

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 :	<code>A = np.array([[1, 2, 3], [3, 2, 1], [2, 1, 3]])</code>	
<code>A[np.newaxis,:]</code>	<code>A[:,np.newaxis]</code>	<code>np.eye(n)</code>
<code>np.stack([x,x,x])</code>	...stack along new axis	<code>np.ones([m,n])</code>
<code>np.vstack([x,x,x])</code>	...stack vertically	<code>np.zeros([m,n])</code>
<code>np.hstack([x,x,x])</code>	...stack horizontally	<code>np.arange(start,stop,step=1)</code>
<code>np.block([[A,B] [C,D]])</code>	...block matrix	<code>np.arange(start,stop,num=50)</code>
<code>np.append(A,newarray)</code>	...adds newarray to the end	
<code>np.insert(A,index,newarray)</code>	...adds new array at index	
<code>np.reshape(A,newshape)</code>	...cycle through deepest axes first	
<code>np.concatenate((A,B,C),axis=0)</code>	...must have same shape except along axis	
<code>np.flip(A,axis=None)</code>	...by default flips all axes	
<code>np.where(A,axis=None)</code>		

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])

index 0 1 2 3 n-1 or -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])

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])

x[:4]

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

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

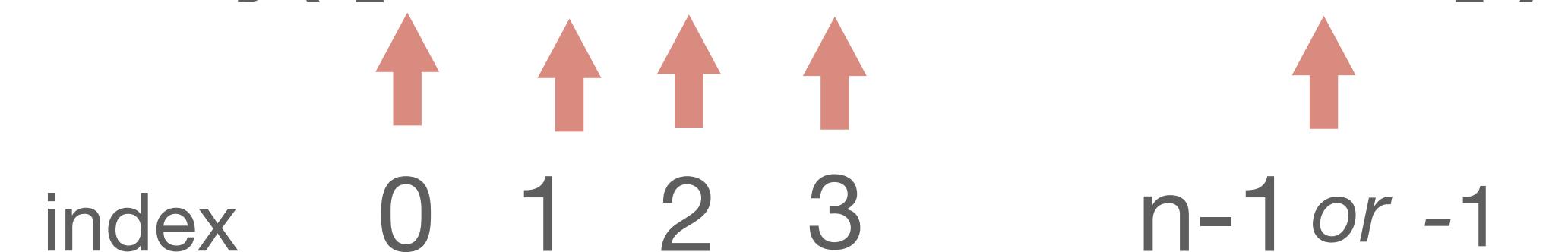
boolean indexing

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]

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

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

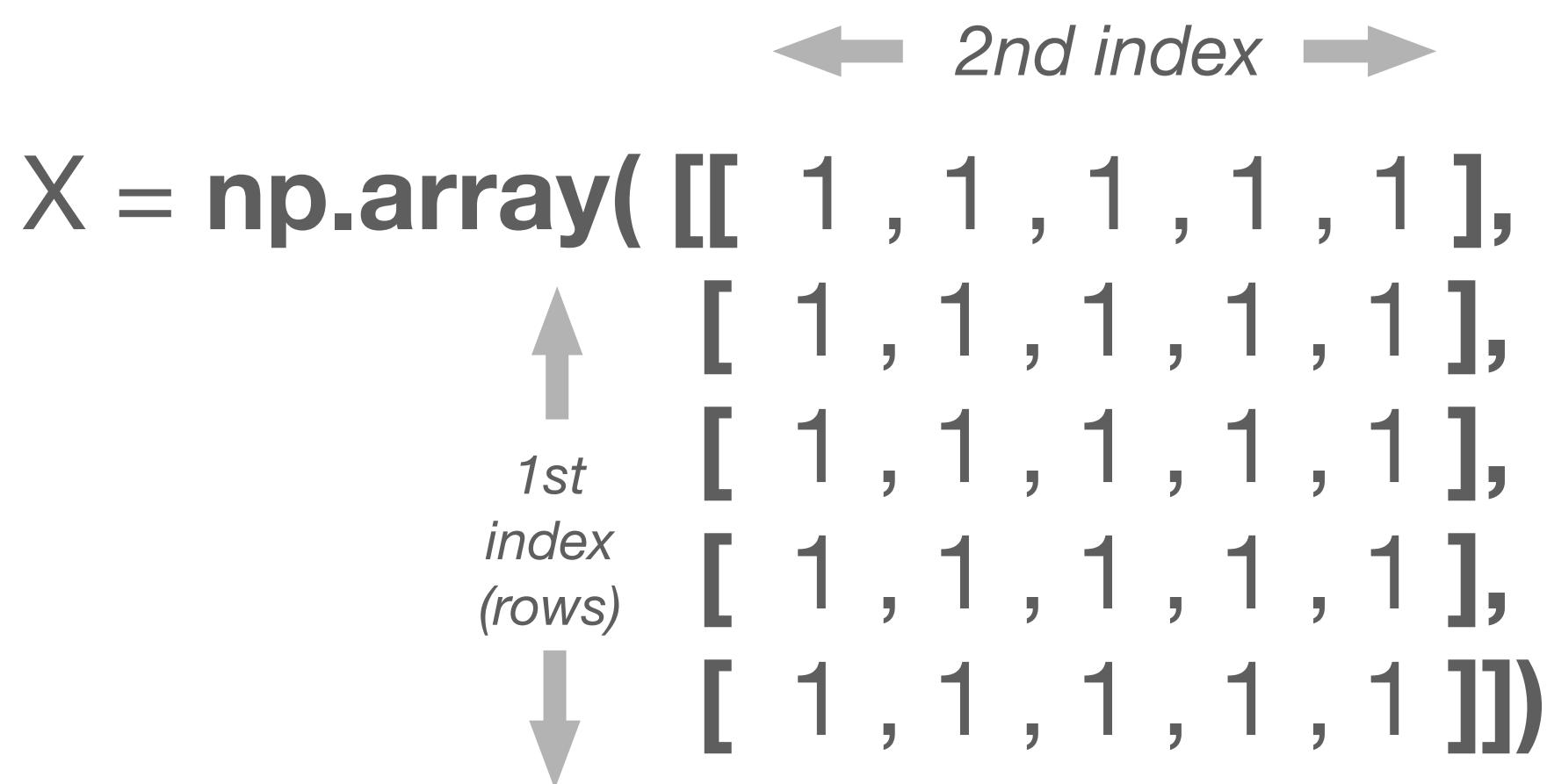
x[bool] - returns 0,1, and 3 element.

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[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

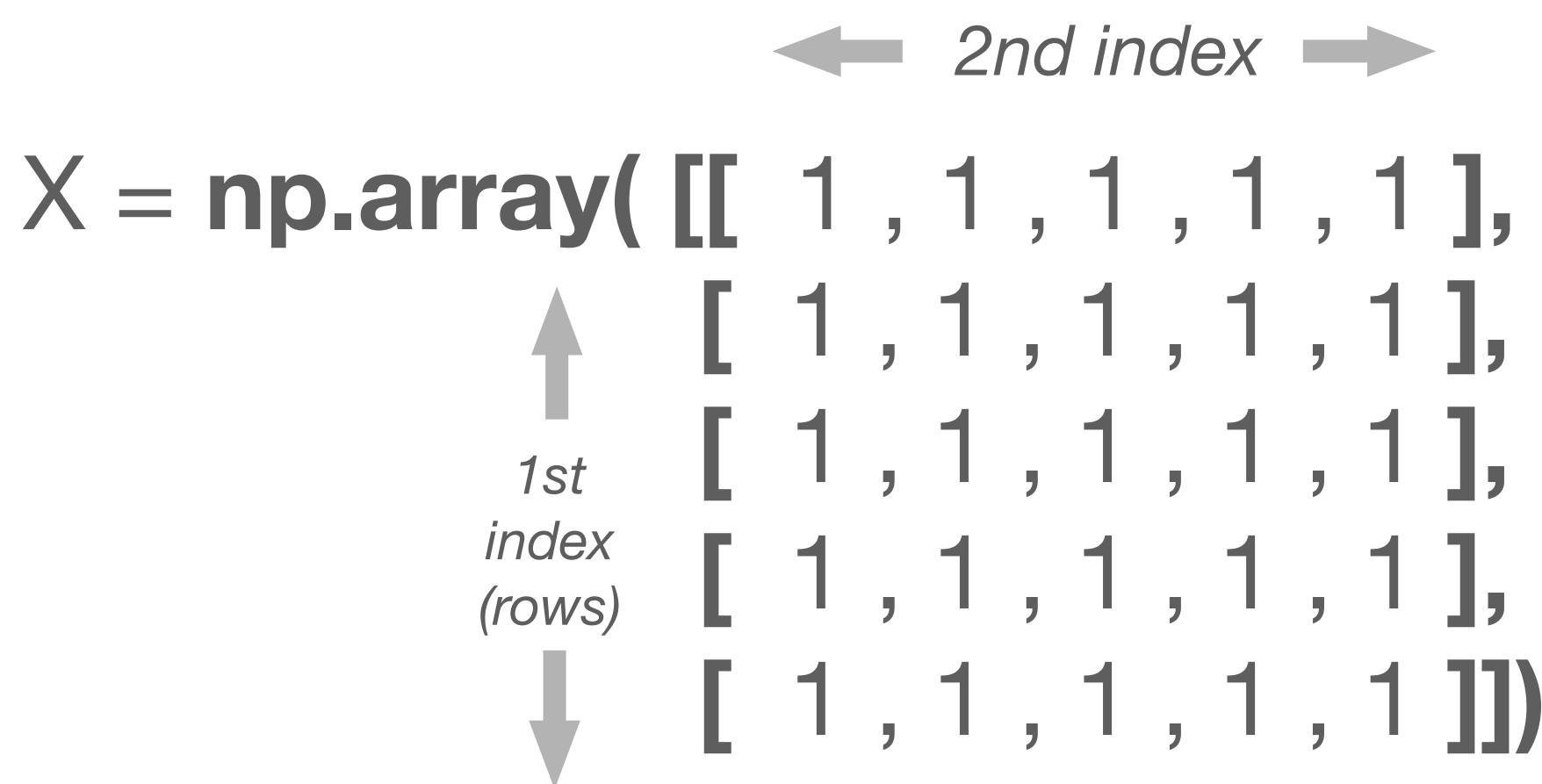
x[bool] - returns 0,1, and 3 element.

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



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

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

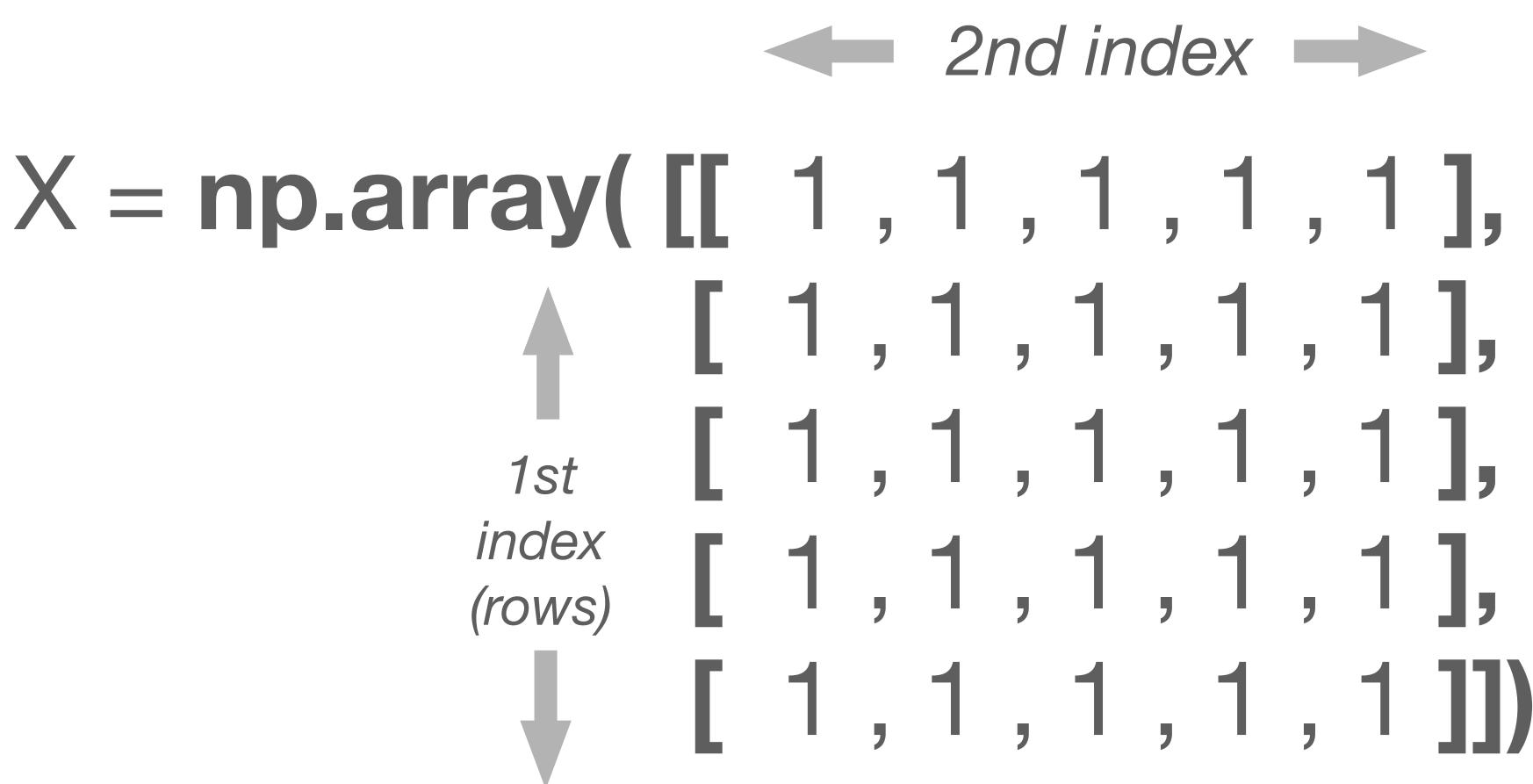
x[bool] - returns 0,1, and 3 element.

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]

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]])

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

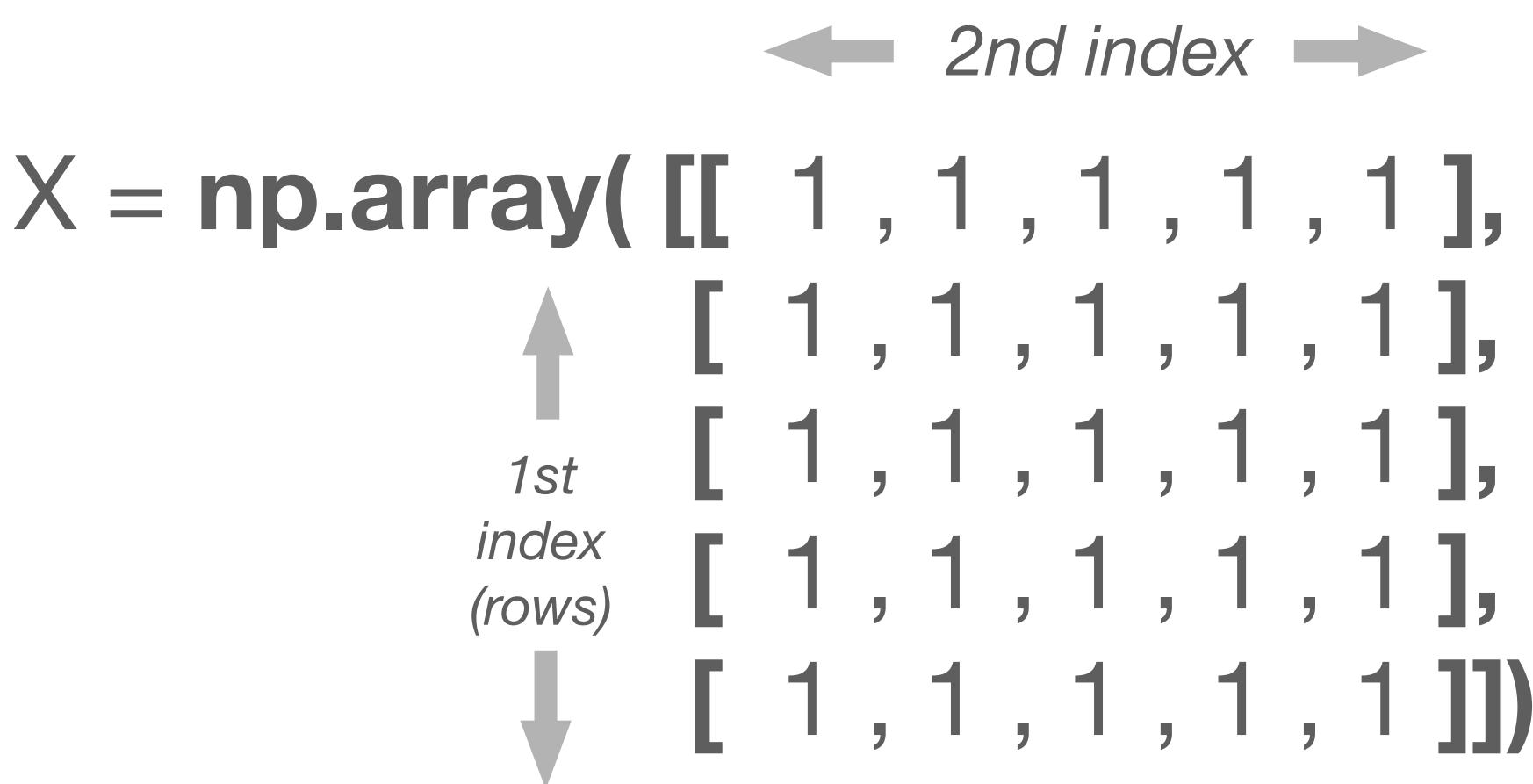
x[bool] - returns 0,1, and 3 element.

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]

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]])

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

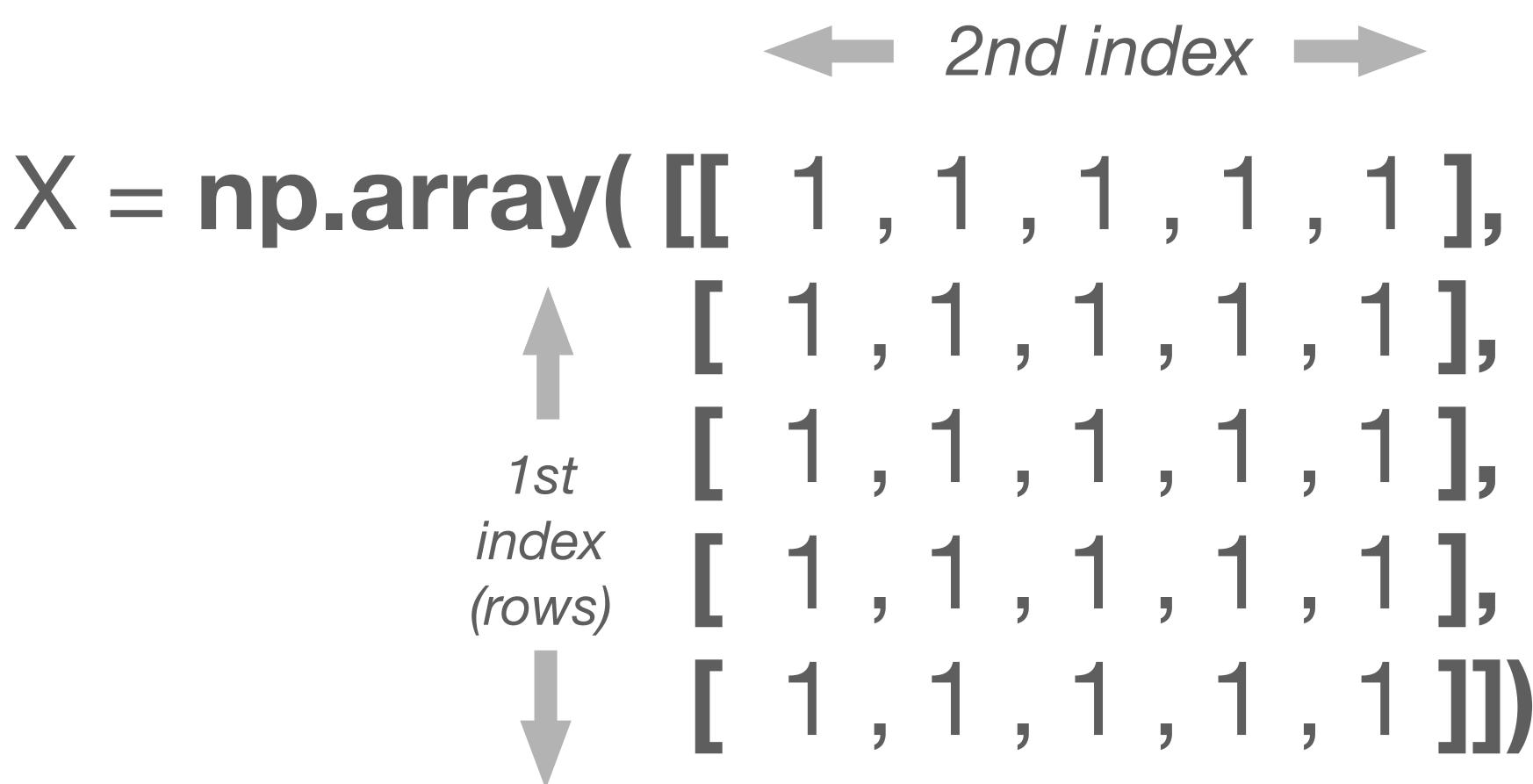
x[bool] - returns 0,1, and 3 element.

