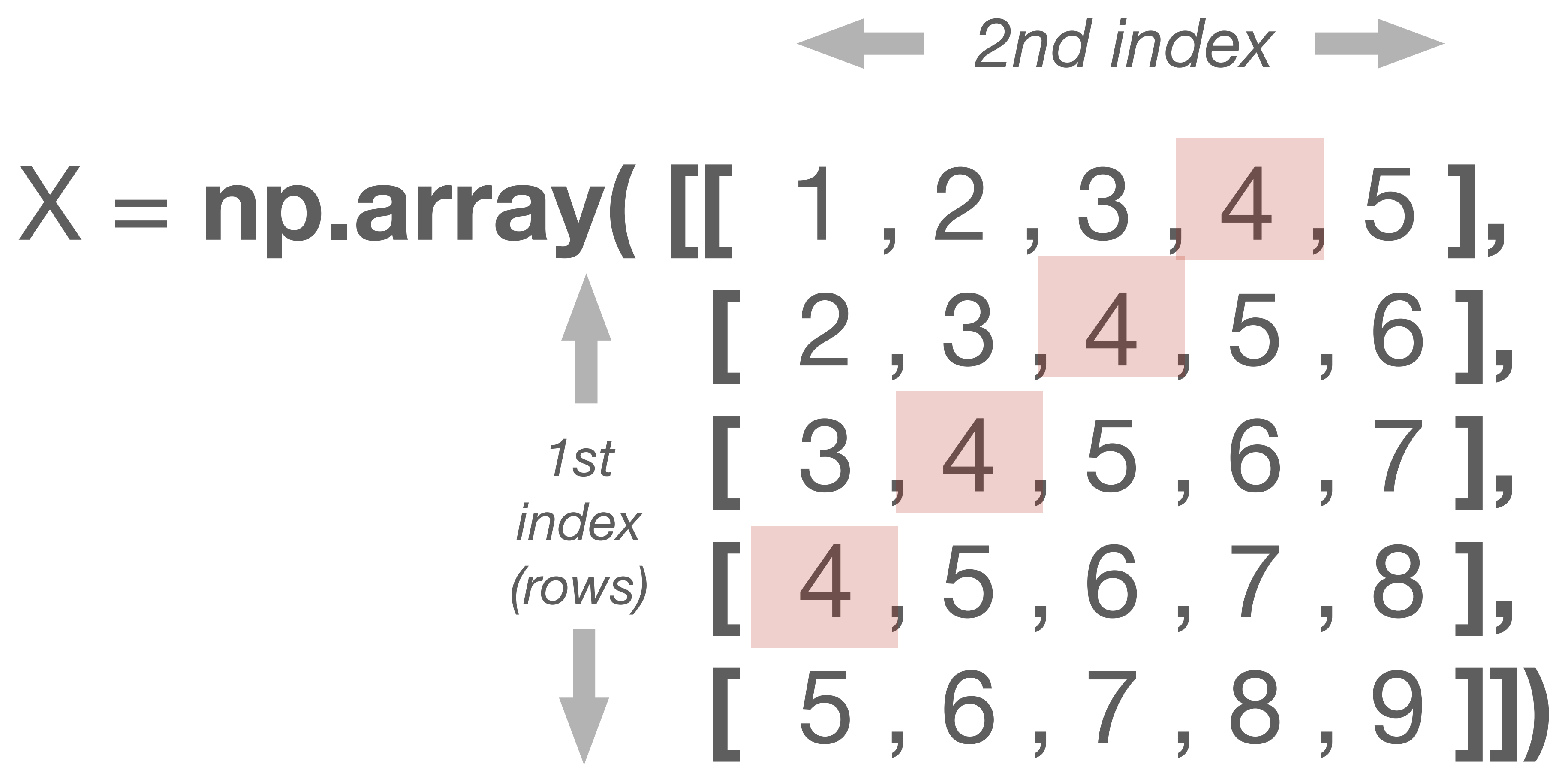
boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.

X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block



2nd index

X = **np.array([[** 1 , 2 , 3 , 4 , 5 **],**
[2 , 3 , 4 , 5 , 6 **],**
[3 , 4 , 5 , 6 , 7 **],**
[4 , 5 , 6 , 7 , 8 **],**
[5 , 6 , 7 , 8 , 9 **]])**

1st index (rows)

Finding Elements:

np.where(X==4) = [(0,1,2,3,), (3,2,1,0)]

X[np.where(X==4)] = [4 , 4 , 4 , 4]

…returns all elements of array satisfying condition collapsed

Python - Indexing

np.array:

A = **np.array(** [[1, 2, 3],
[3, 2, 1]])