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]

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

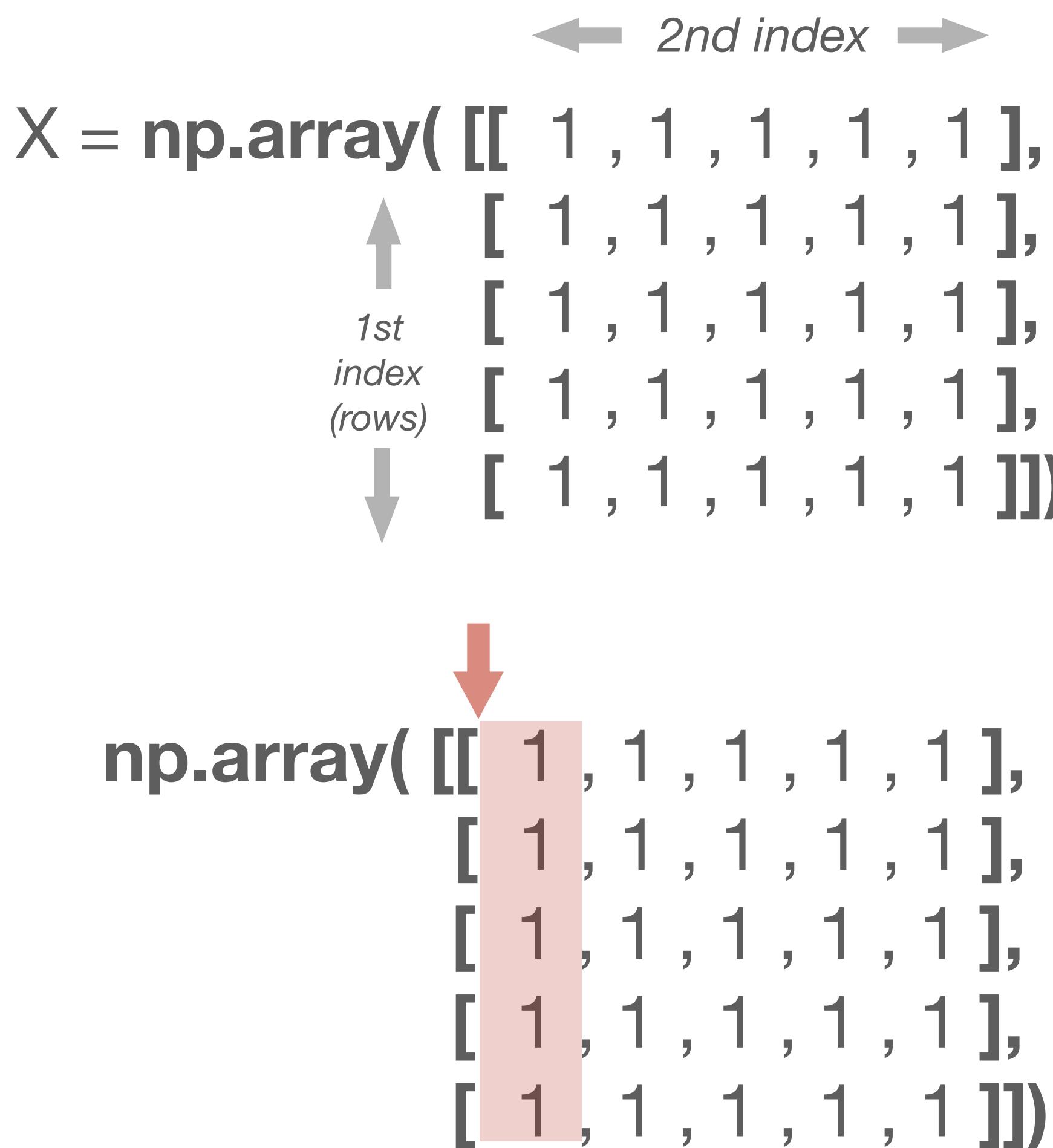
x[bool] - returns 0,1, and 3 element.

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



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 , 1 , 1 , 1 , 1 , 1]])

X[:,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

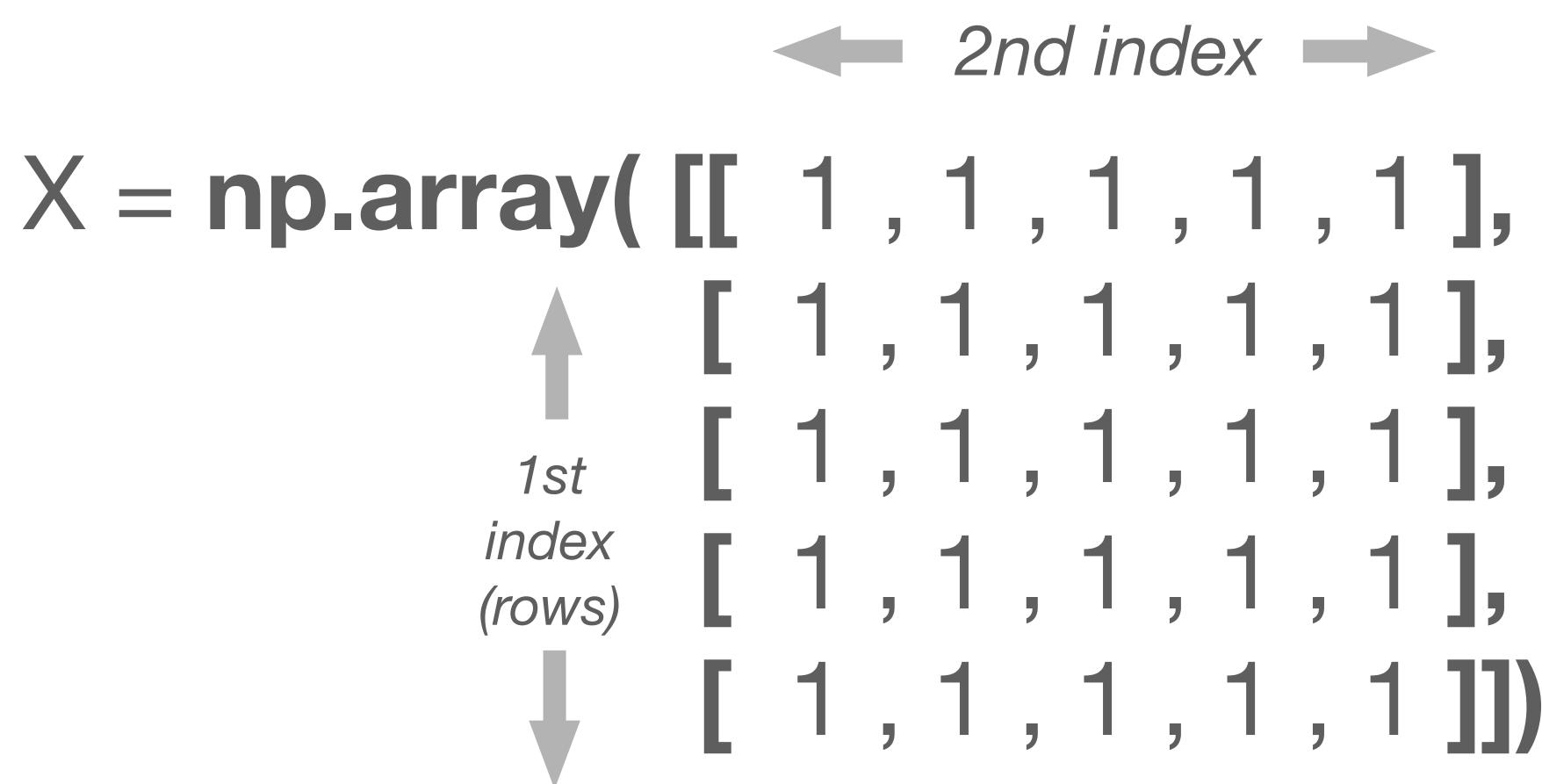
x[bool] - returns 0,1, and 3 element.

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



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]])

X[:,3]

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

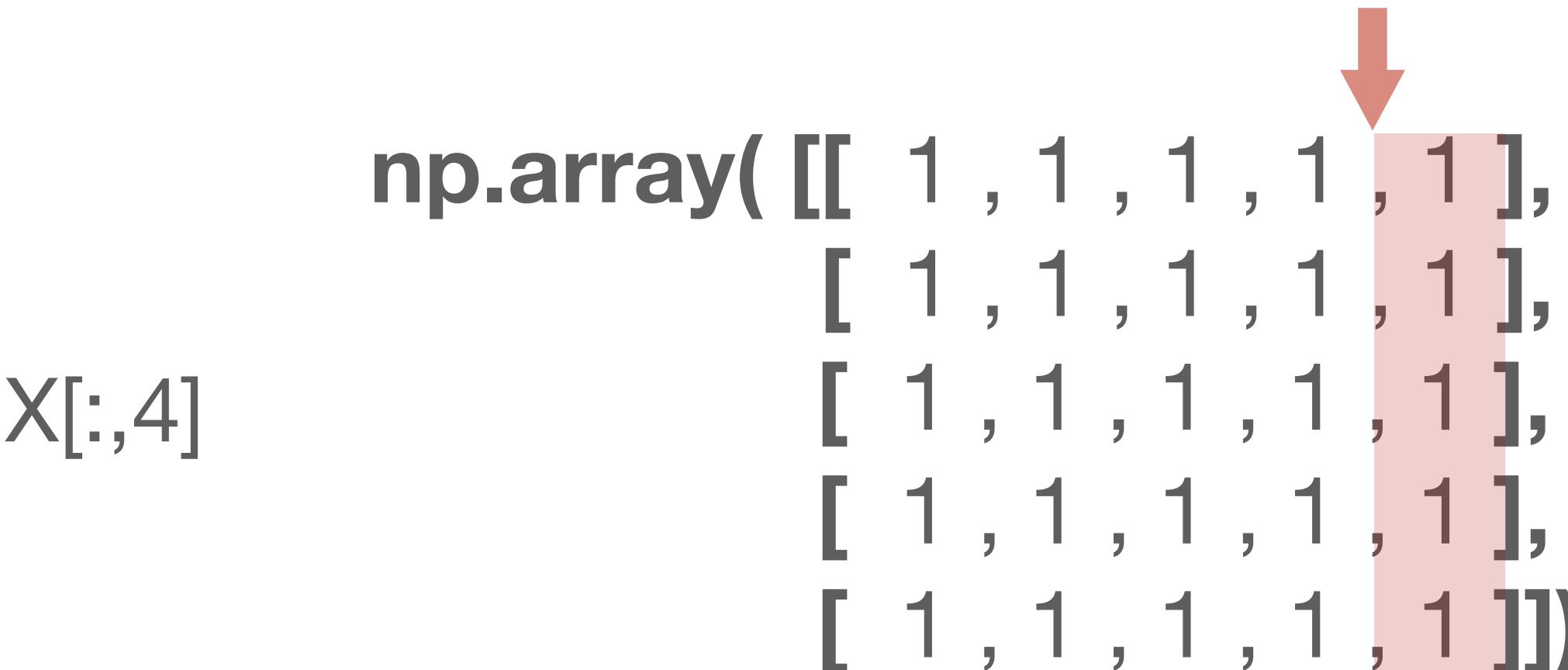
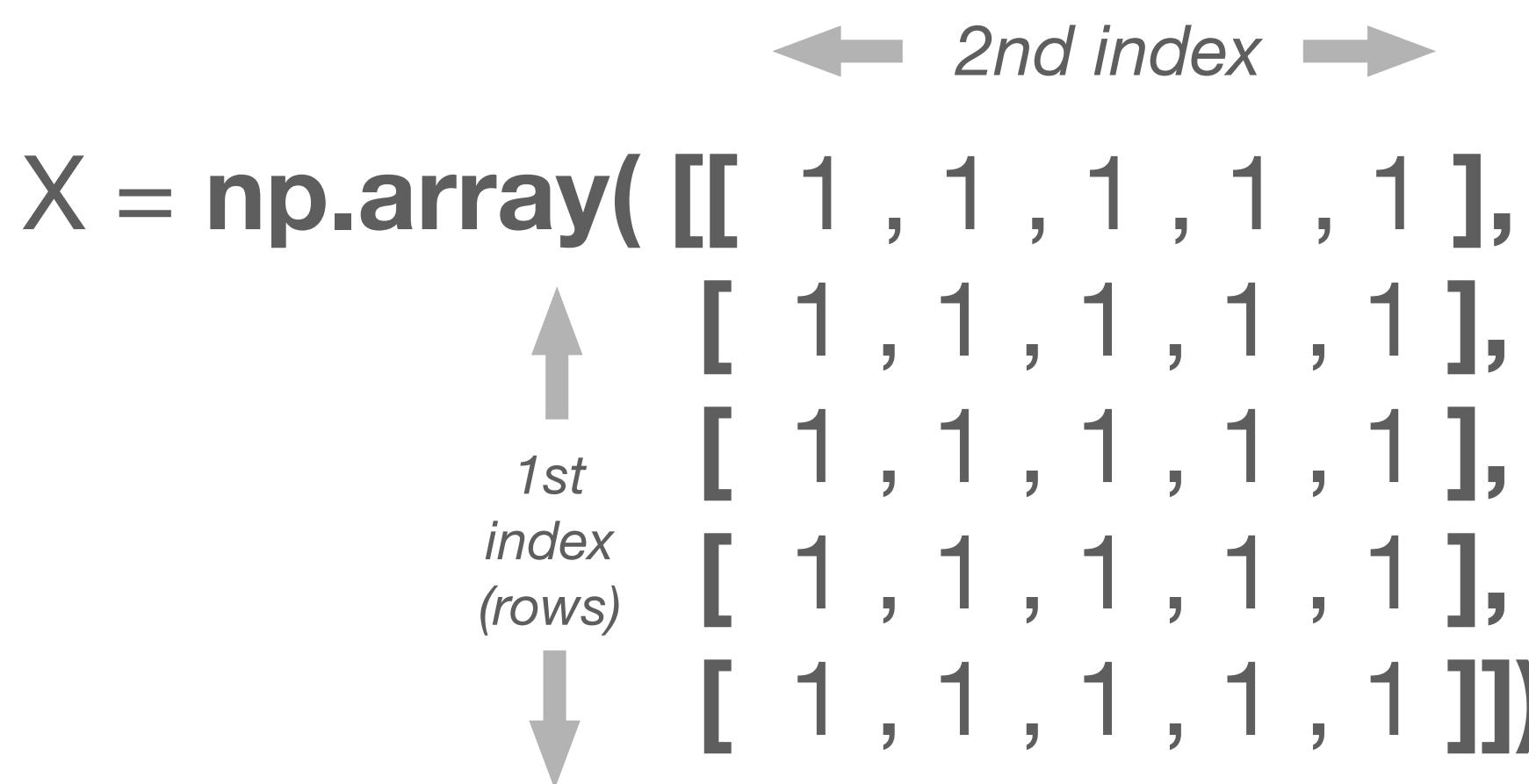
x[bool] - returns 0,1, and 3 element.

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



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

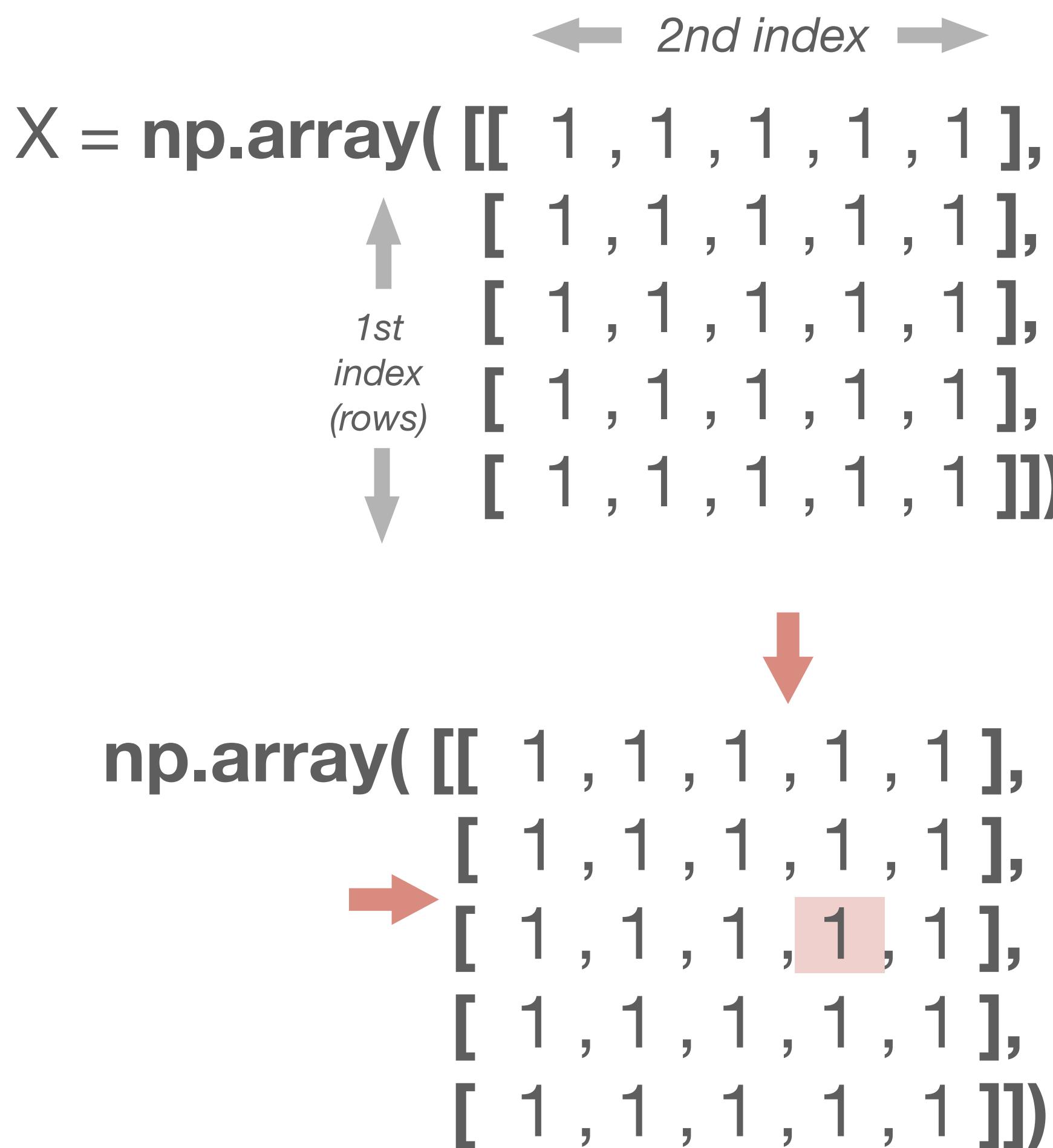
x[bool] - returns 0,1, and 3 element.

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



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

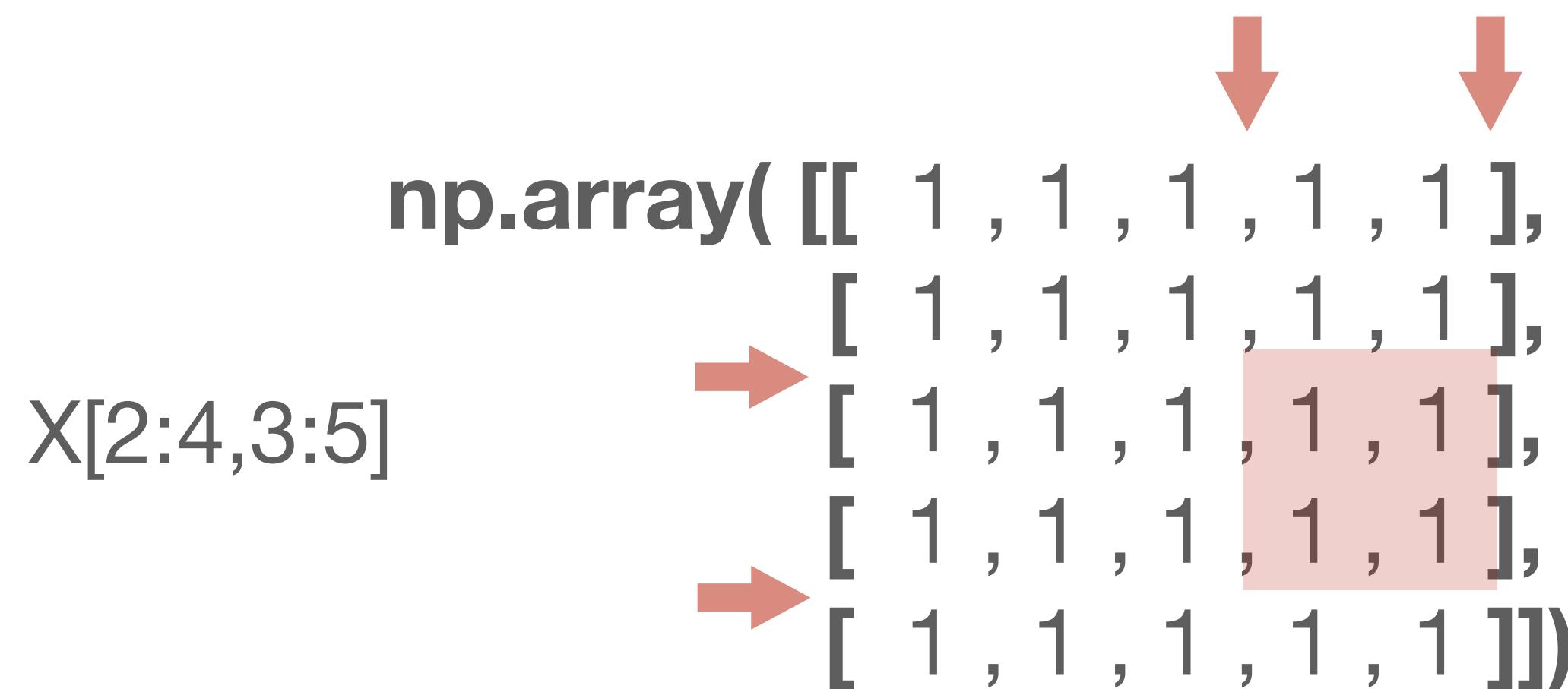
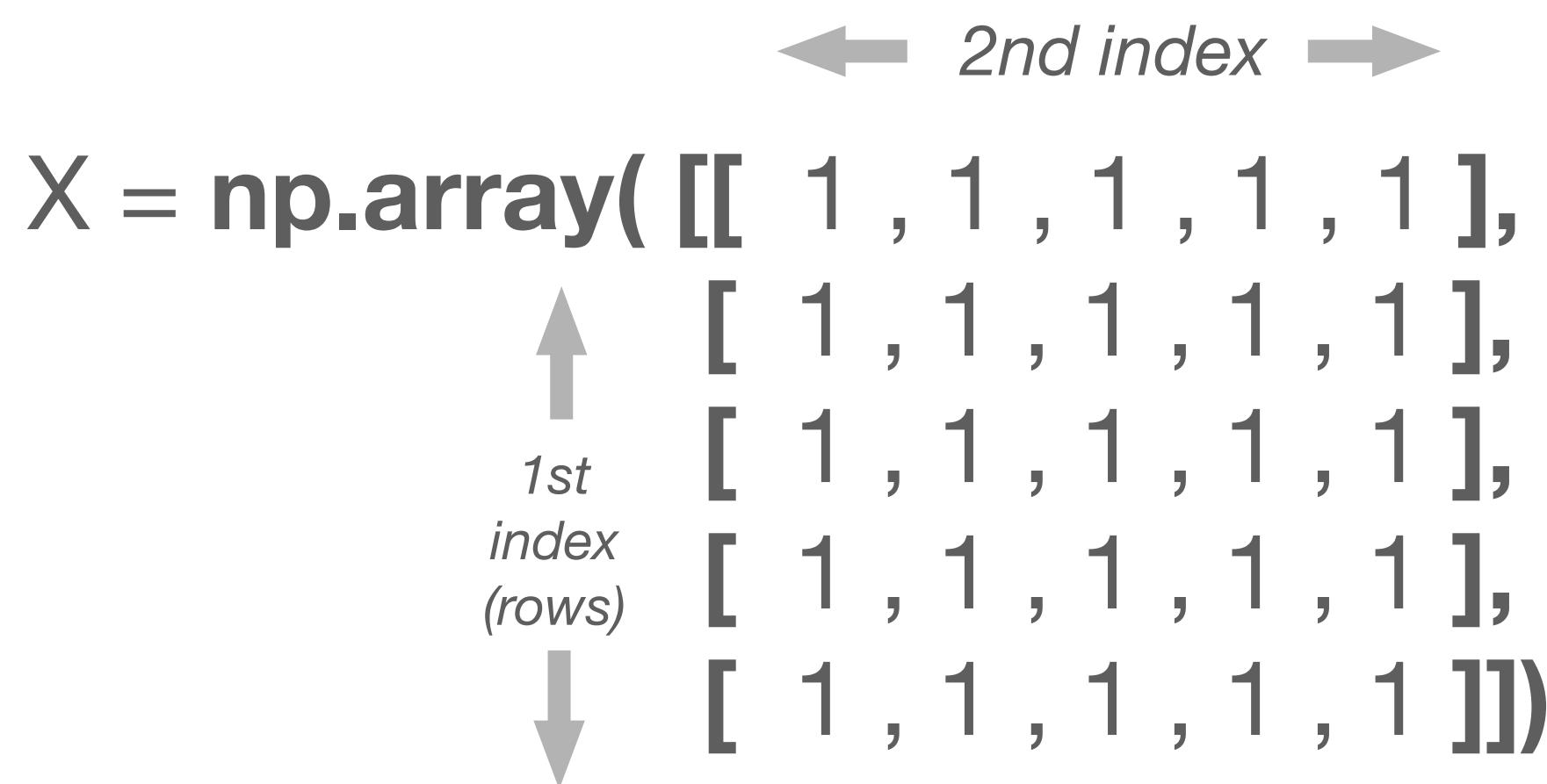
x[bool] - returns 0,1, and 3 element.

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



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

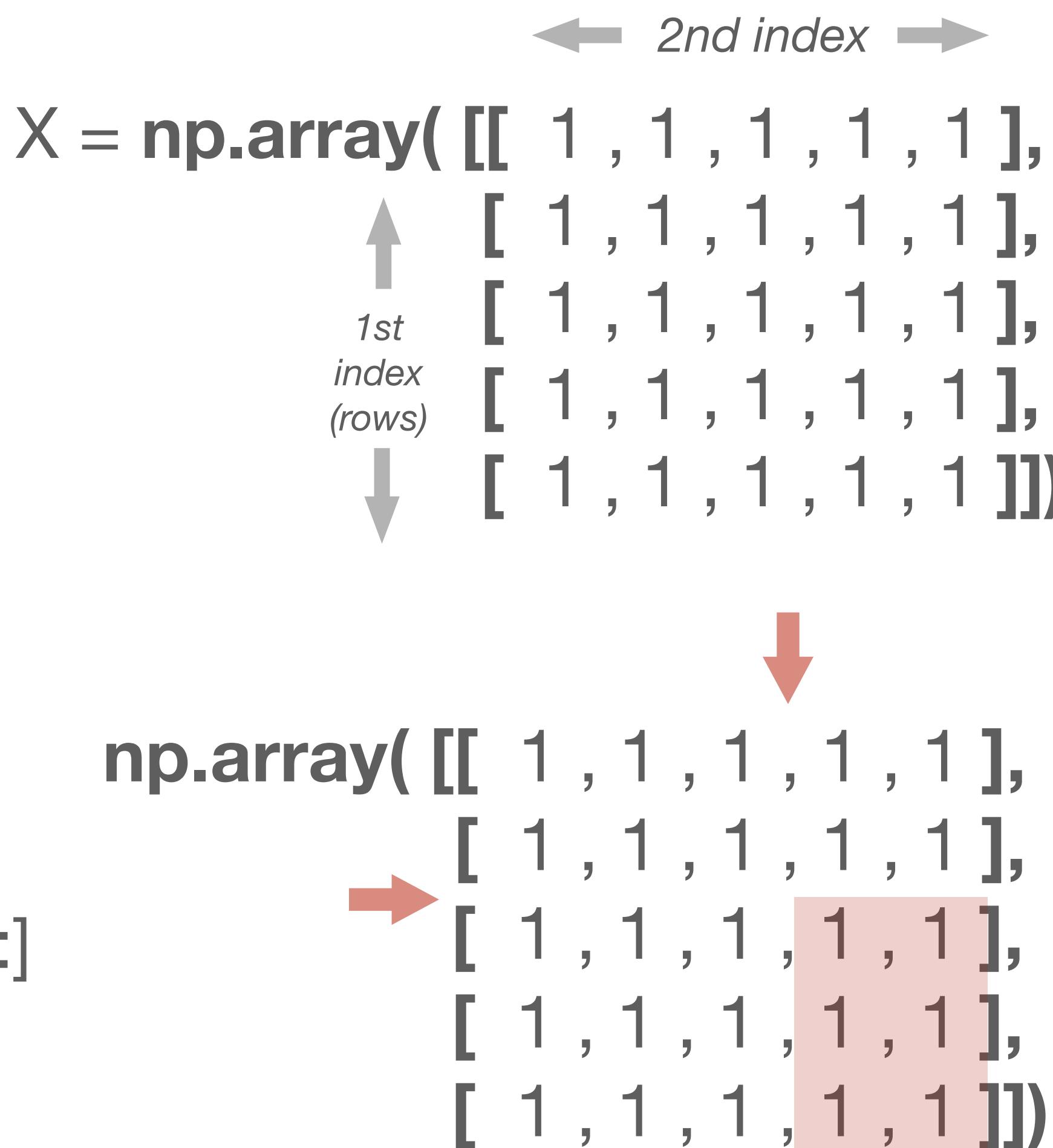
x[bool] - returns 0,1, and 3 element.

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



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

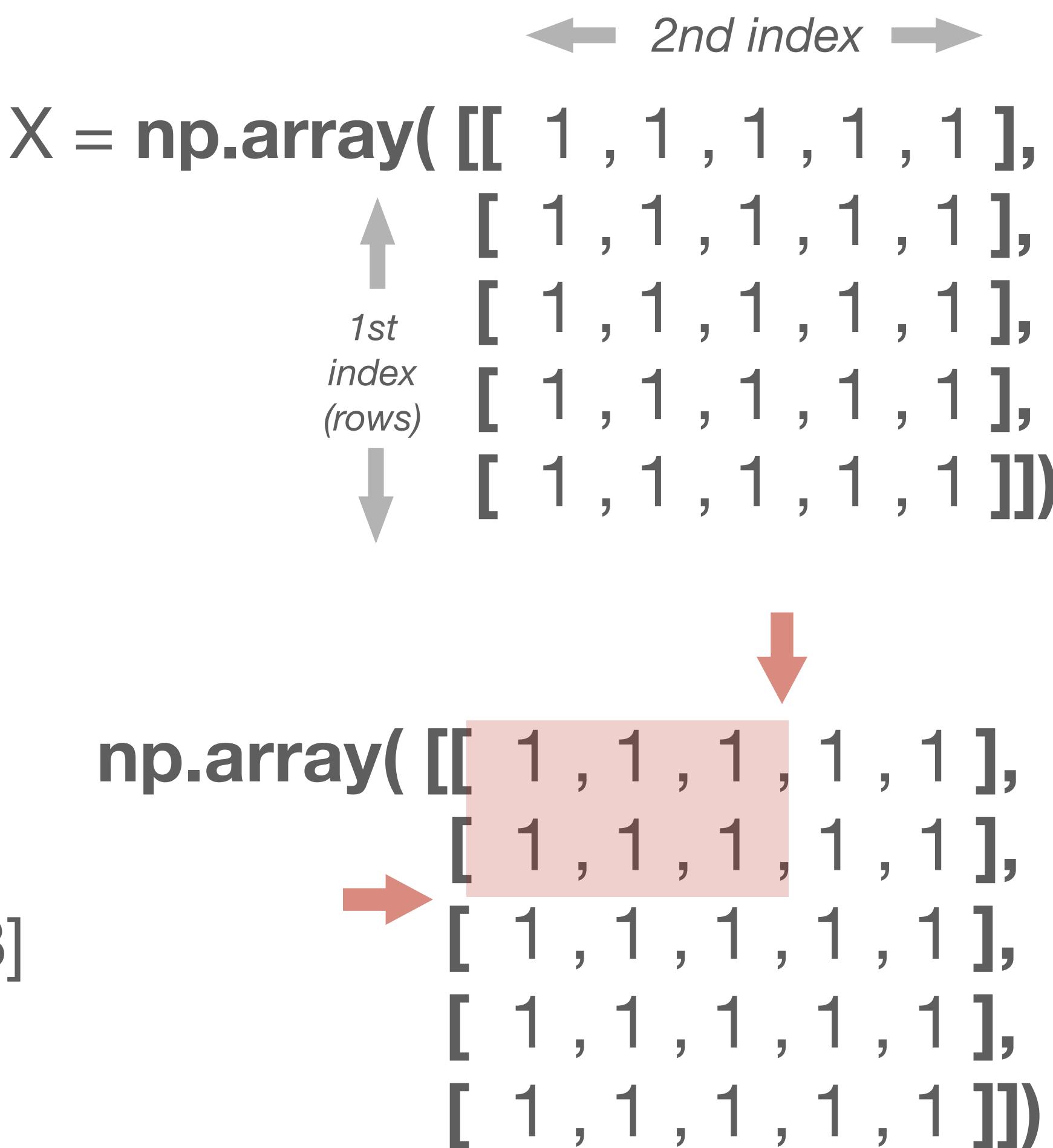
x[bool] - returns 0,1, and 3 element.

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



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

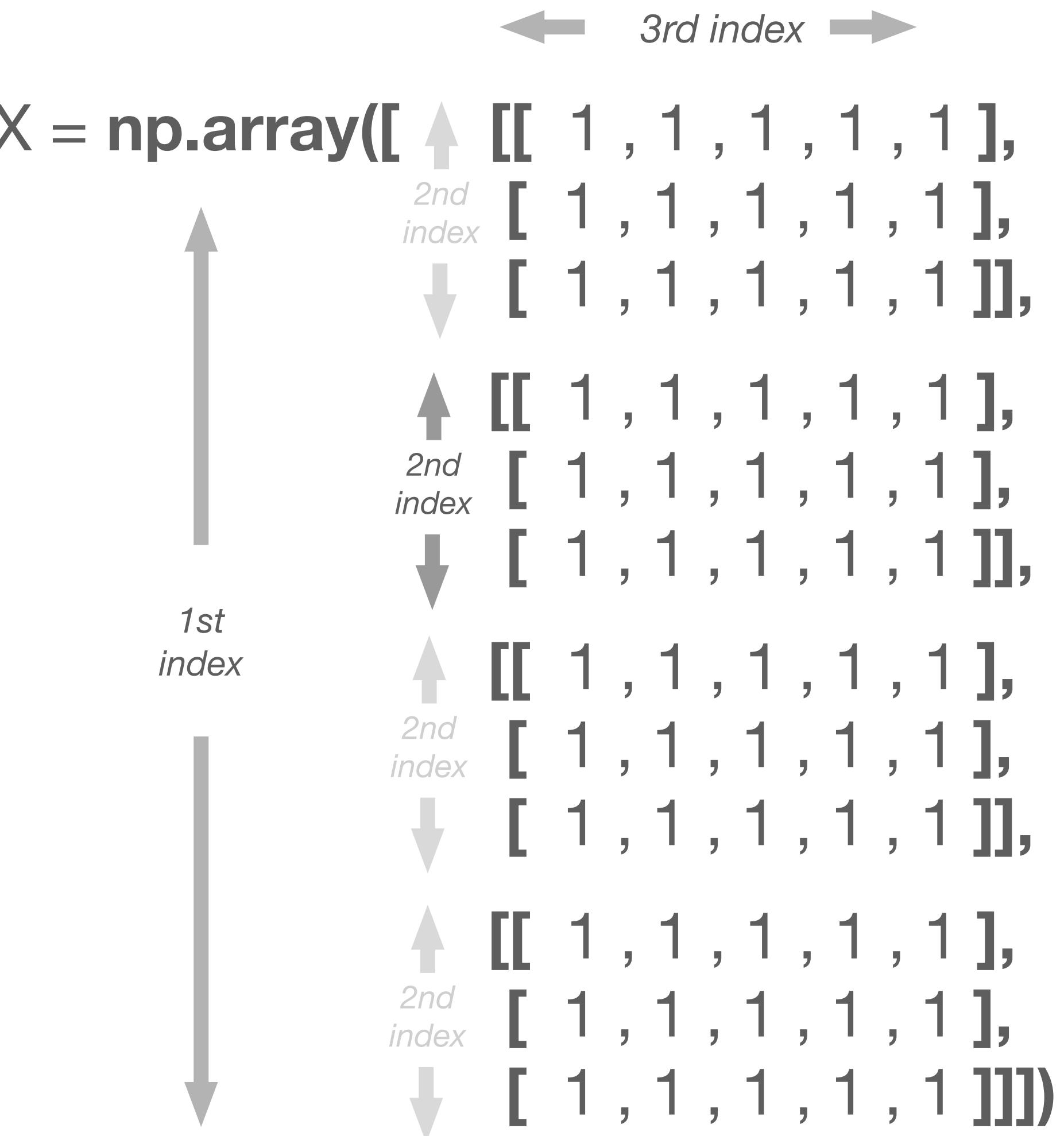
x[bool] - returns 0,1, and 3 element.

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



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

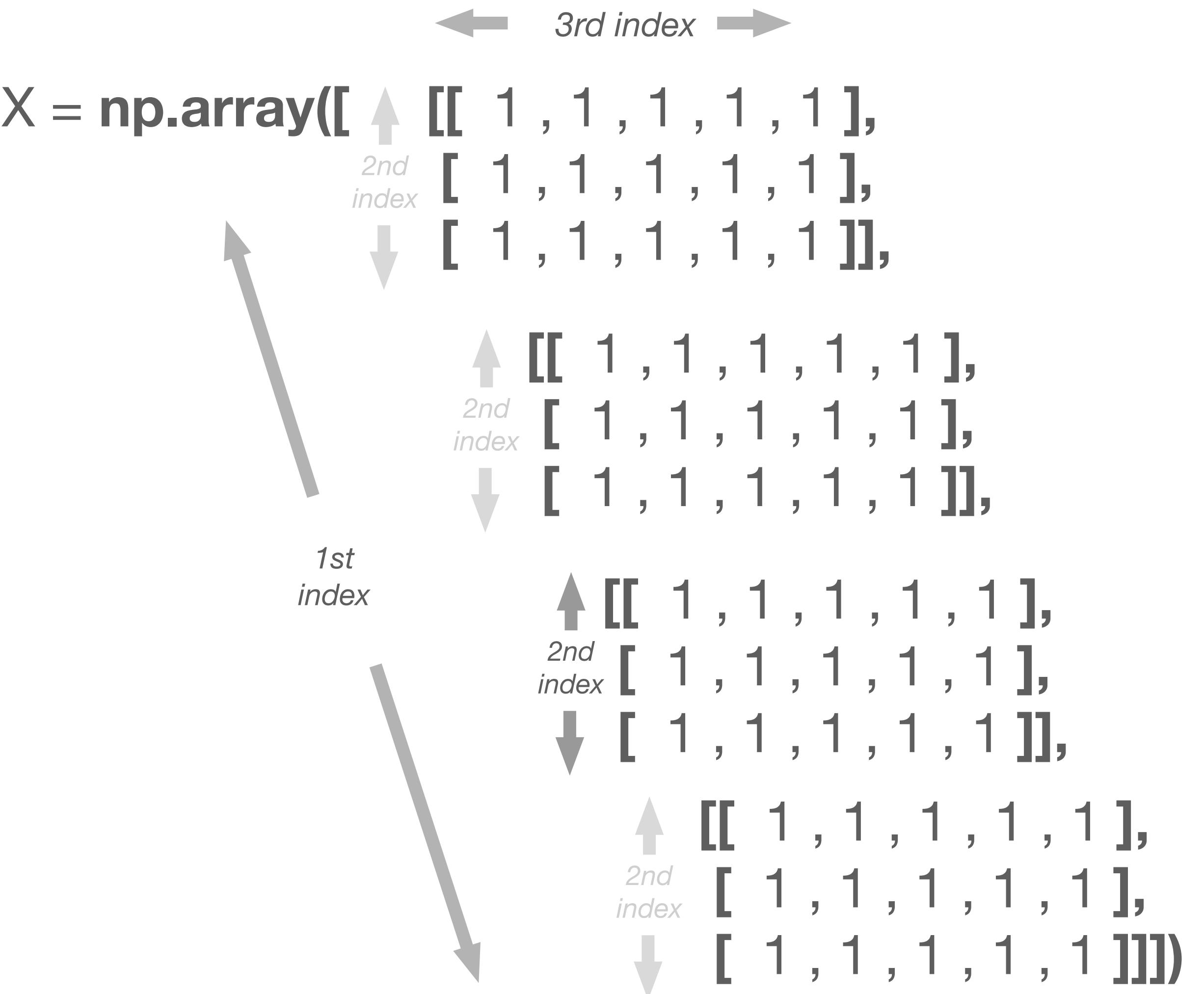
x[bool] - returns 0,1, and 3 element.

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



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

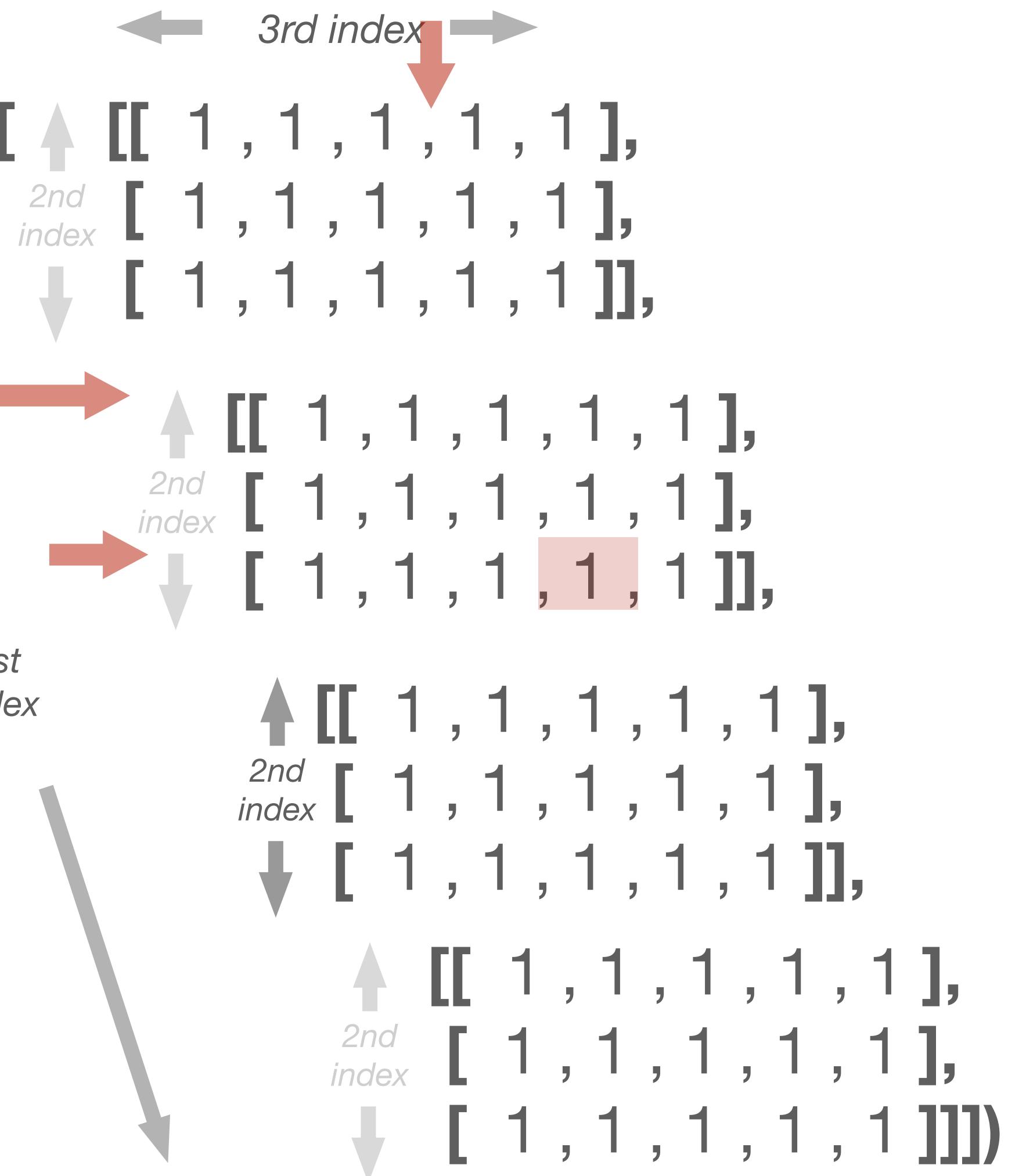
bool = [True, True, False, True]; MUST BE ARRAY LENGTH

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

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

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]])



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];

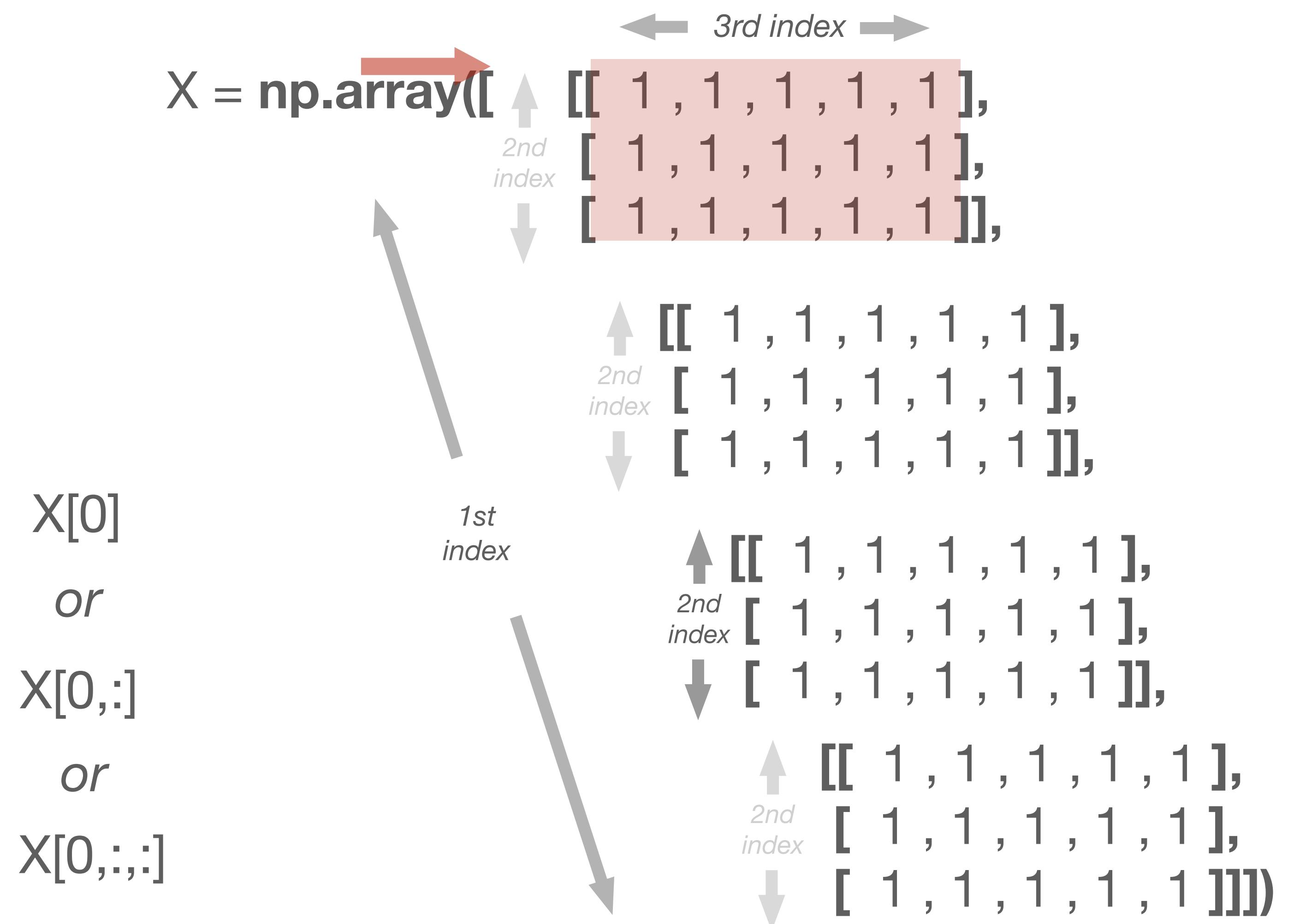
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