zero indexed

x[0] - *first element…*
x[1] - *second element…*

negative indexing

x[-1] - last element…

slicing **start : end : step**

x[k1:k2:s1] - from k1 to k2 step by s1

array indexing

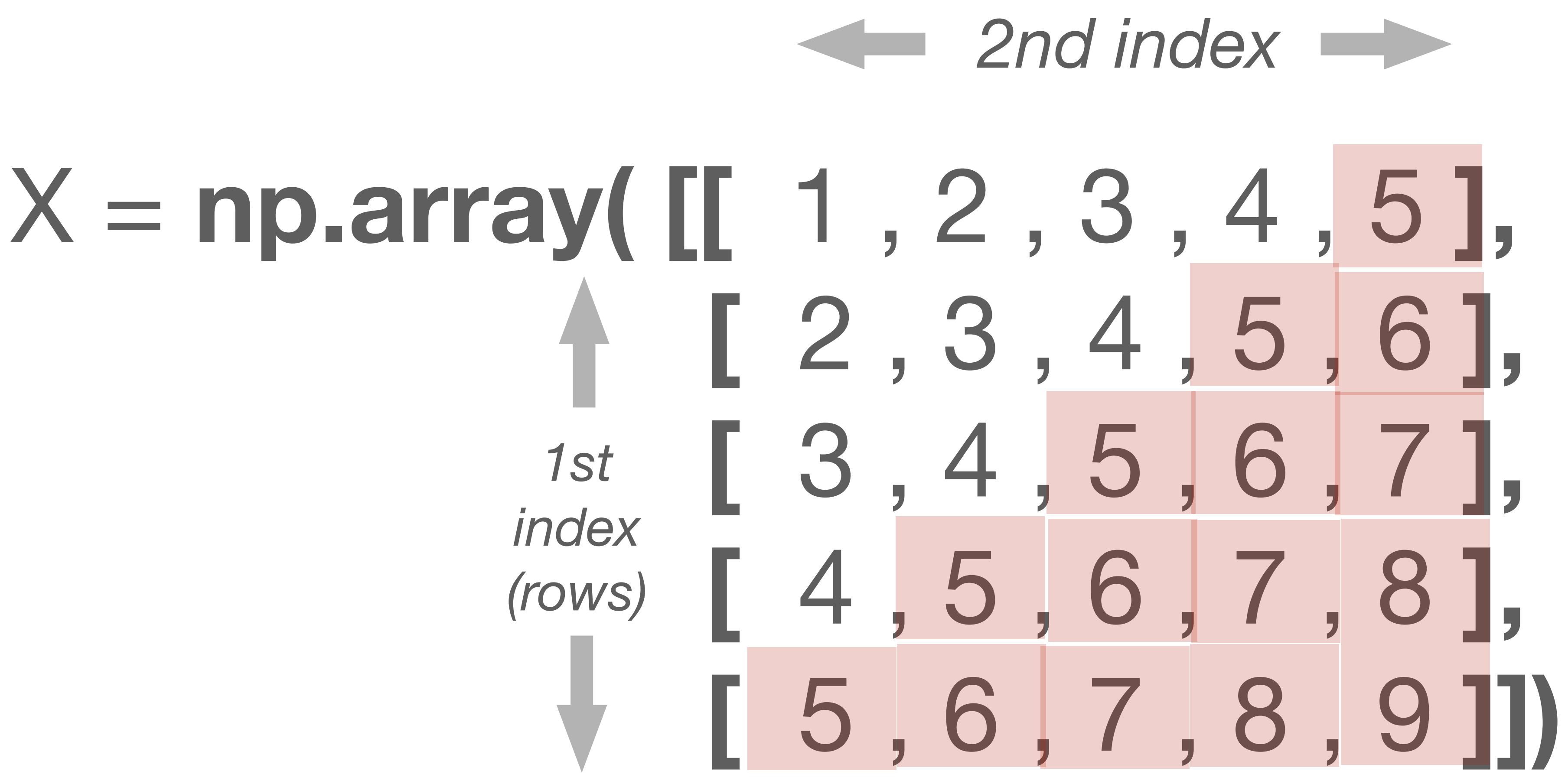
ind = [0, 2, 3];
x[ind] - returns 0,2, and 3 elements
ind1 = [0, 2, 3]; ind2 = [0,3,2];
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

boolean indexing

bool = **[True, True, False, True];** MUST BE ARRAY LENGTH
x[bool] - returns 0,1, and 3 element.
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

X[np.ix_(ind1,ind2)] - returns the [0,2,3] x [3,2] block
X[np.ix_(bool,bool)] - returns the [0,1,3] x [0,1,3] block

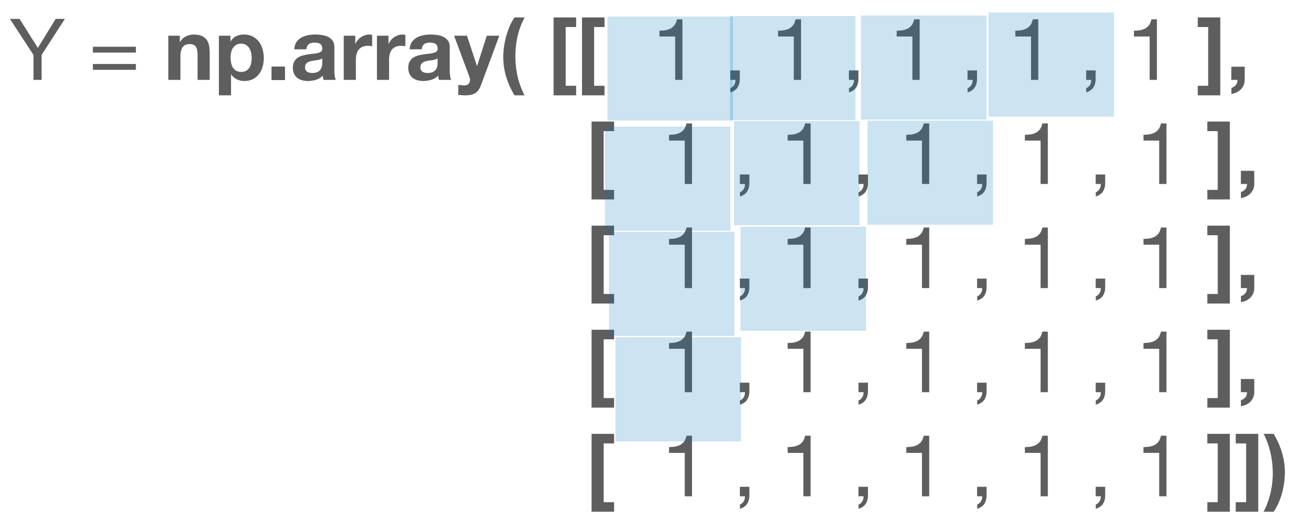


```
X = np.array( [[ 1 , 2 , 3 , 4 , 5 ],
               [ 2 , 3 , 4 , 5 , 6 ],
               [ 3 , 4 , 5 , 6 , 7 ],
               [ 4 , 5 , 6 , 7 , 8 ],
               [ 5 , 6 , 7 , 8 , 9 ]])
```

Finding Elements:

np.where(condition, X, Y)

…chooses elements from X if true…
…chooses elements from Y if false…
…respects array structure…