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

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

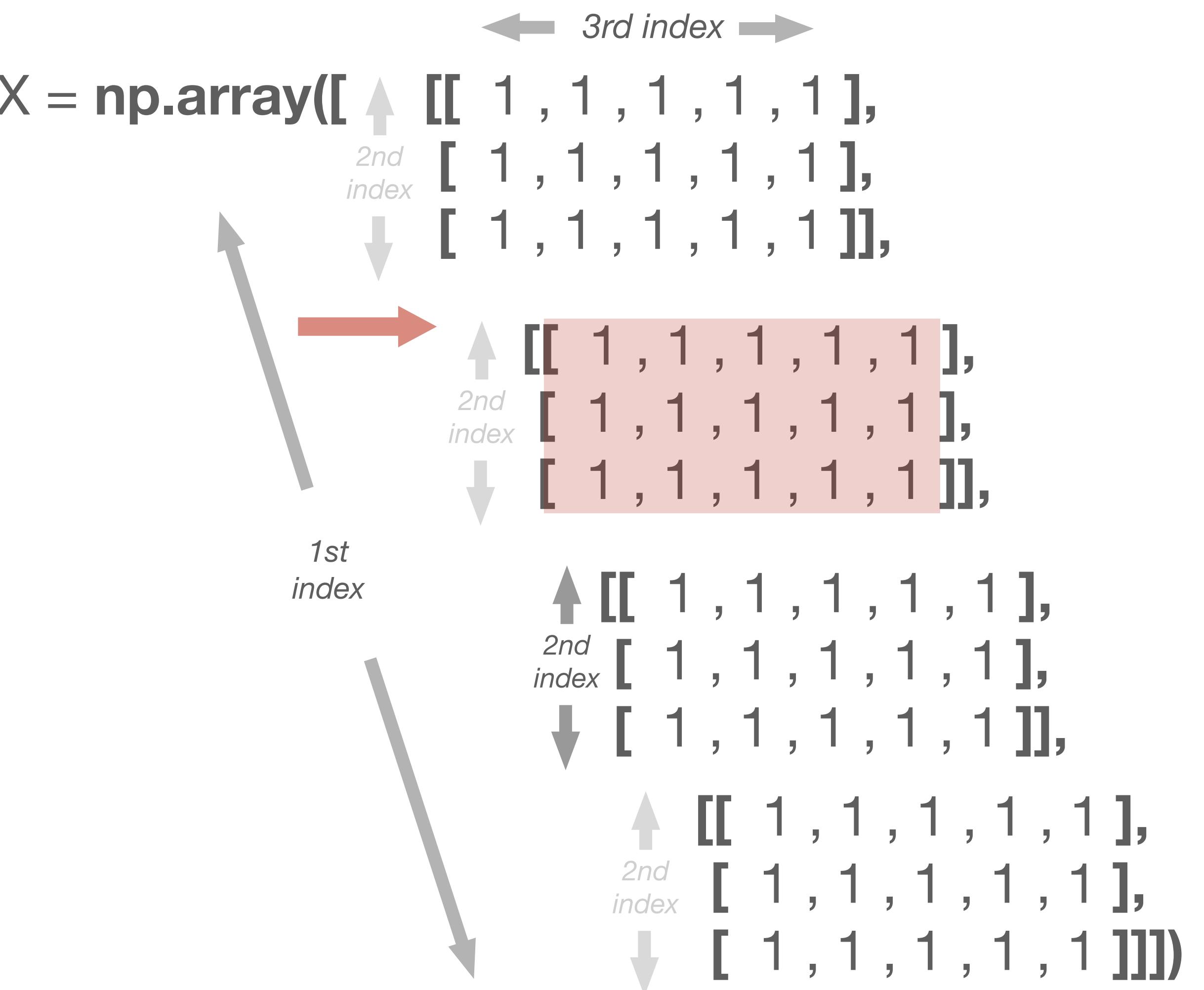
boolean indexing

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]

X[1]



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

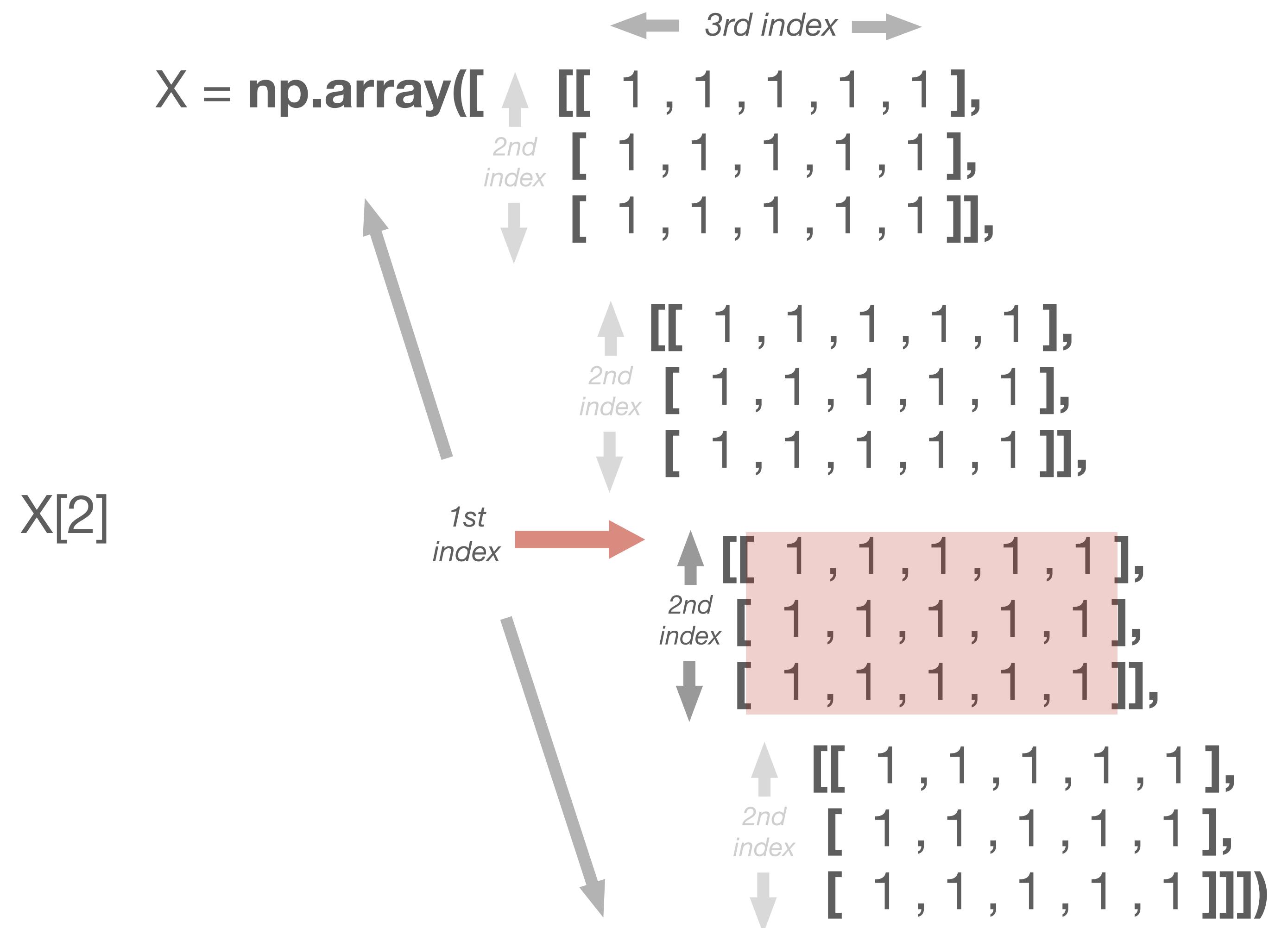
`x[bool]` - returns 0-1- and 3 element.

`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



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.

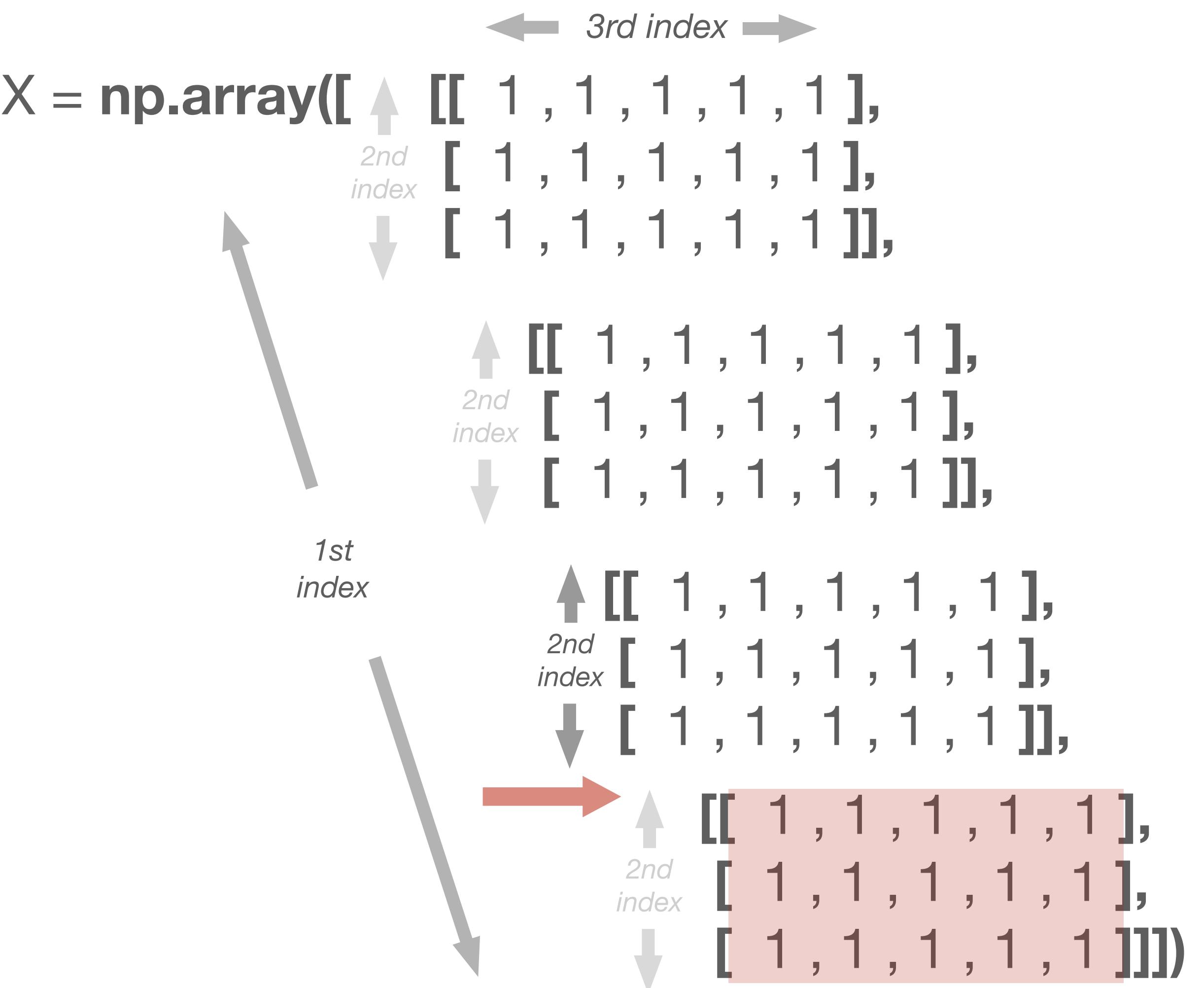
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

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]



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

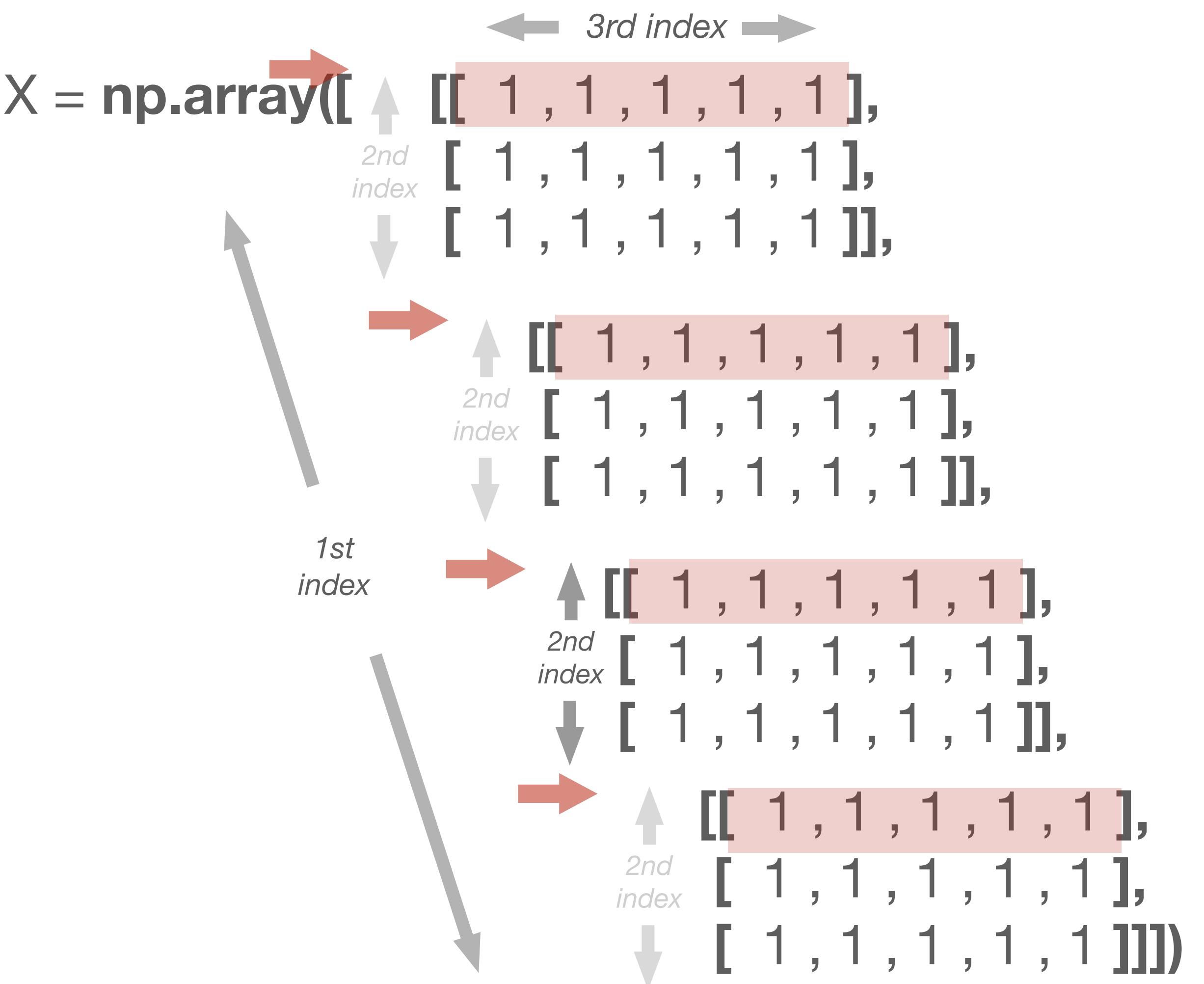
boolean indexing

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]

X[:,0]



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

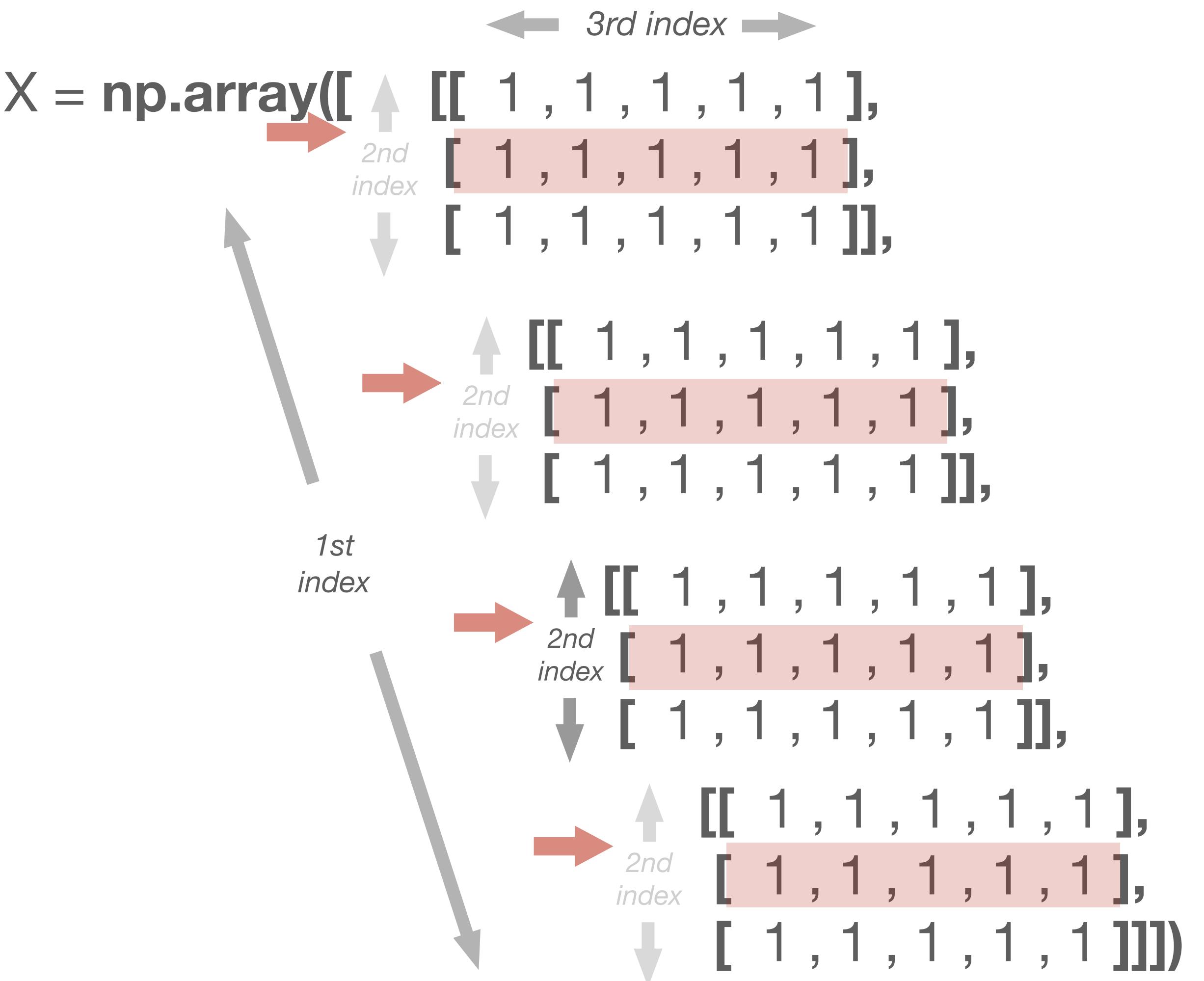
boolean indexing

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]

X[:,1]



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

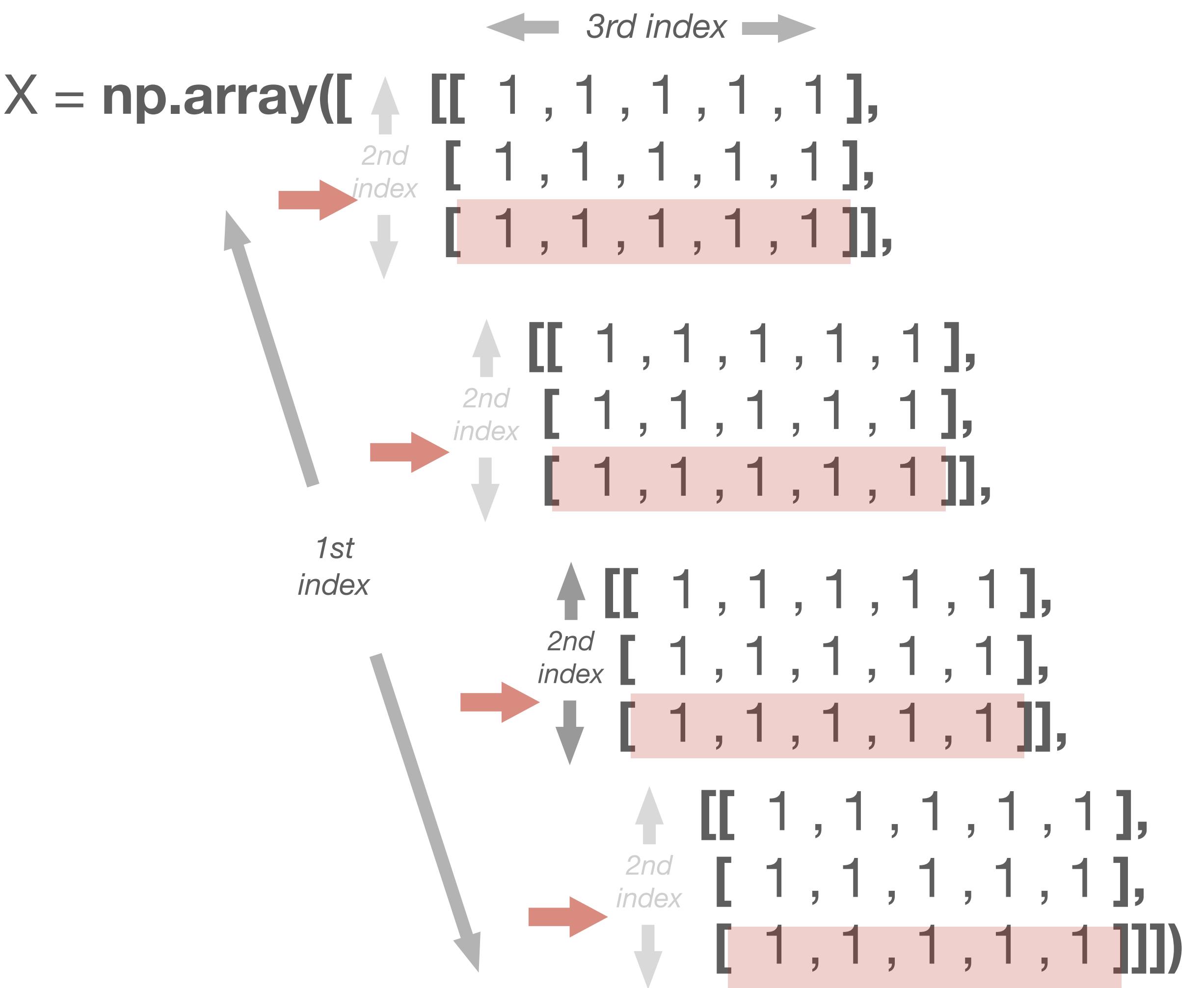
boolean indexing

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]

X[:,2]



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

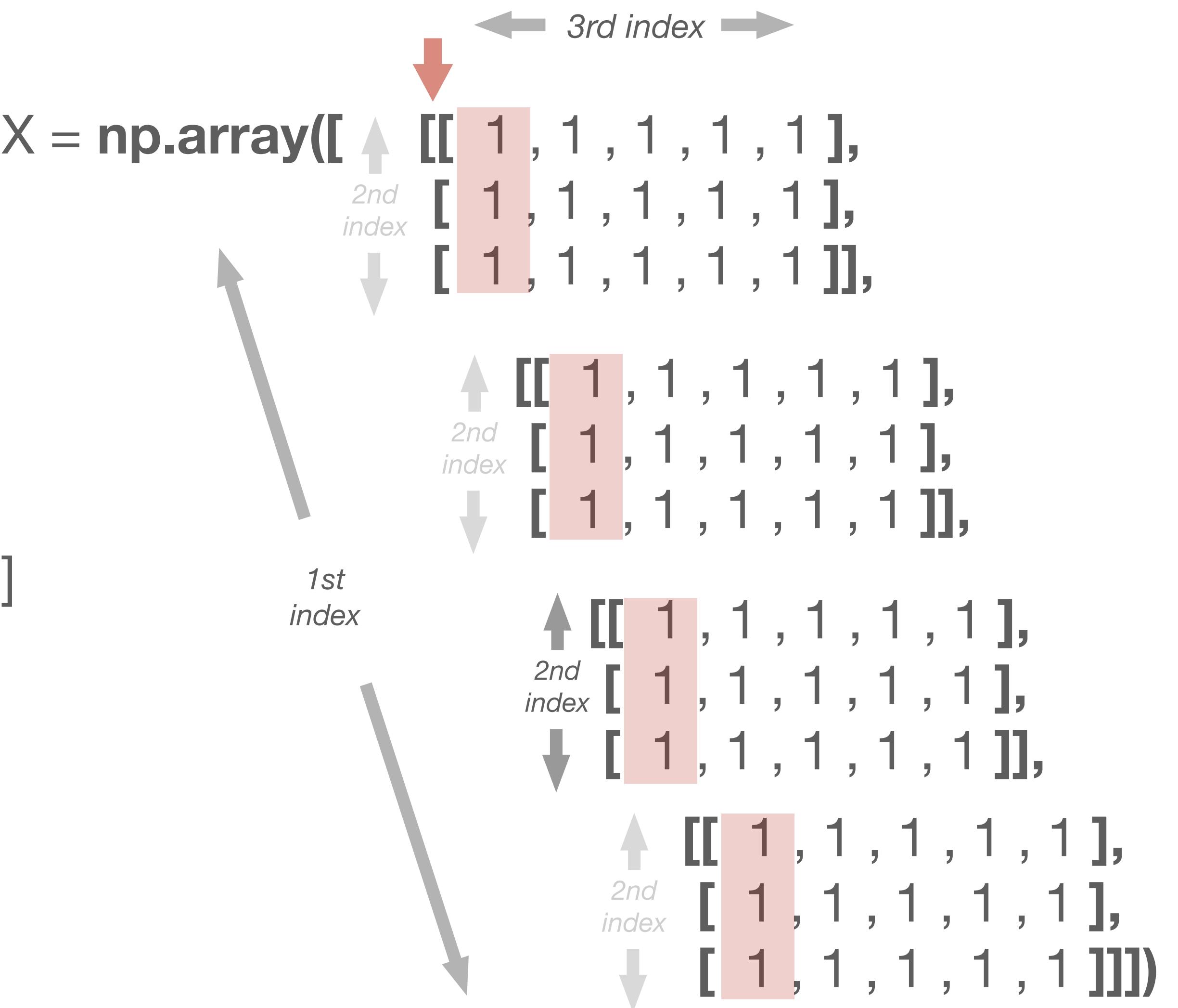
boolean indexing

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]

X[:, :, 0]



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

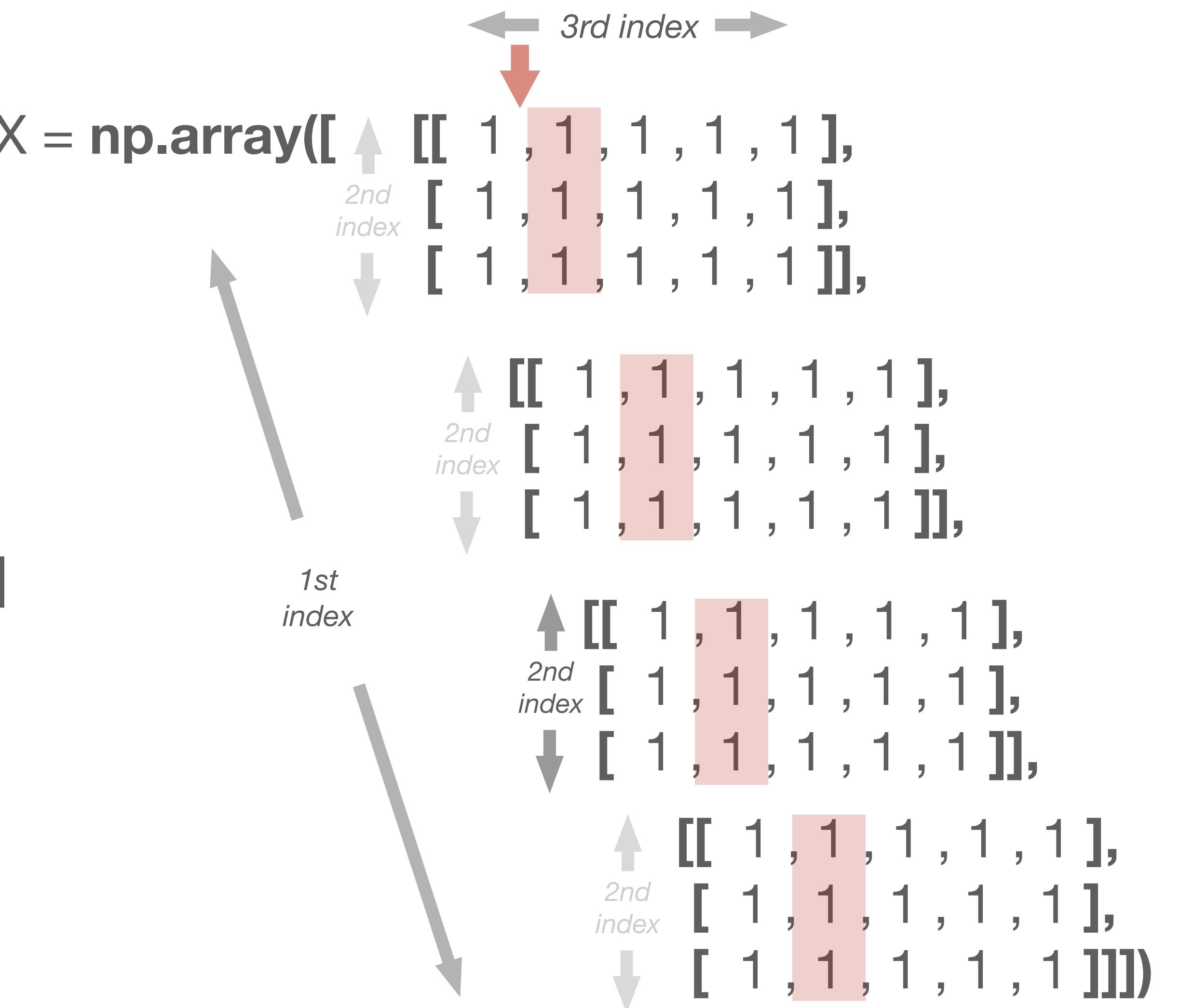
boolean indexing

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]

X[:, :, 1]



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

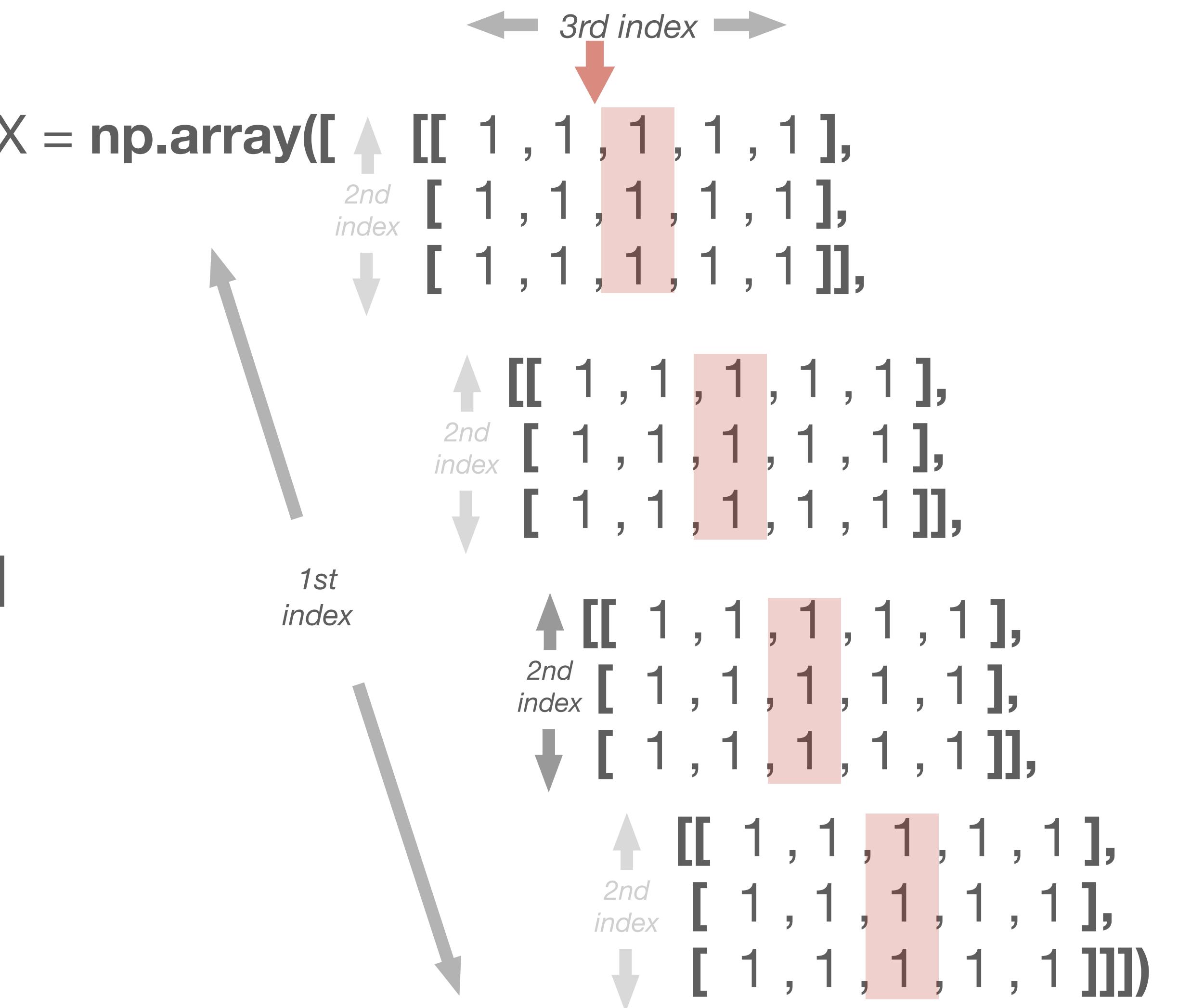
boolean indexing

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]

X[:, :, 2]



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

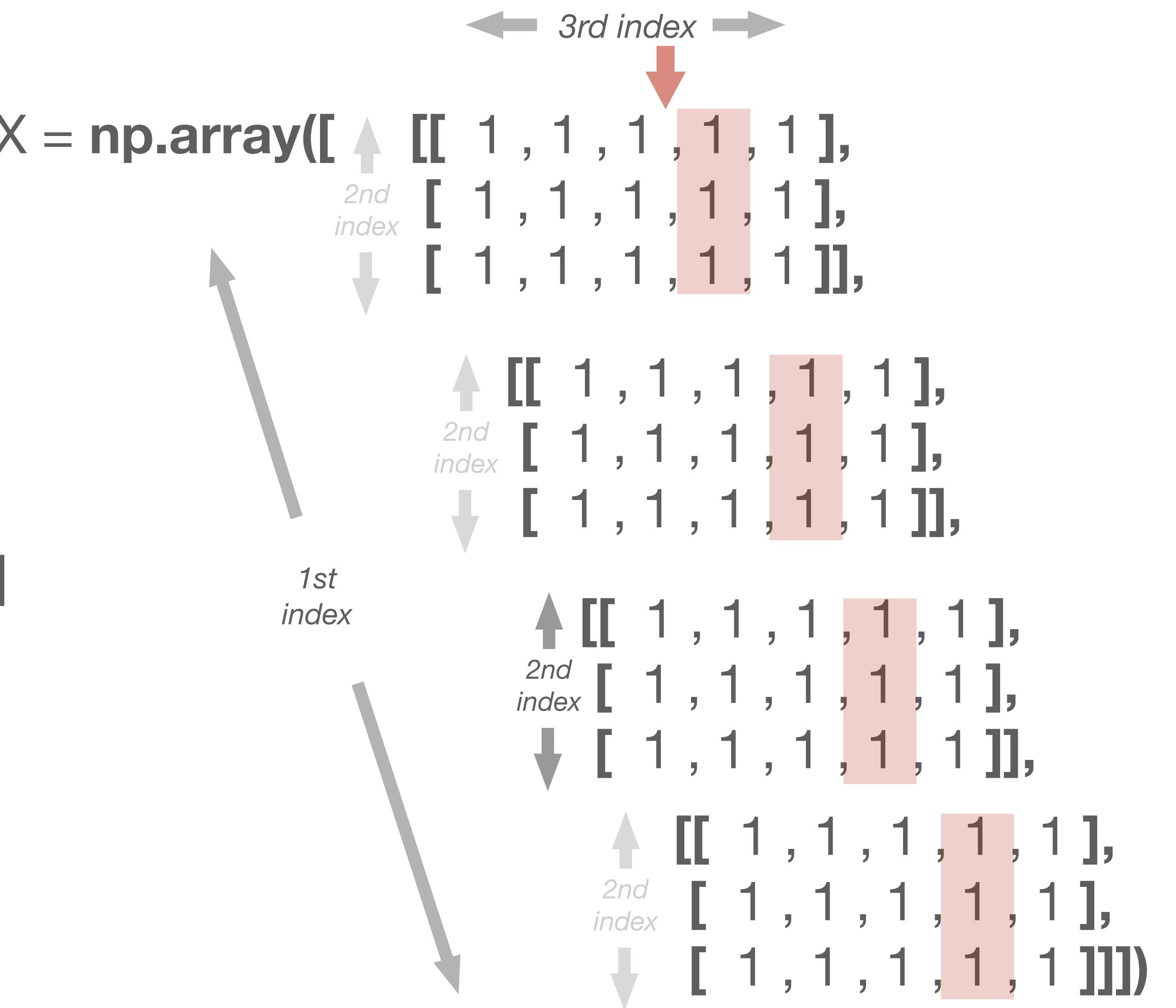
boolean indexing

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]

X[:, :, 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

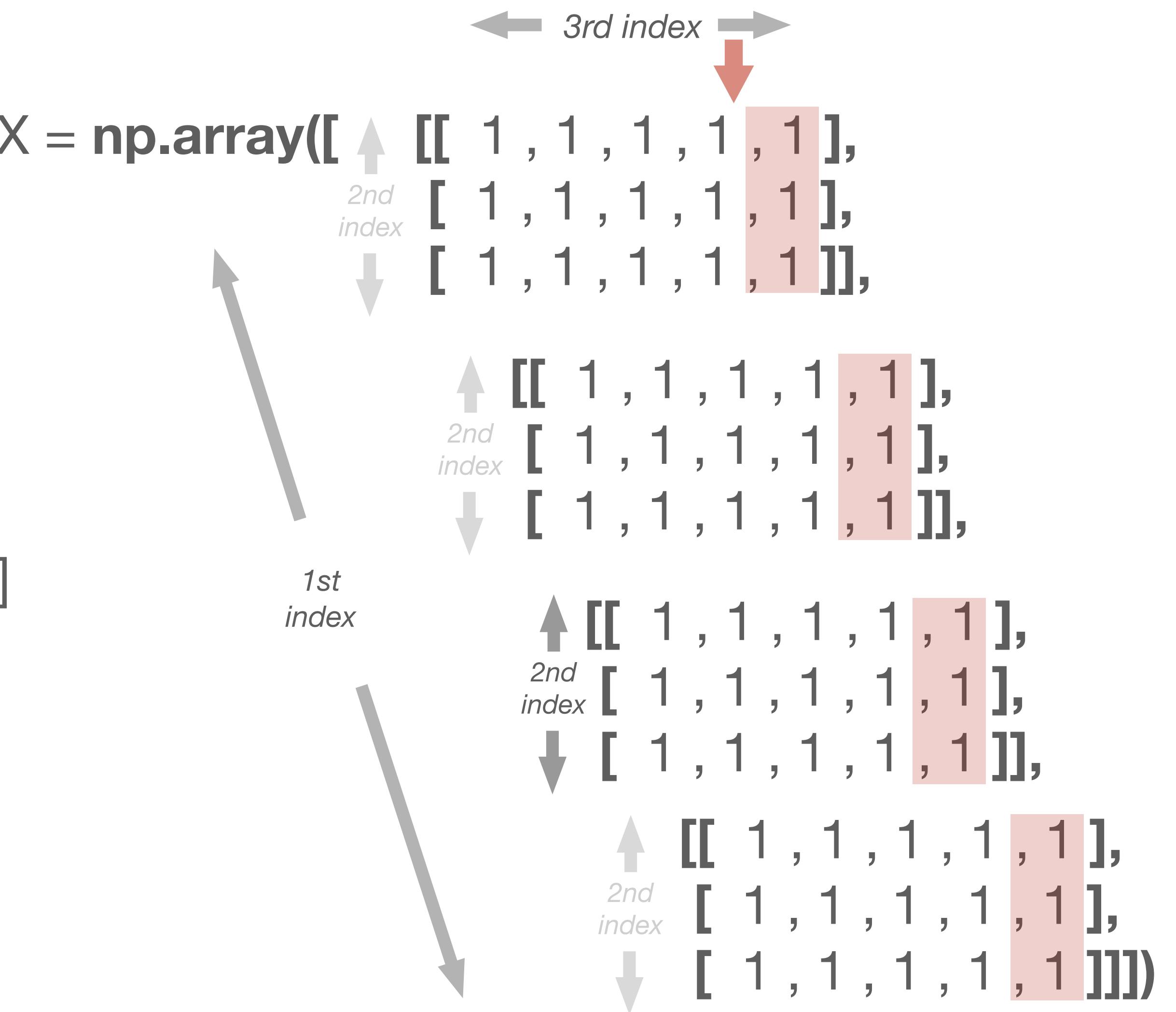
boolean indexing

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]

X[:, :, 4]



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

X[1:,2,3:]

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

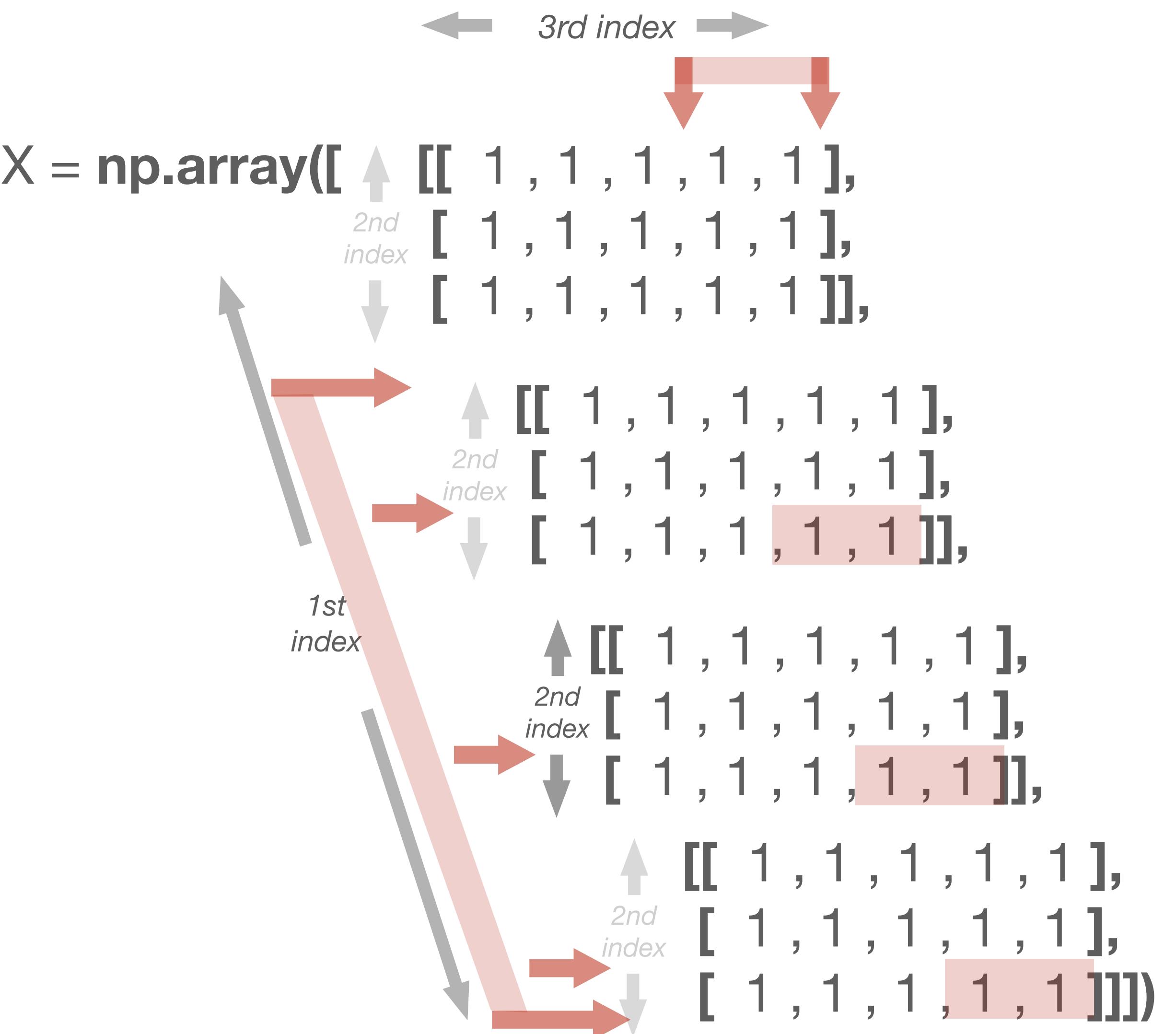
x[bool] - returns 0,1, and 3 element.