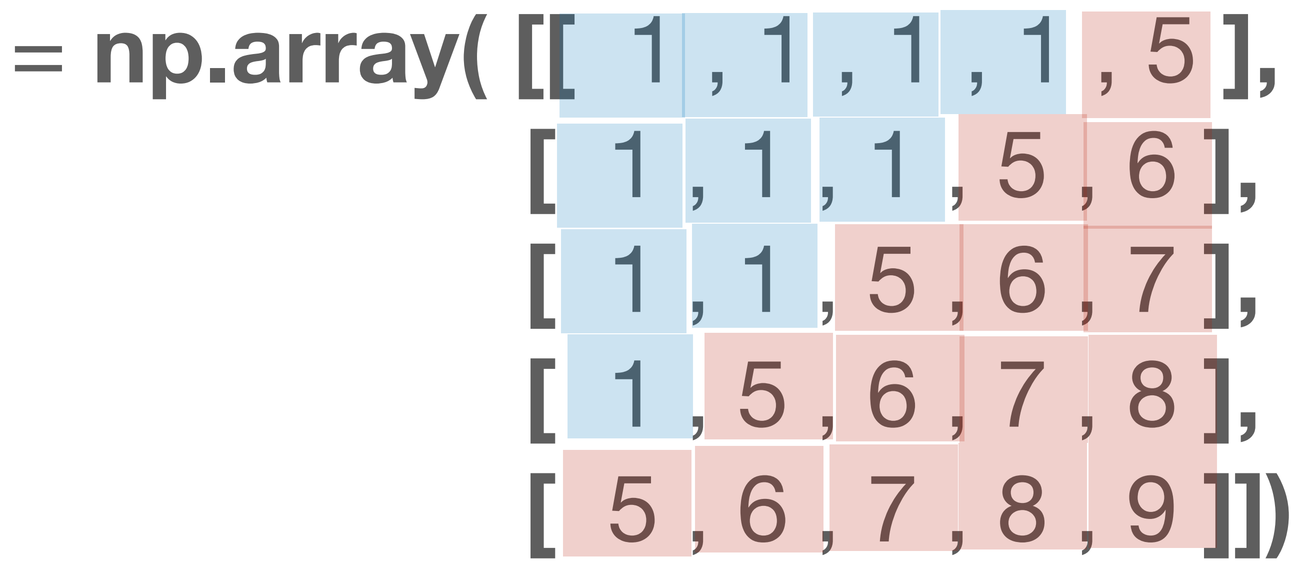


```
Y = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```

np.where(X >= 5, X, Y)

or…

np.where(X >= 5, X, 1)



```
= np.array( [[ 1 , 1 , 1 , 1 , 5 ],
             [ 1 , 1 , 1 , 5 , 6 ],
             [ 1 , 1 , 5 , 6 , 7 ],
             [ 1 , 5 , 6 , 7 , 8 ],
             [ 5 , 6 , 7 , 8 , 9 ]])
```

Python - Matrix Multiplication

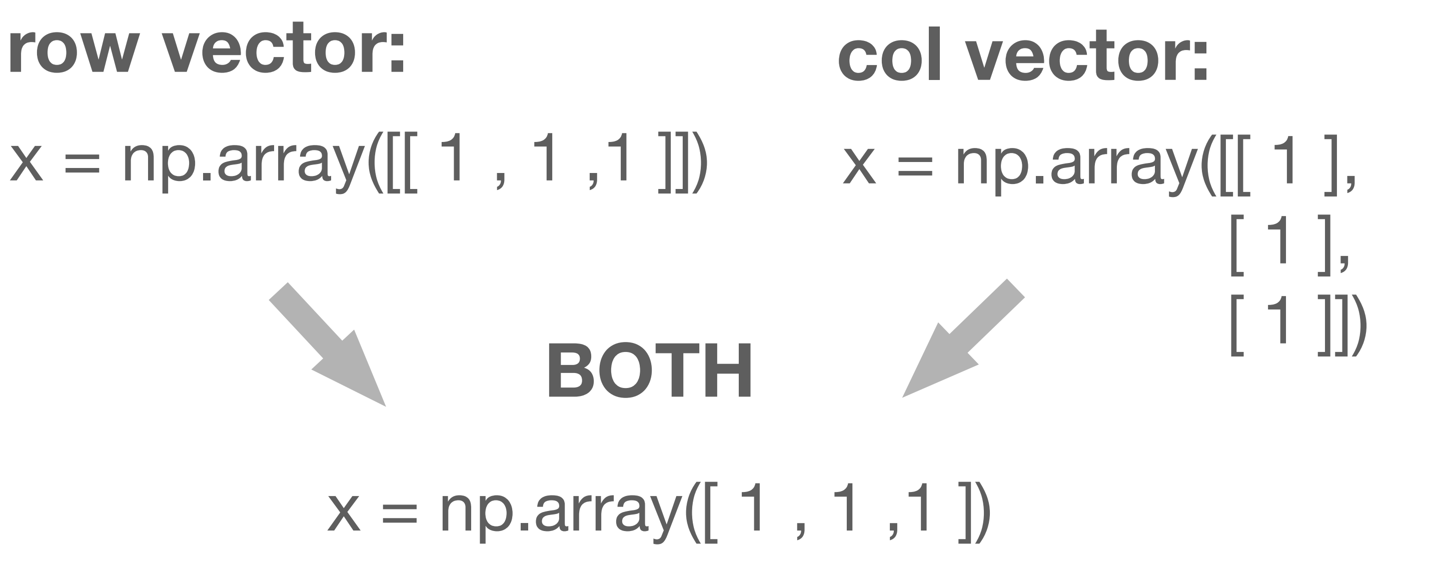
$$y = Ax$$

$$y_i = \sum_j A_{ij} x_j = A_{ij} x_j$$

vector: 1D x = np.array([1,2,3])

matrix 2D