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



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,:,: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

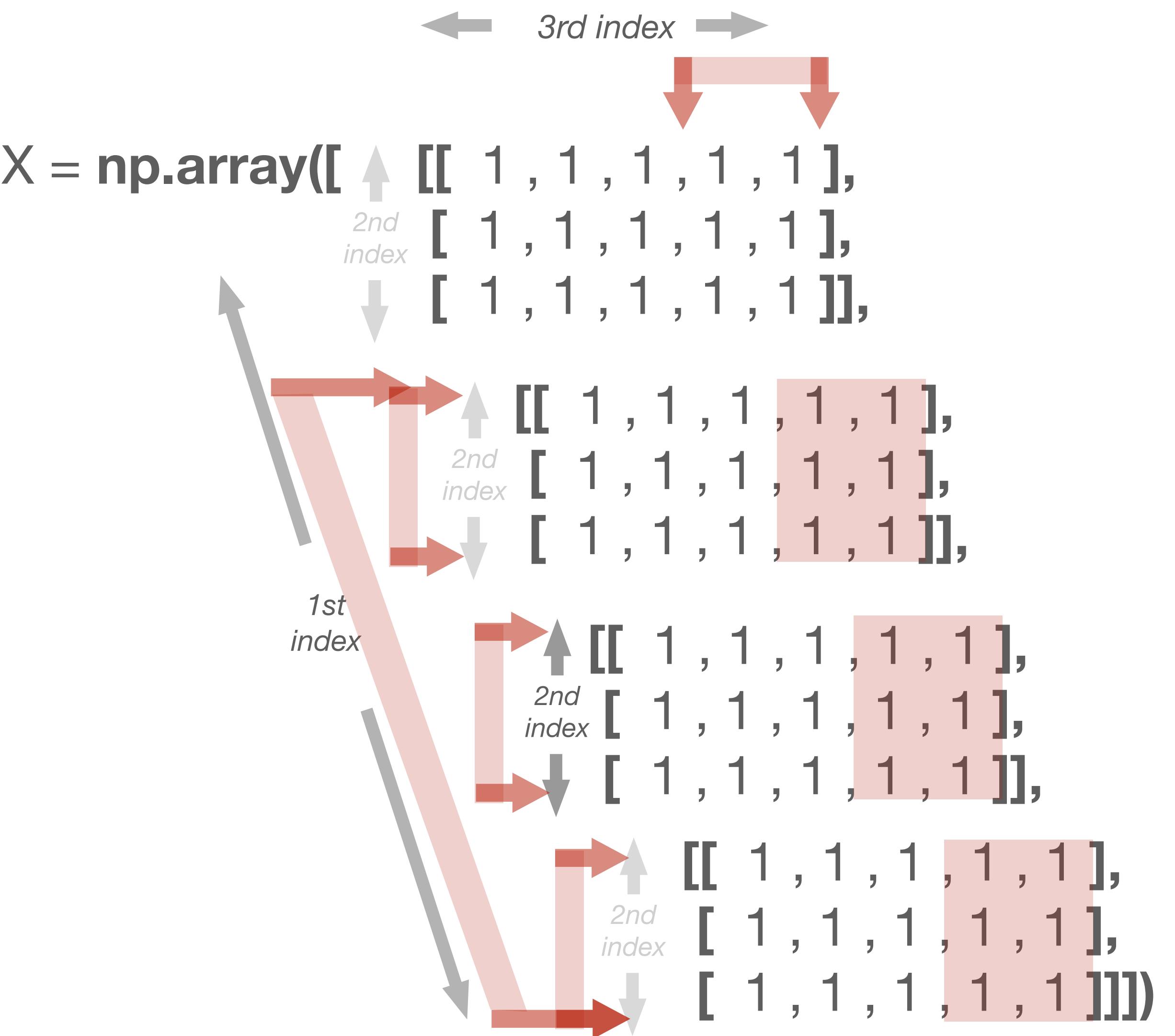
x[bool] - returns 0,1, and 3 element.

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



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

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

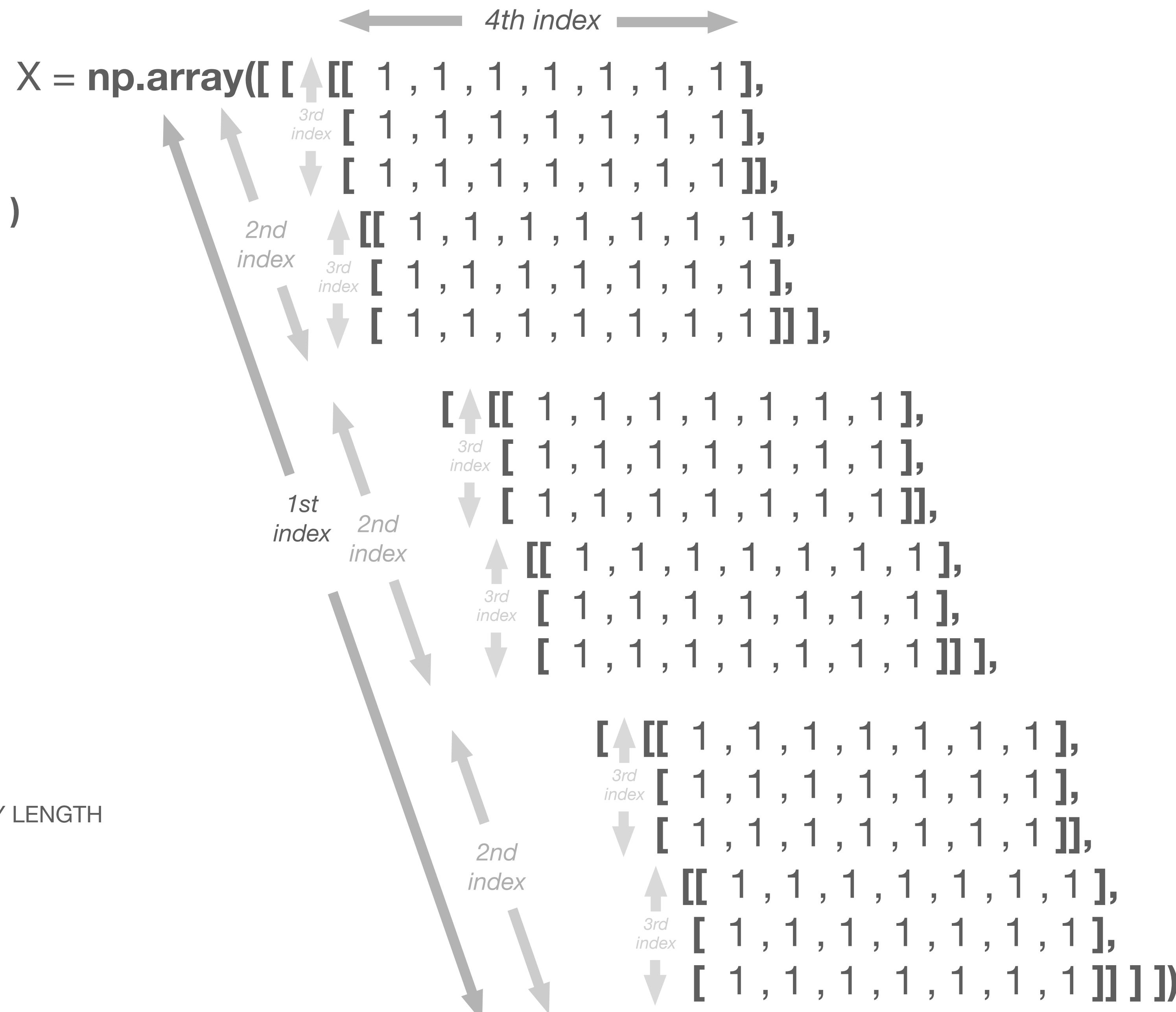
x[bool] - returns 0,1, and 3 element.

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



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

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],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
2nd index
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
4th index
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]]])

[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
2nd index
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
1st index
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]]]])

[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
2nd index
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[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

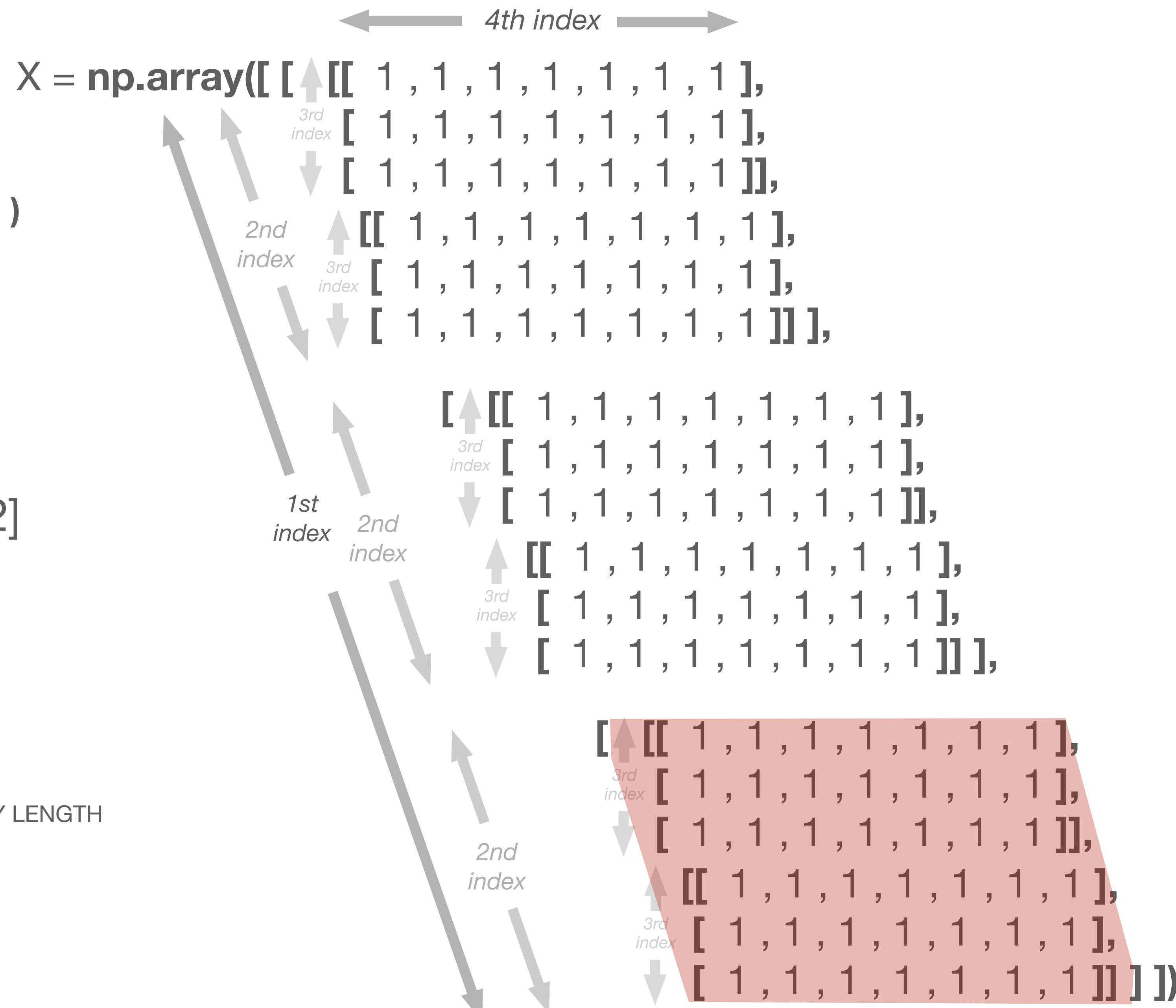
x[bool] - returns 0,1, and 3 element.

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



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],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
2nd index
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
4th index
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
1st index
2nd index
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
2nd index
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
2nd index
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]]]])

[[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]]]])

[[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index
[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

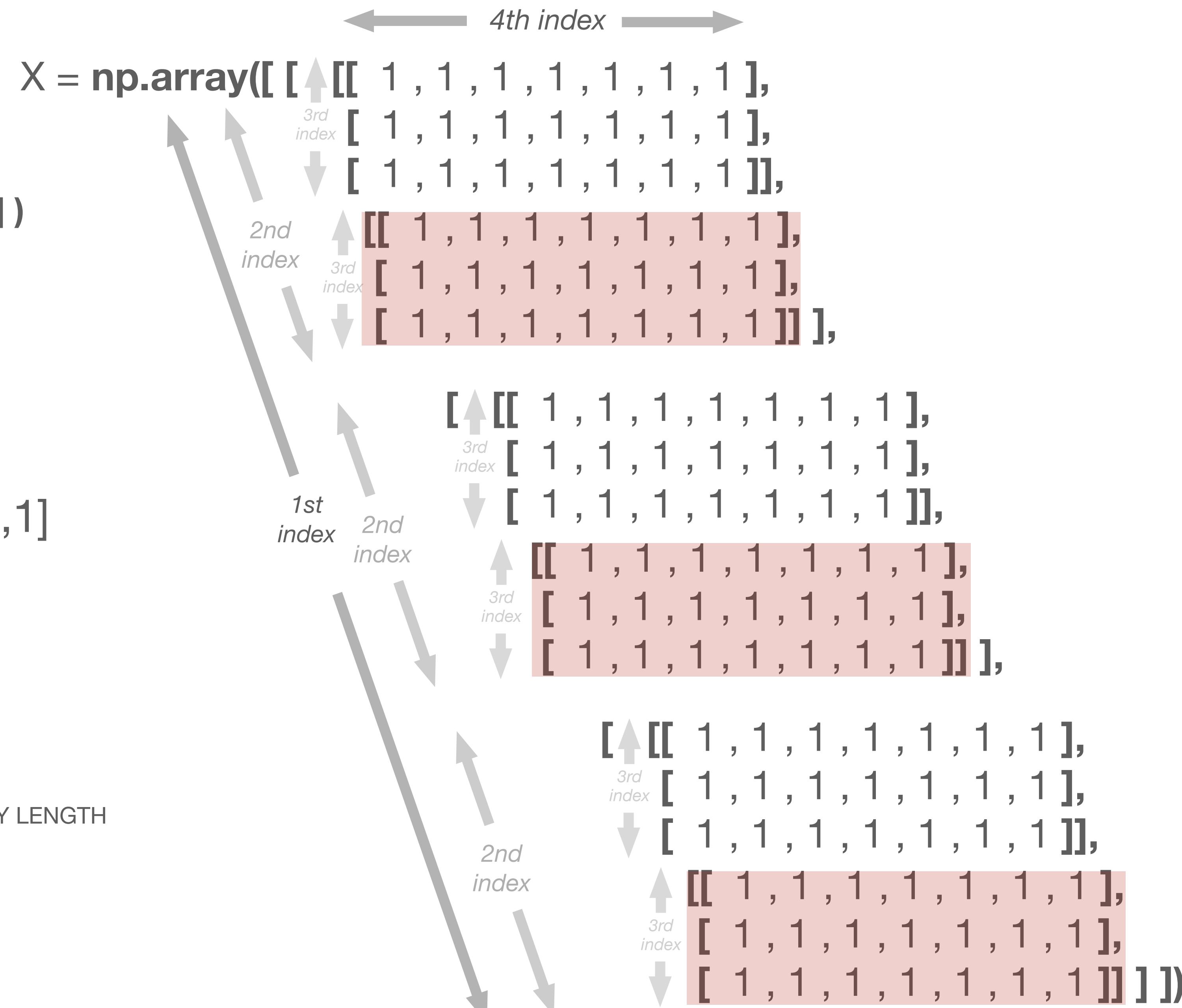
x[bool] - returns 0,1, and 3 element.

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



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

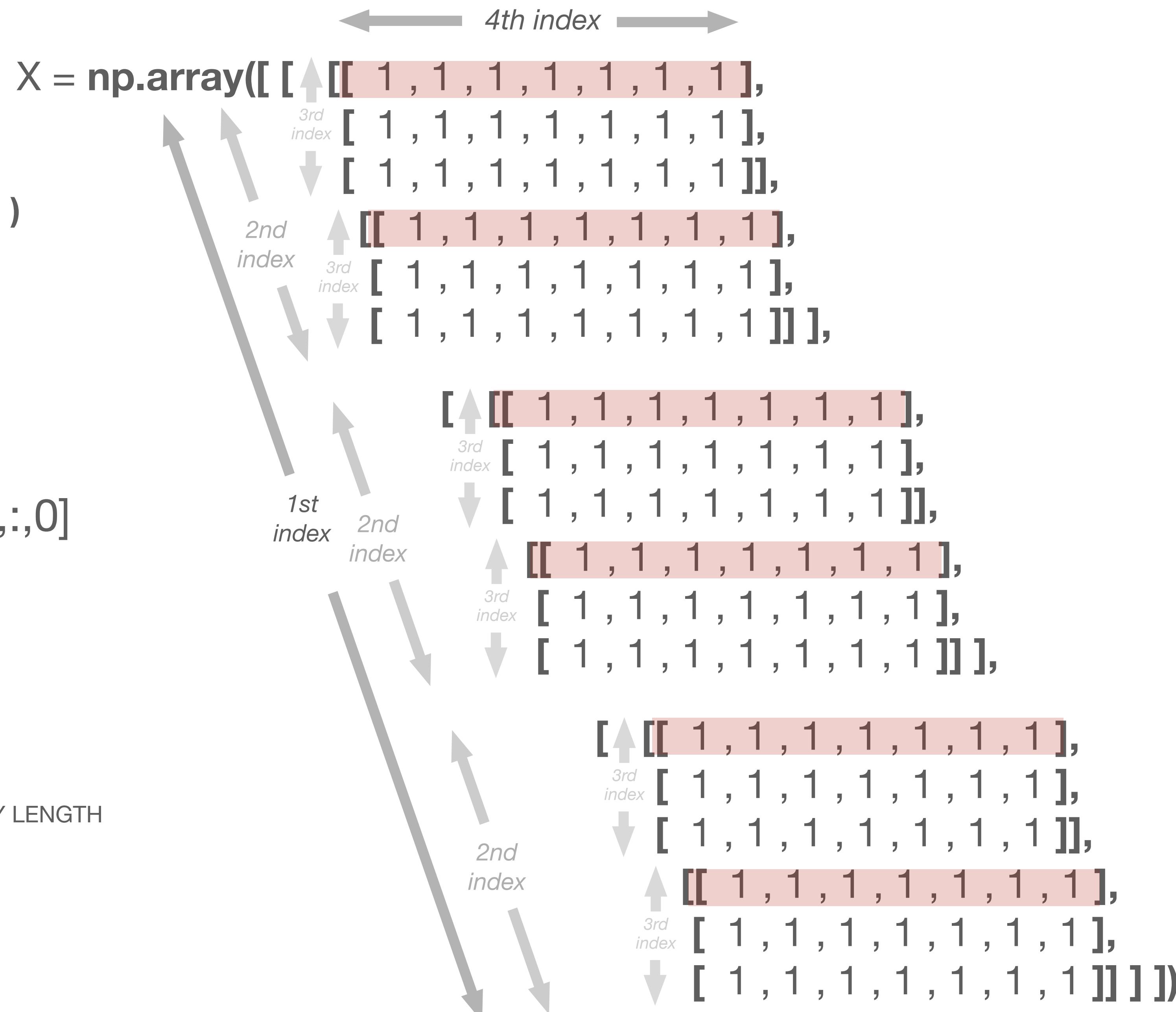
x[bool] - returns 0,1, and 3 element.

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



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

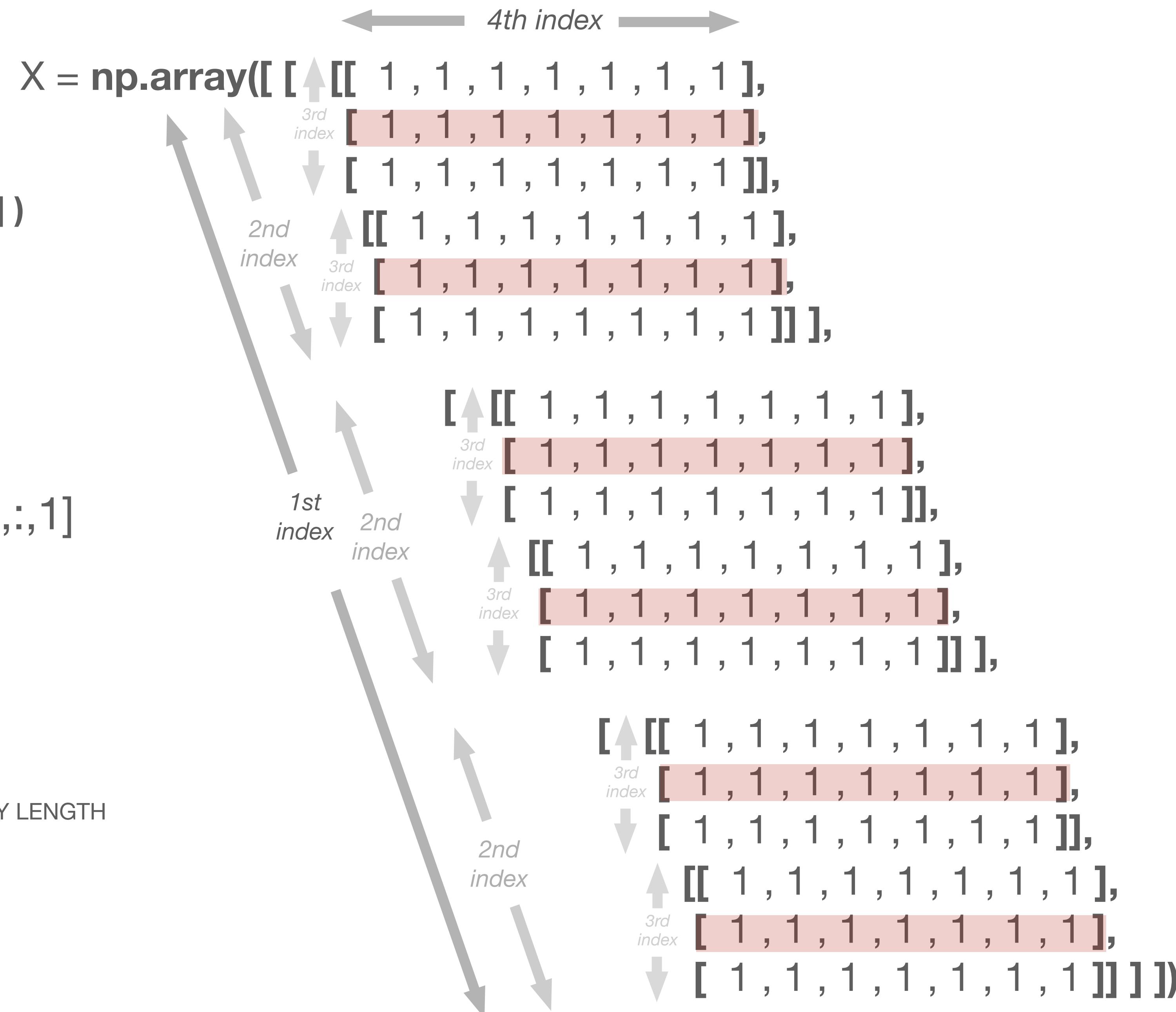
x[bool] - returns 0,1, and 3 element.