```
A = np.array([[ 1, 1, 1 ],
              [ 1, 1, 1 ],
              [ 1, 1, 1 ]])
```



row vector:

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

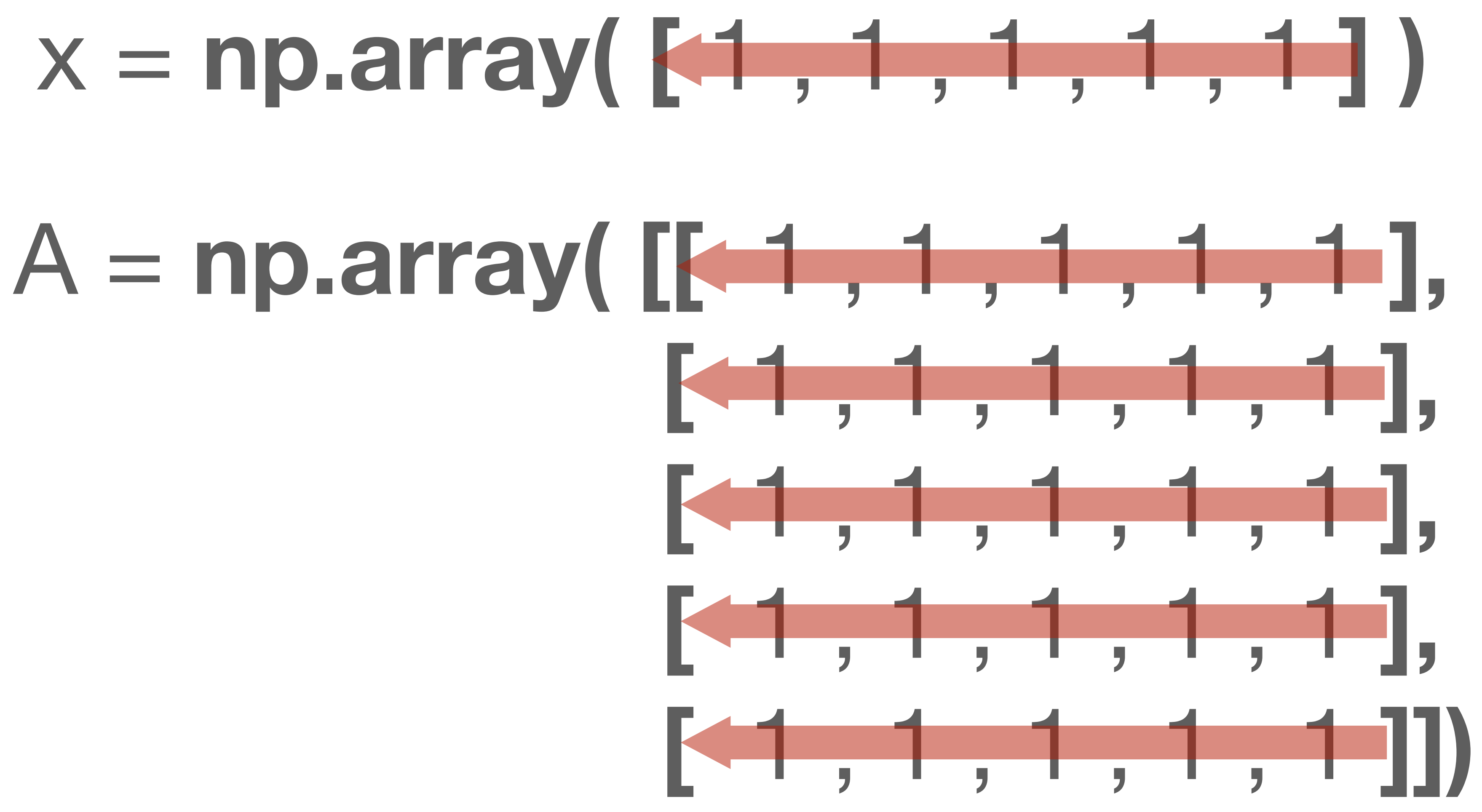
col vector:

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

BOTH

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

A @ x



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

A = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```

Matrix multiplication:

A@x = A.dot(x) = np.dot(A,x)

x = np.array([1,1,1])

A@x - A times col vector x
x@A - row vector x times A

np.einsum(‘ij,j’,A,x)

Tranpose A.T

Python - Matrix Multiplication

vector: 1D `x = np.array([1,2,3])`

matrix 2D

```
A = np.array([[ 1, 1, 1 ],
              [ 1, 1, 1 ],
              [ 1, 1, 1 ]])
```

row vector:

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

col vector:

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

BOTH

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

Matrix multiplication:

A@x = A.dot(x) = np.dot(A,x)

x = np.array([1,1,1])

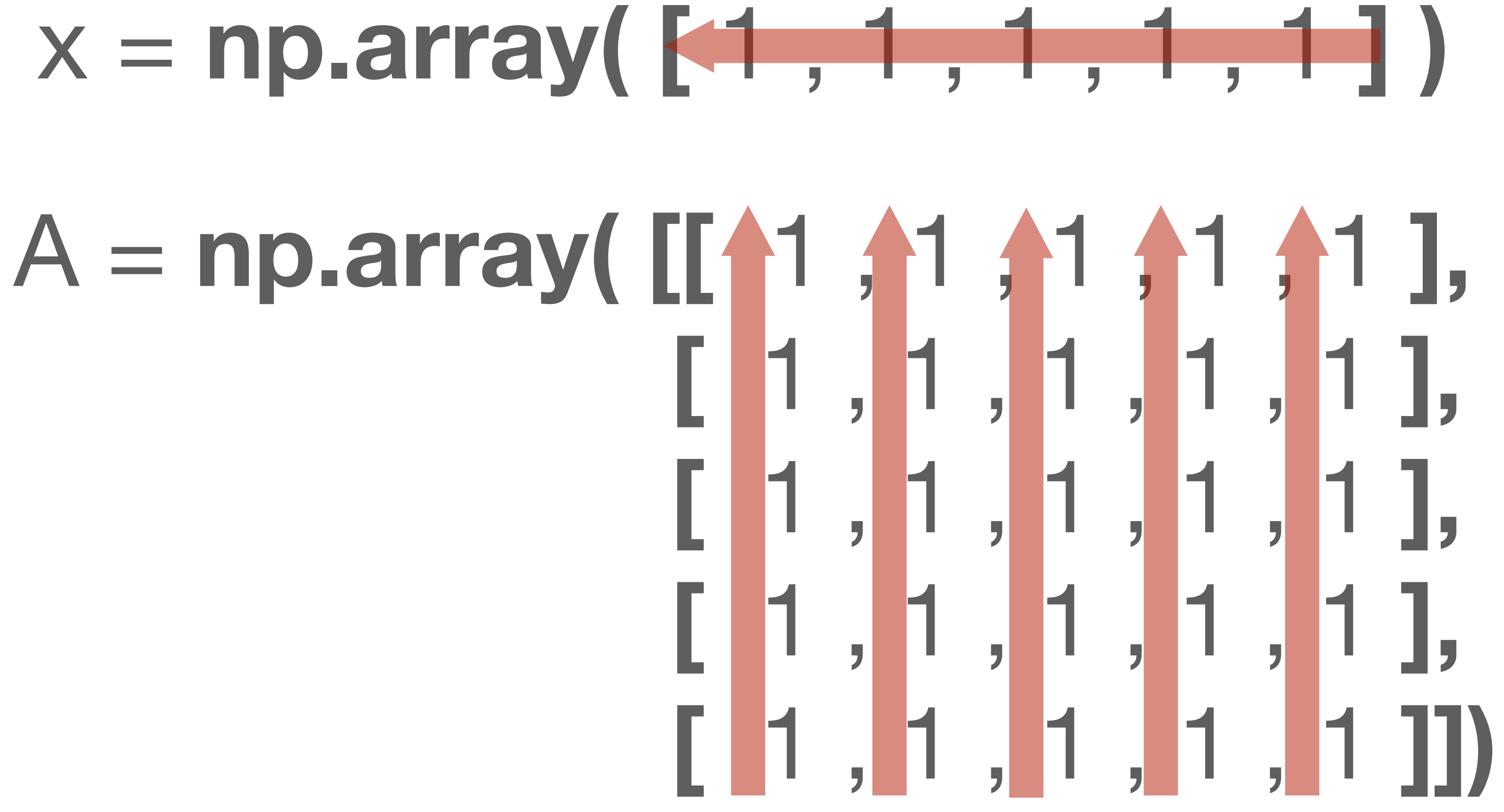
A@x - A times col vector x
x@A - row vector x times A

Tranpose A.T

$$x^T A = y^T$$

$$y_j = \sum_i A_{ij} x_i = A_{ij} x_i$$

x @ A



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

A = np.array( [[ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ],
               [ 1 , 1 , 1 , 1 , 1 ]])
```

np.einsum(‘ij,i’,A,x)

np.einsum(‘i,ij’,x,A)

Python - Matrix Multiplication

vector: 1D x = np.array([1,2,3])

matrix 2D
```
A = np.array([[ 1, 1, 1 ],
              [ 1, 1, 1 ],
              [ 1, 1, 1 ]])
```

row vector:

x = np.array([[1 , 1 ,1]])

col vector:
```
x = np.array([[ 1 ],
              [ 1 ],
              [ 1 ]])
```

BOTH

x = np.array([1 , 1 ,1])

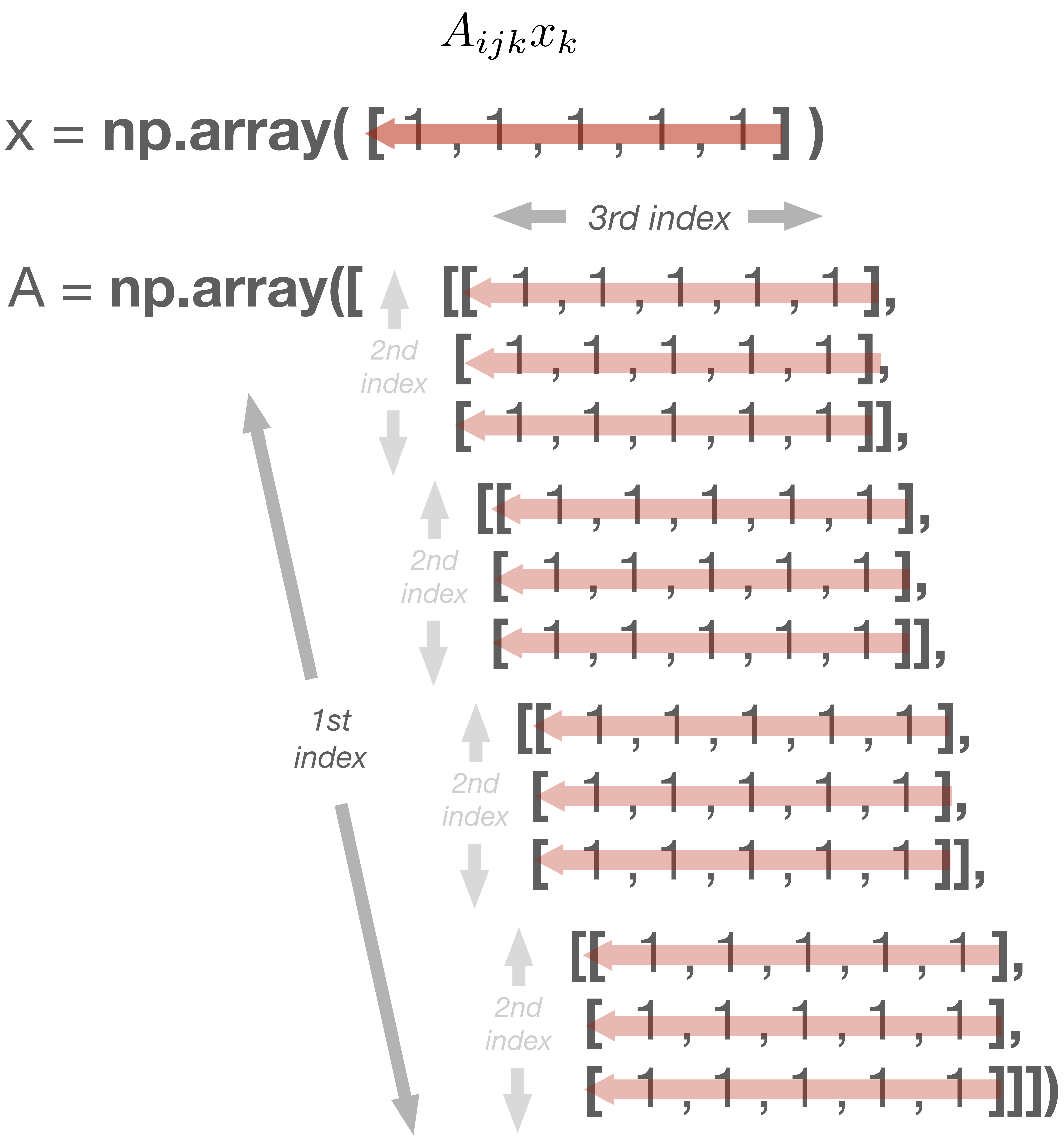
Matrix multiplication:

A@x = A.dot(x) = np.dot(A,x)

x = np.array([1,1,1])

A@x - A times col vector x
x@A - row vector x times A

Tranpose A.T



$A_{ijk}x_k$

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

A = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ]],
                   [[ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ]],
                      [[ 1 , 1 , 1 , 1 , 1 ],
                       [ 1 , 1 , 1 , 1 , 1 ],
                       [ 1 , 1 , 1 , 1 , 1 ]]])
```

np.einsum(‘ijk,k’,A,x)

Python - Matrix Multiplication

vector: 1D x = np.array([1,2,3])

matrix 2D