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



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

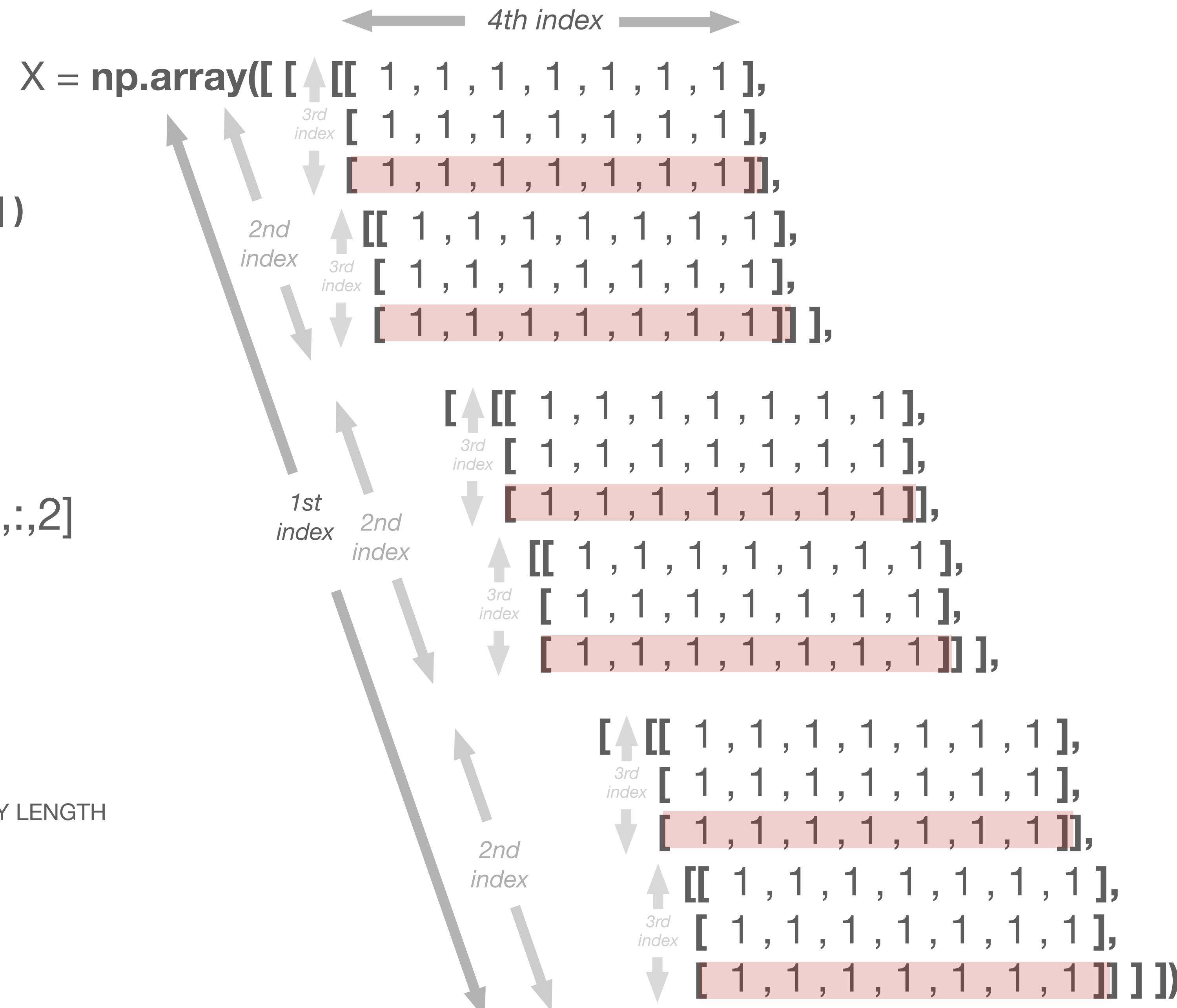
x[bool] - returns 0,1, and 3 element.

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



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.

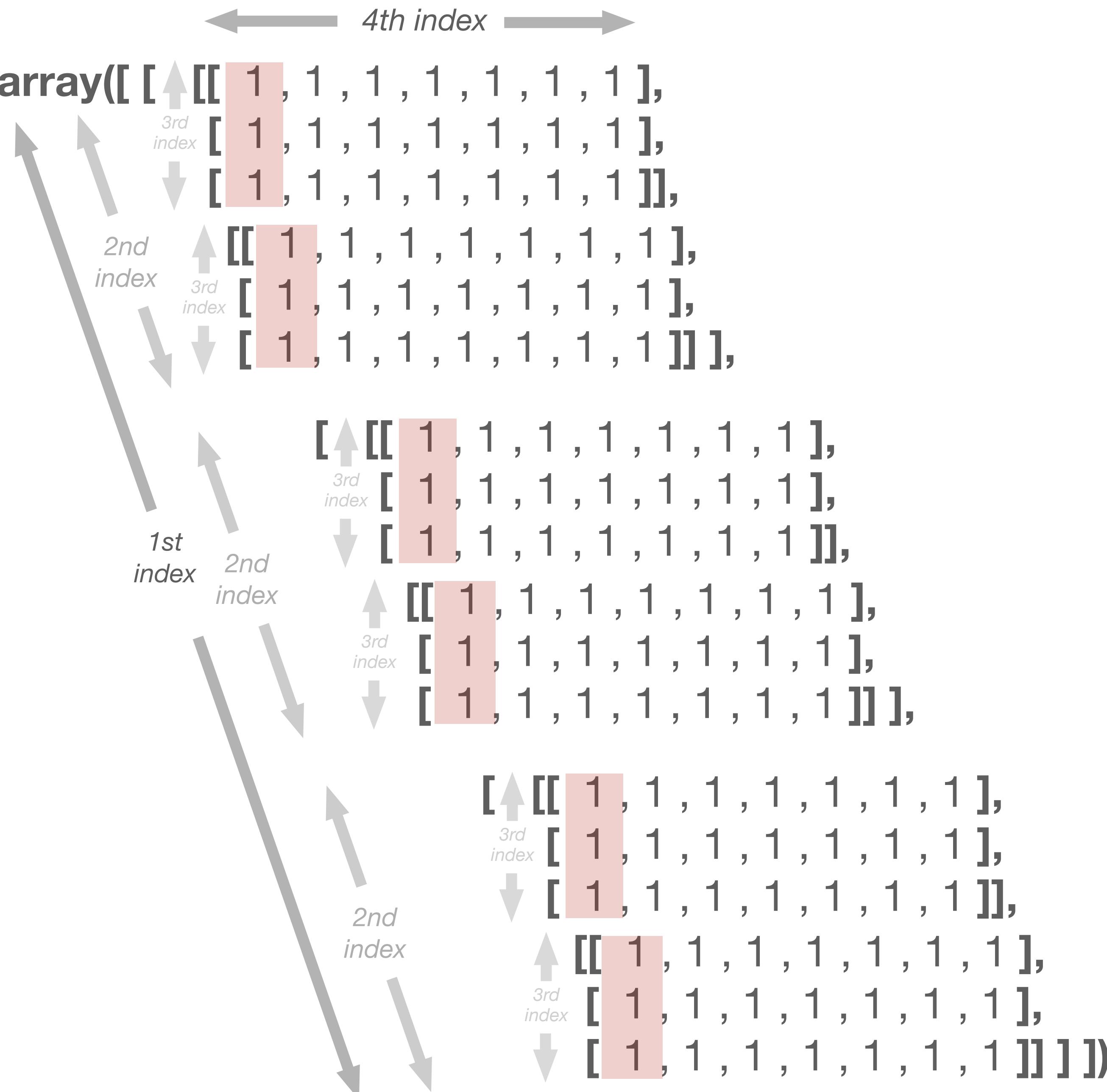
X[bool,bool] - returns the [0,0], [1,1], and [3,3]

block indexing - np.ix_

`X[np.ix_(ind1,ind2)]` - returns the $[0,2,3] \times [3,2]$ block

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

$X[:, :, :, 0]$



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 ],  
 3rd  
index [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],  
  
[ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 3rd  
index [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
```

```
[ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 3rd  
index [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],  
  
[ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 3rd  
index [ 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]]]])

X[:,:,:,:2]

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

[[[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[[[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],
3rd index [1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
2nd index [[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index [1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
4th index
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]]]])

[[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index [1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
2nd index [[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index [1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]]]])

[[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index [1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
2nd index [[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
3rd index [1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[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]]]])

X[:,:,:,:4]

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

[[[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[[[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
[1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]],
[[[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 $[0 \ 1 \ 3] \times [0 \ 1 \ 3]$ 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 ],  
 3rd  
index [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],  
  
[ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 3rd  
index [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],
```

```
[ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 3rd  
index [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],  
  
[ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 3rd  
index [ 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 $[0,2,3] \times [3,2]$ 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 ] ],  
               )
```

The diagram illustrates a 5x10 array X . The array is represented as a list of five lists, where each inner list contains ten elements, all of which are 1. A double-headed arrow at the top right is labeled "4th index". A vertical arrow on the left points upwards and is labeled "2nd index". Two horizontal arrows pointing up and down between the first two columns are labeled "3rd index". The last column of the array is highlighted in red.

```
[ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 3rd index [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ]],  
  