```
A = np.array([[ 1, 1, 1 ],
              [ 1, 1, 1 ],
              [ 1, 1, 1 ]])
```

row vector:

x = np.array([[1 , 1 ,1]])

col vector:

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

BOTH

x = np.array([1 , 1 ,1])

Matrix multiplication:

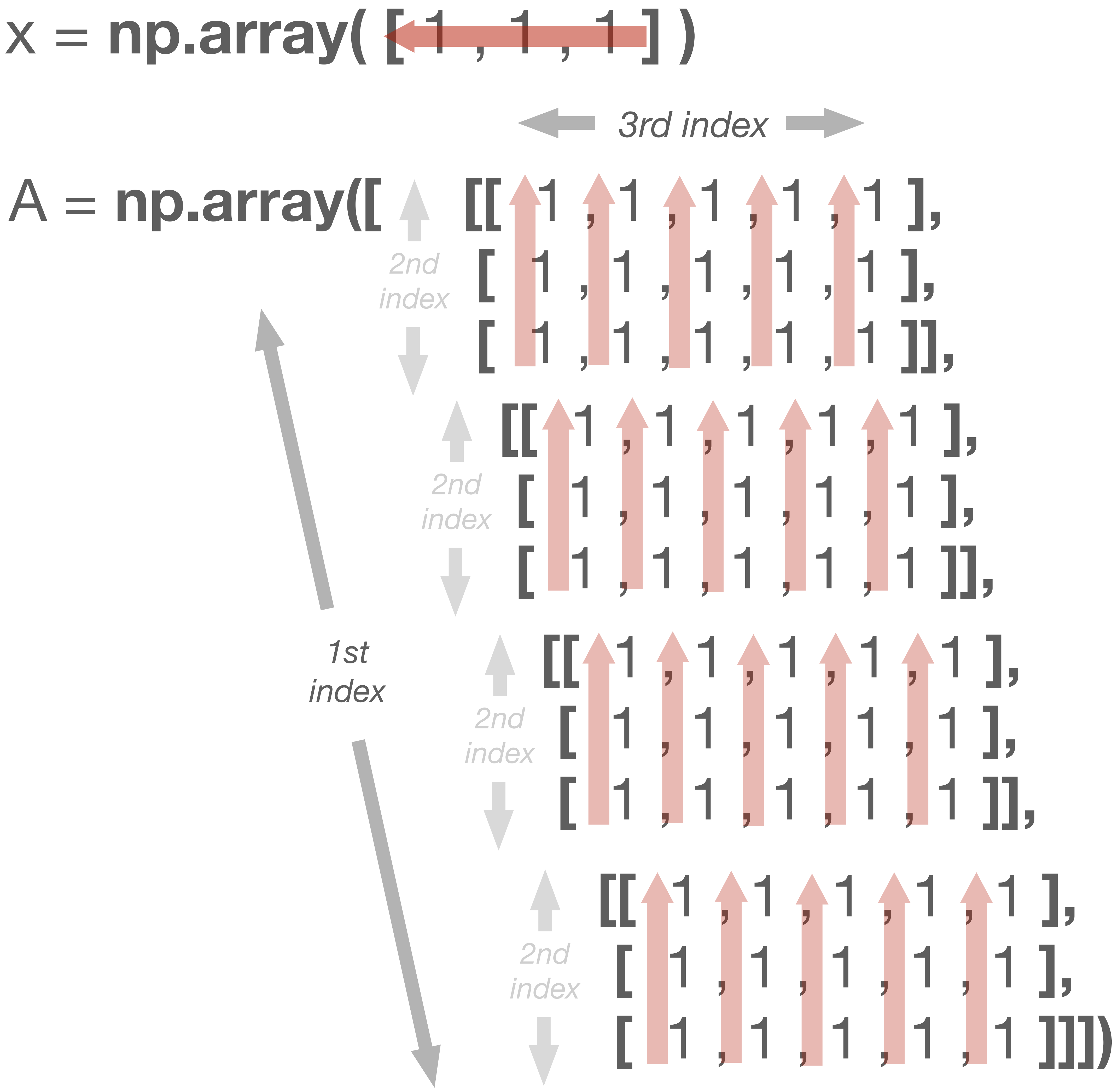
A@x = A.dot(x) = np.dot(A,x)

x = np.array([1,1,1])

A@x - A times col vector x
x@A - row vector x times A

Tranpose A.T

$$A_{ijk}x_j$$



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

A = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
                 [[ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ],
                  [ 1 , 1 , 1 , 1 , 1 ]],
                   [[ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ],
                    [ 1 , 1 , 1 , 1 , 1 ]],
                     [[ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ],
                      [ 1 , 1 , 1 , 1 , 1 ]]])
```

np.einsum('ijk,j',A,x)

Python - Matrix Multiplication

vector: 1D x = np.array([1,2,3])

matrix 2D
```
A = np.array([[ 1, 1, 1 ],
              [ 1, 1, 1 ],
              [ 1, 1, 1 ]])
```

row vector:

x = np.array([[1 , 1 ,1]])

col vector:
```
x = np.array([[ 1 ],
              [ 1 ],
              [ 1 ]])
```

BOTH

x = np.array([1 , 1 ,1])

Matrix multiplication:

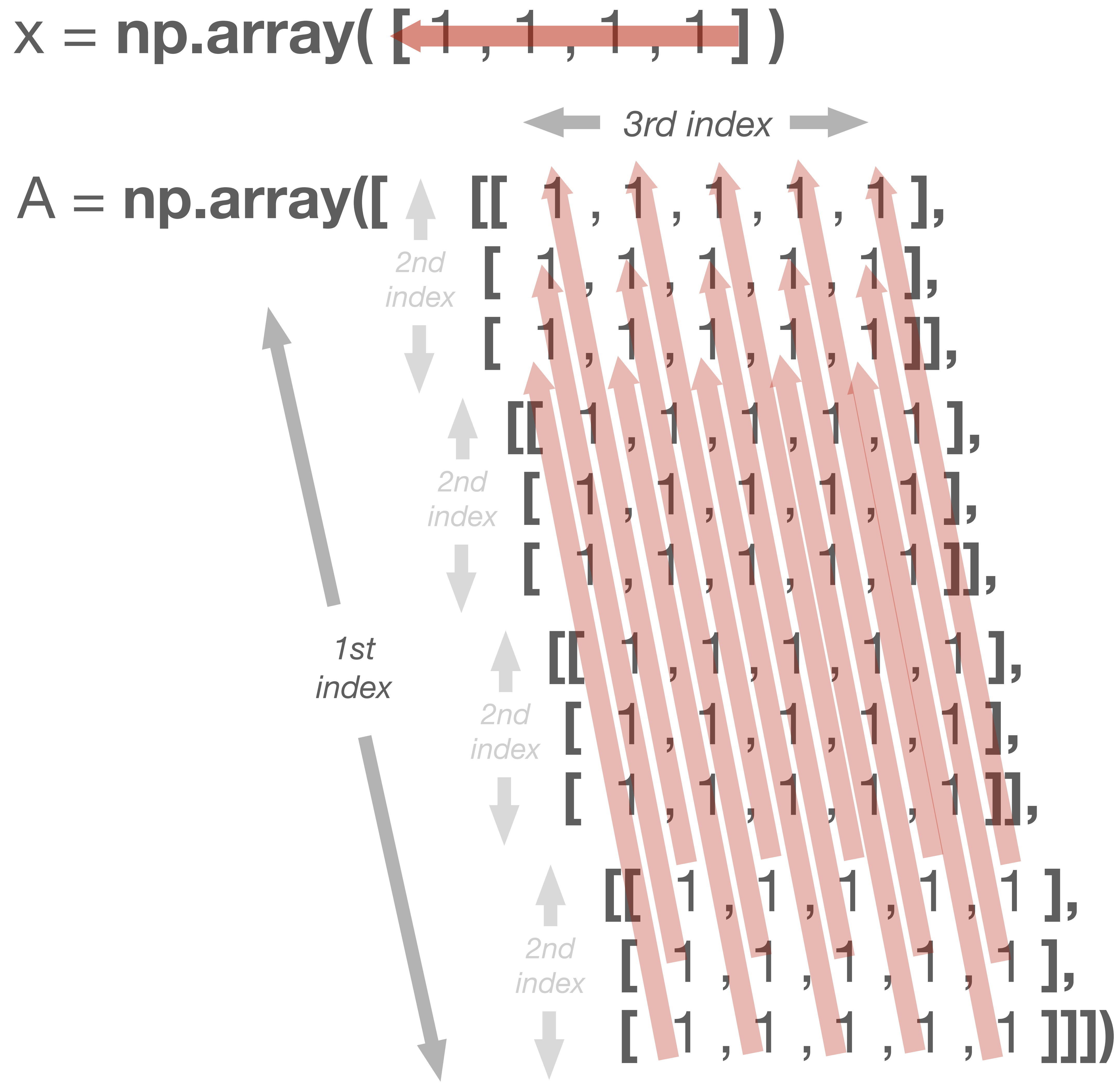
A@x = A.dot(x) = np.dot(A,x)

x = np.array([1,1,1])

A@x - A times col vector x
x@A - row vector x times A

Tranpose A.T

$A_{ijk}x_i$



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

A = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]]])
```

np.einsum(‘ijk,i’,A,x)

Python - Matrix Multiplication

vector: 1D `x = np.array([1,2,3])`

matrix 2D

```
A = np.array([[ 1, 1, 1 ],
              [ 1, 1, 1 ],
              [ 1, 1, 1 ]])
```

row vector:

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

col vector:

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

BOTH

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

Matrix multiplication:

A@x = A.dot(x) = np.dot(A,x)

x = np.array([1,1,1])

A@x - A times col vector x

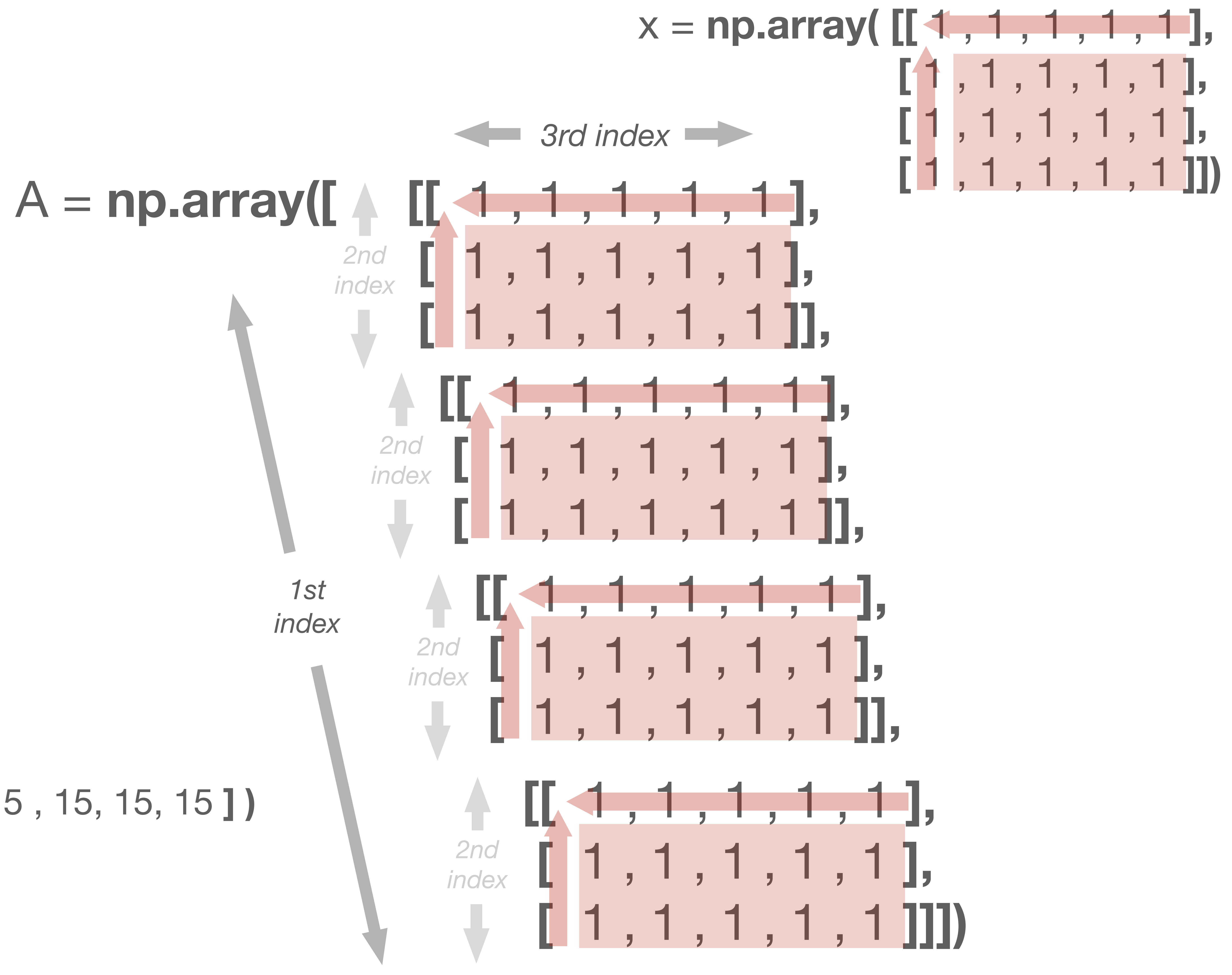
x@A - row vector x times A

RESULT:

y = **np.array([** 15 , 15, 15, 15 **])**

Tranpose A.T

$A_{ijk}x_{jk}$



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

```
A = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                 [ 1 , 1 , 1 , 1 , 1 ],
                 [ 1 , 1 , 1 , 1 , 1 ]],
                [[ 1 , 1 , 1 , 1 , 1 ],
                 [ 1 , 1 , 1 , 1 , 1 ],
                 [ 1 , 1 , 1 , 1 , 1 ]],
                [[ 1 , 1 , 1 , 1 , 1 ],
                 [ 1 , 1 , 1 , 1 , 1 ],
                 [ 1 , 1 , 1 , 1 , 1 ]],
                [[ 1 , 1 , 1 , 1 , 1 ],
                 [ 1 , 1 , 1 , 1 , 1 ],
                 [ 1 , 1 , 1 , 1 , 1 ]]])
```

np.einsum(‘ijk,jk’,A,x)

Python - Matrix Multiplication

vector: 1D x = np.array([1,2,3])

matrix 2D

```
A = np.array([[ 1, 1, 1 ],
              [ 1, 1, 1 ],
              [ 1, 1, 1 ]])
```

row vector:

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

col vector:

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

BOTH

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

Matrix multiplication:

A@x = A.dot(x) = np.dot(A,x)

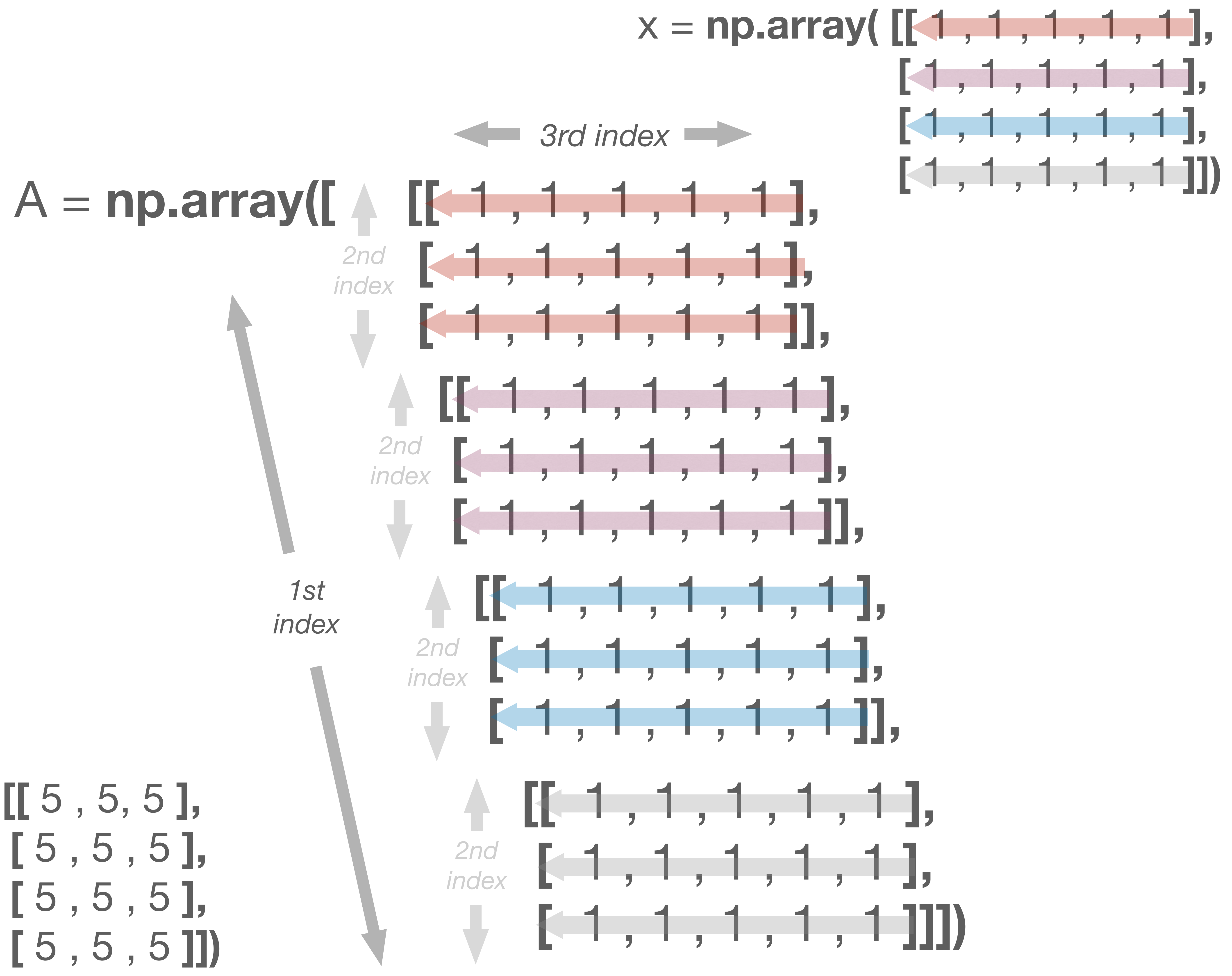
x = np.array([1,1,1])

A@x - A times col vector x
x@A - row vector x times A

Tranpose A.T

RESULT:

```
y = np.array( [[ 5 , 5, 5 ],
               [ 5 , 5 , 5 ],
               [ 5 , 5 , 5 ],
               [ 5 , 5 , 5 ]])
```



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

```
A = np.array([ [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]],
               [[ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ],
                [ 1 , 1 , 1 , 1 , 1 ]]])
```

np.einsum(‘ijk,ik->ij’,A,x)