[ [[ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 3rd index [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ],  
 [ 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ]]]]
```

The diagram illustrates a mutation operation on a 3D array. The array is represented as a list of lists of lists:

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

A red vertical bar highlights the 8th element across all three inner lists. A grey arrow labeled "3rd index" points to the third element in the first inner list. Another grey arrow points downwards from the first inner list to the second inner list.

After the mutation, the array becomes:

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

The 8th element in each inner list has been mutated to 1, indicated by the red background.

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

X[[1,3,4]]

boolean indexing

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

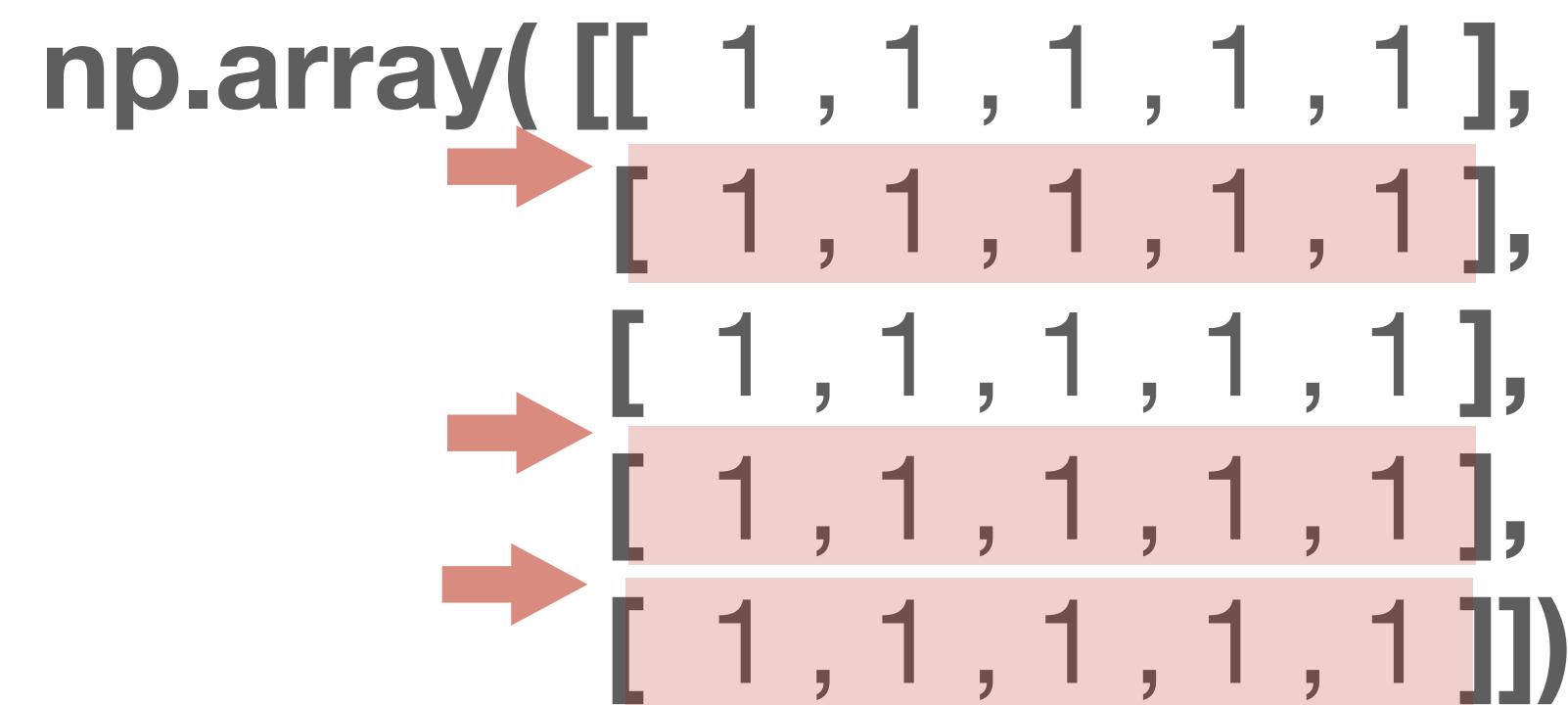
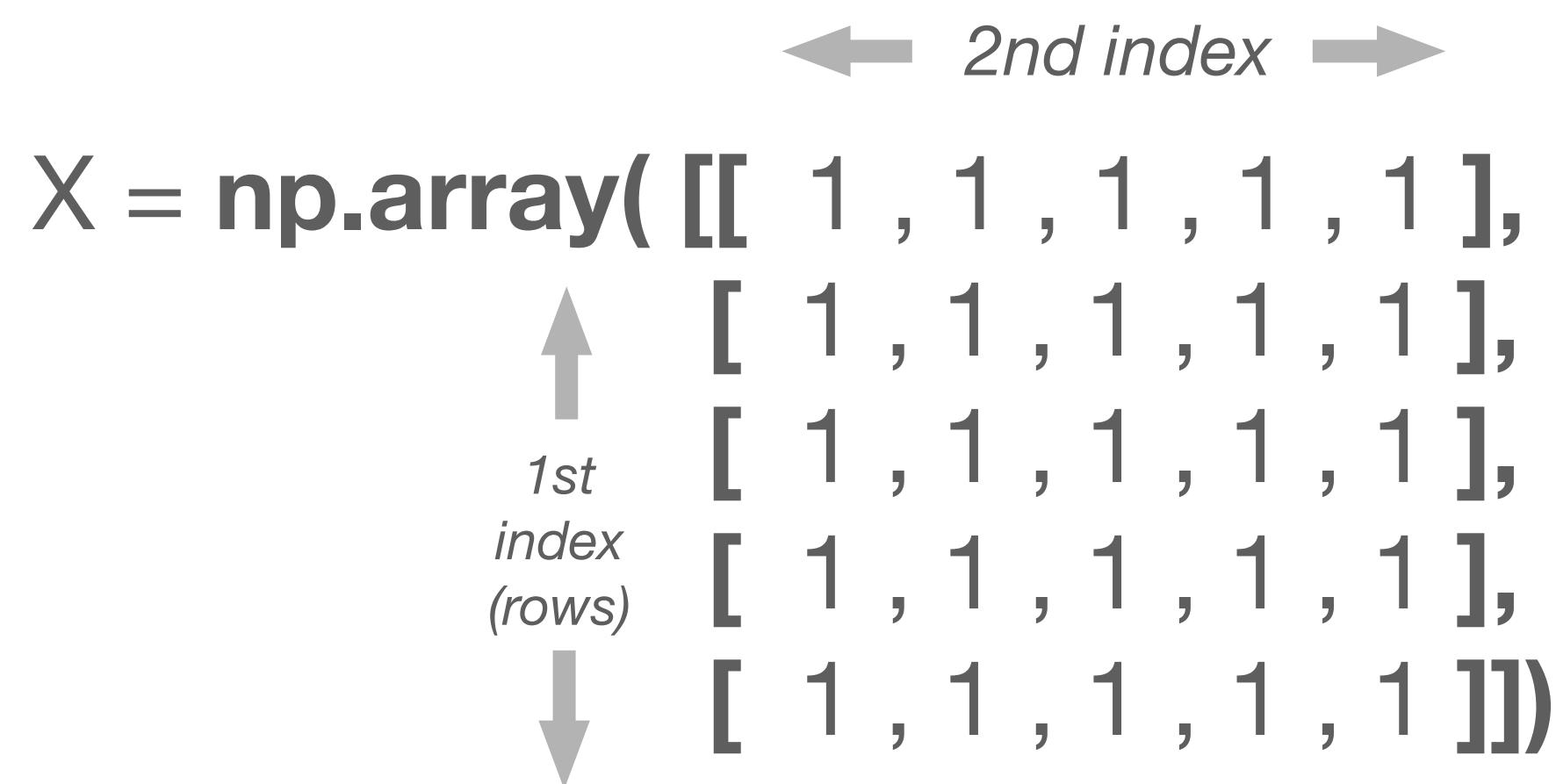
x[bool] - returns 0,1, and 3 element.

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

↑
1st index (rows)
↓

← 2nd index →

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],[2,4,0]]

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];

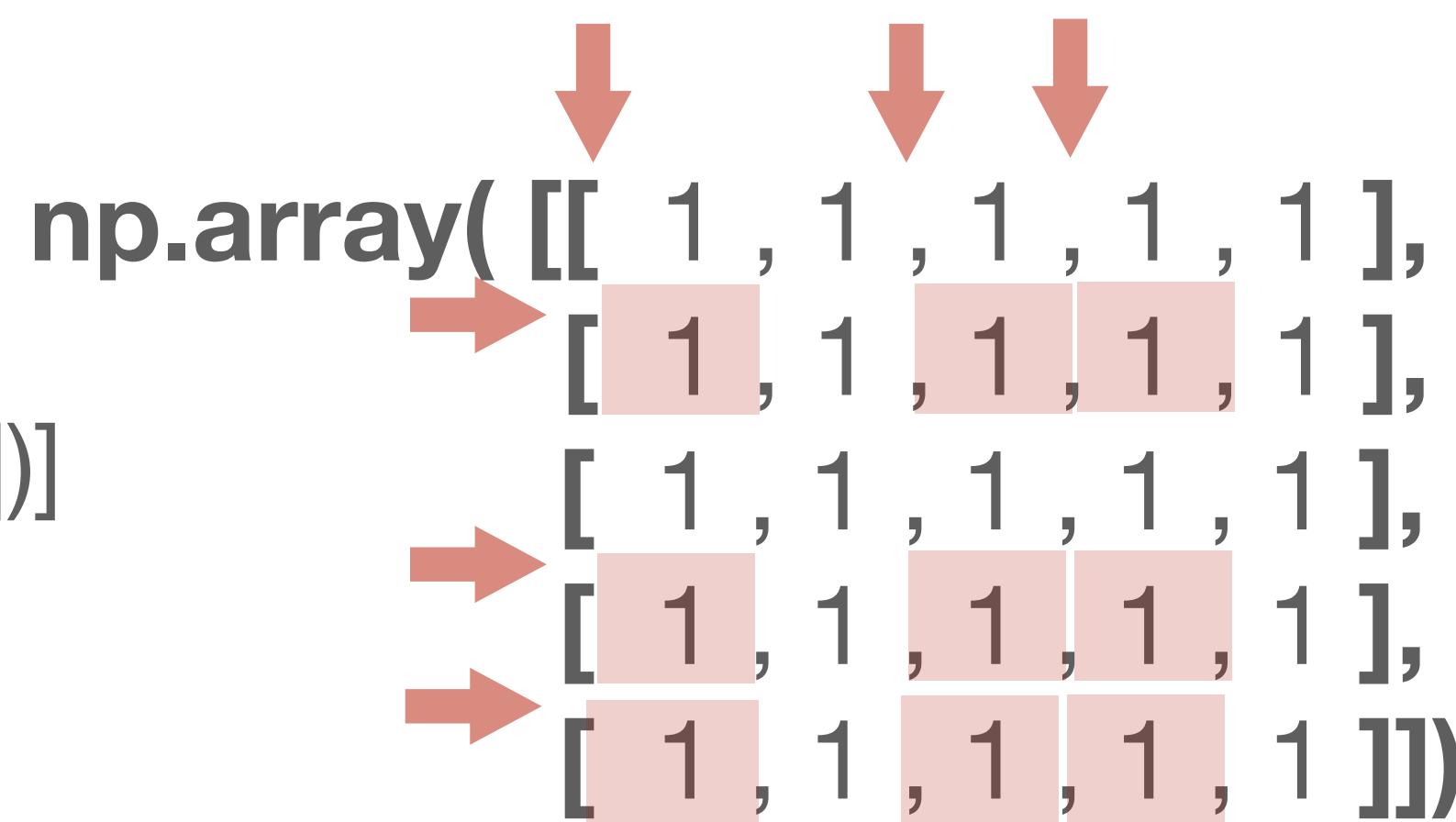
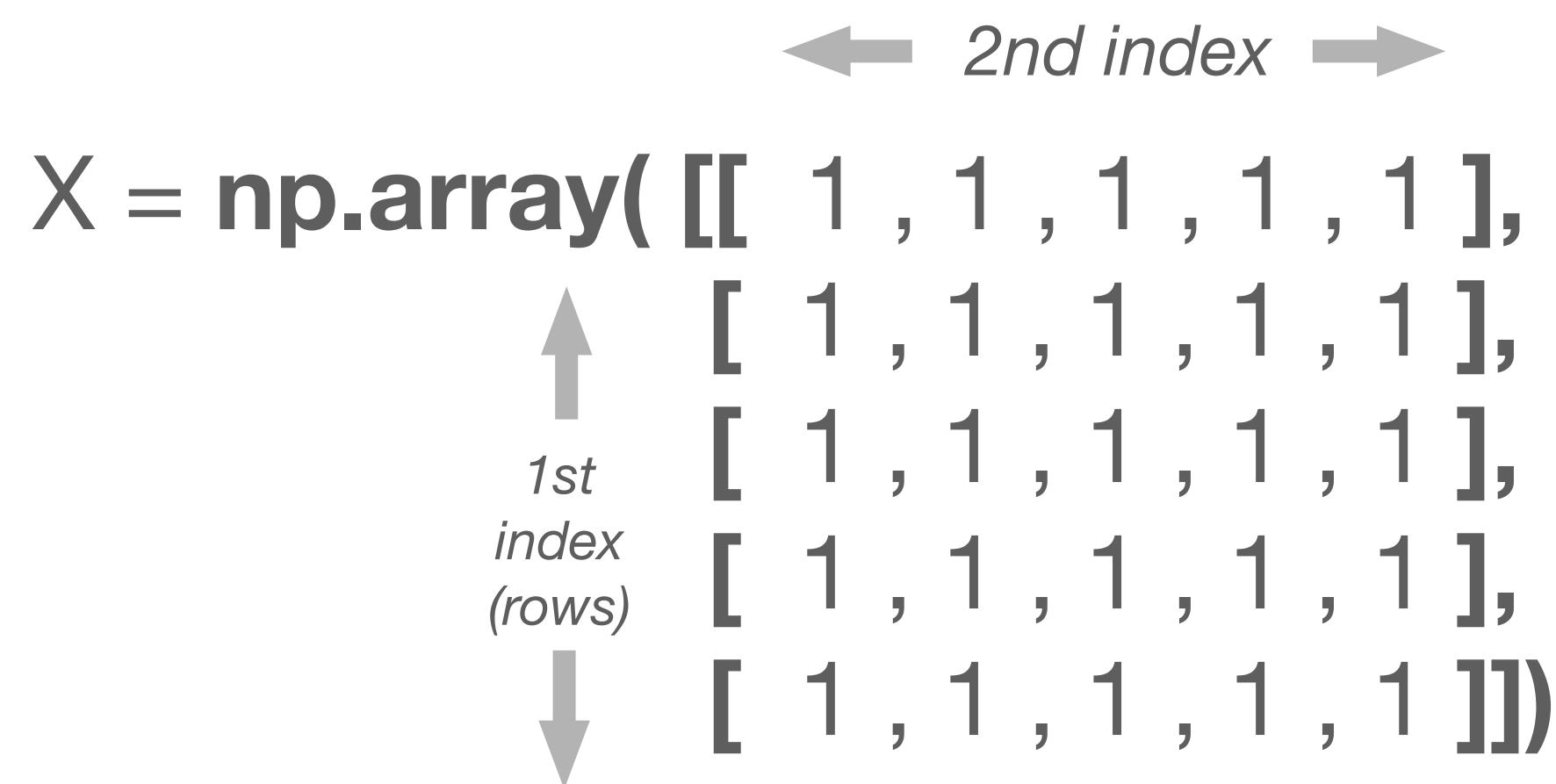
X[ind1,ind2] - returns [0,0],[2,3], and [3,2] elements

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]



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

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

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

X[bools]

boolean indexing

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

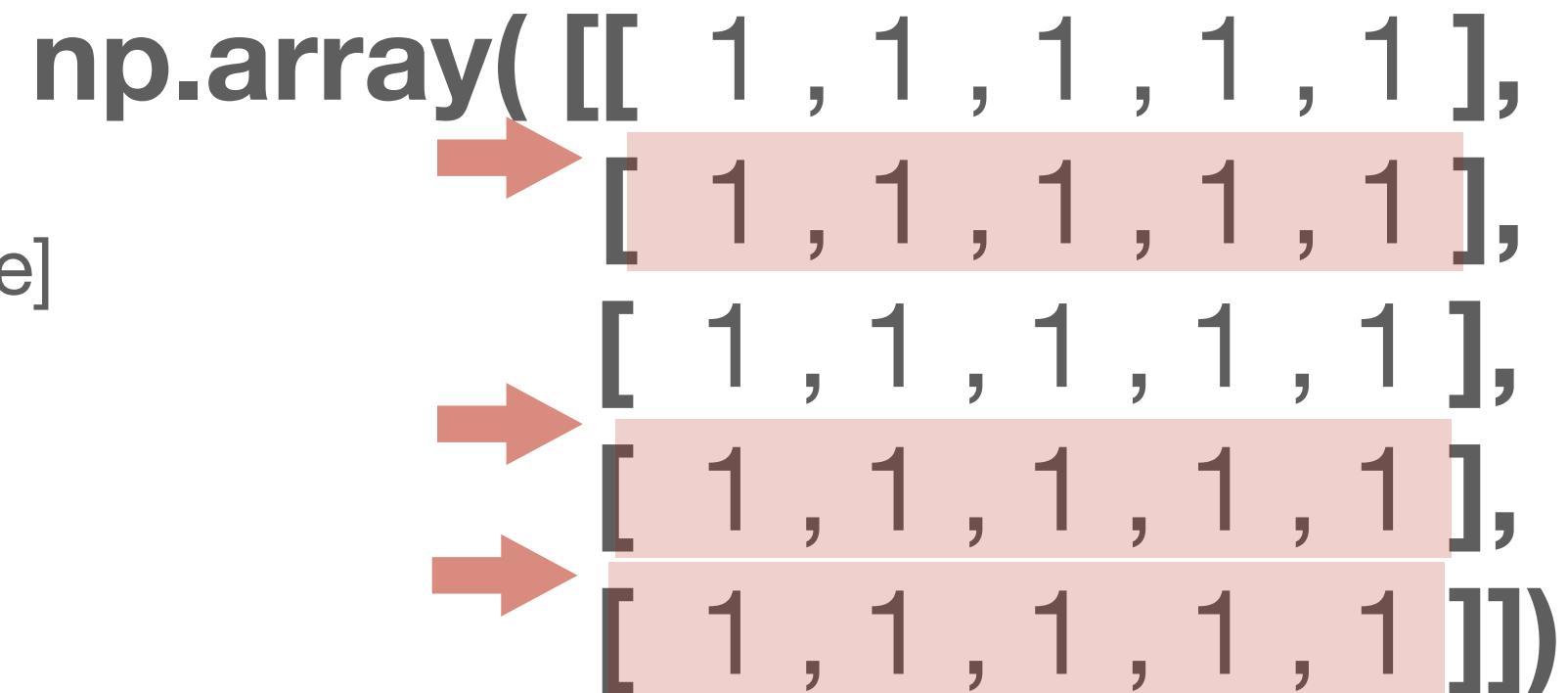
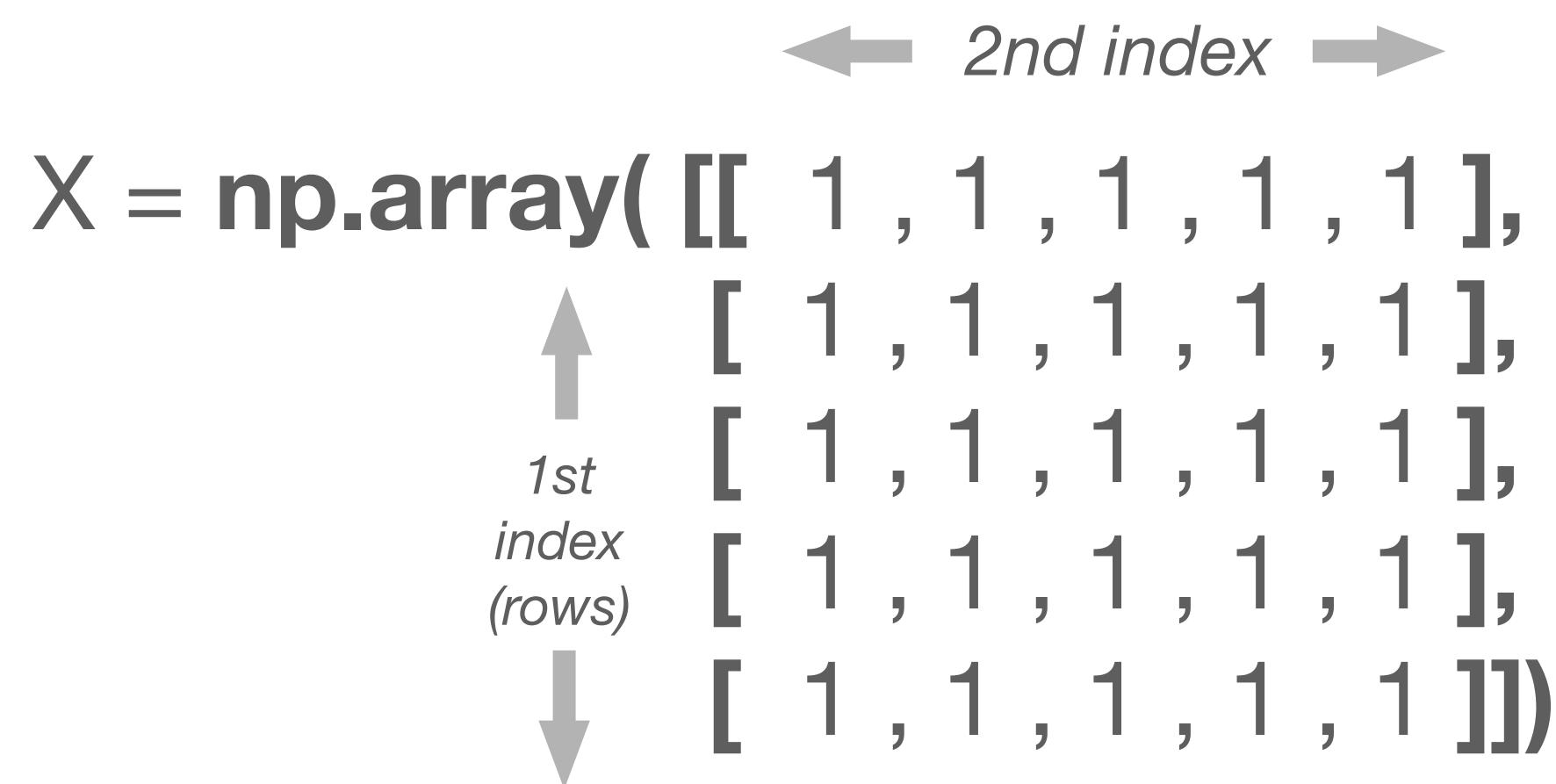
x[bool] - returns 0,1, and 3 element.

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

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

X[bools, bools]

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)
↓

↓ ↓ ↓

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

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

X[np.ix_(bools,bools)]

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 , 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

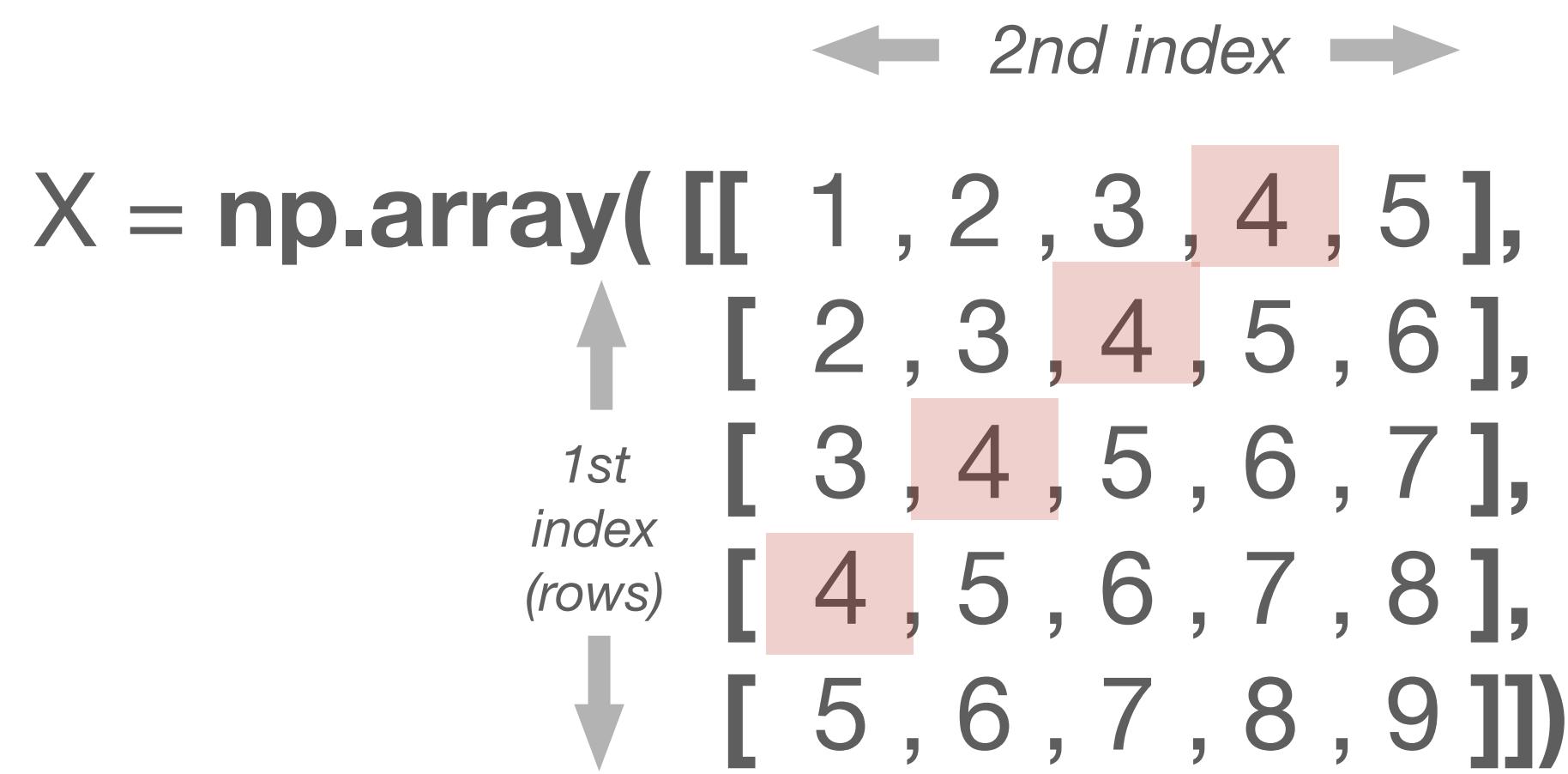
x[bool] - returns 0,1, and 3 element.

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



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

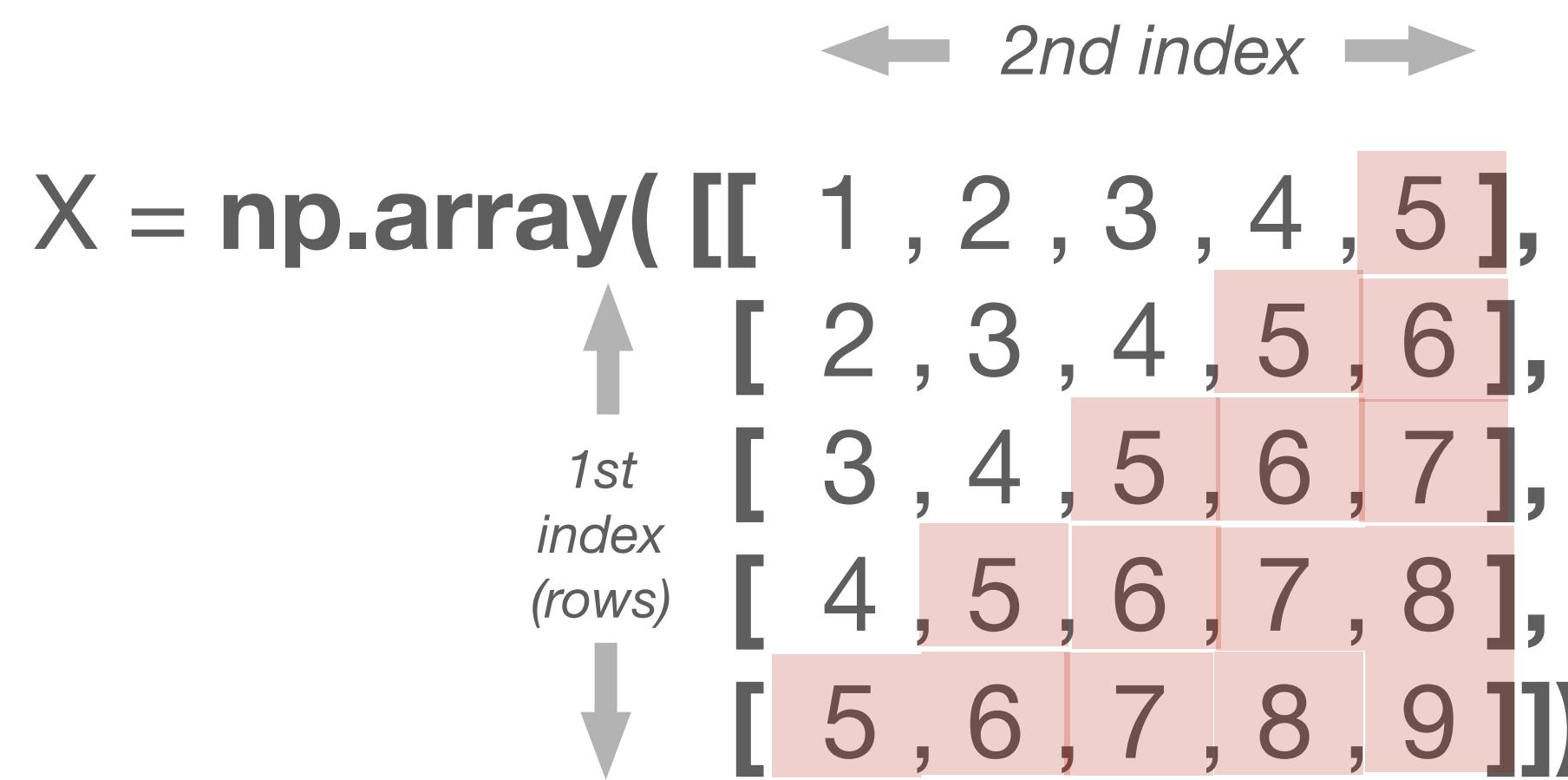
x[bool] - returns 0,1, and 3 element.

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



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) = 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]])

or...

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

[[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

$$y = Ax$$

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

Transpose `A.T`

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

`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]])`

`np.einsum('ij,j',A,x)`

Python - Matrix Multiplication

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

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]])`

BOTH

col vector:

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

`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

Transpose `A.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: col vector:

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

BOTH

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

Matrix multiplication:

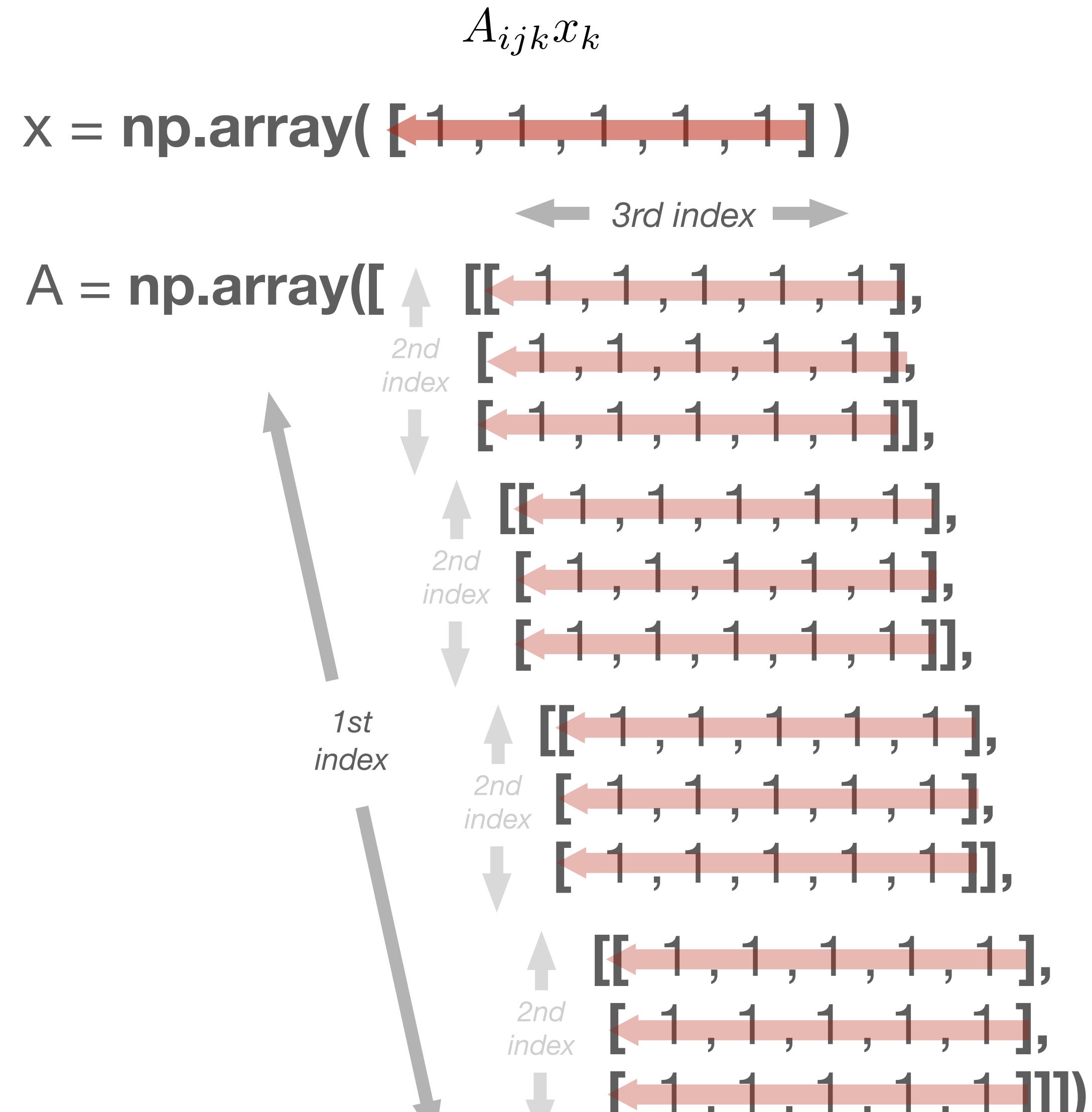
`A@x = A.dot(x) = np.dot(A,x)`

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

`A@x` - A times col vector x

`x@A` - row vector x times A

Transpose `A.T`



`np.einsum('ijk,k',A,x)`

Python - Matrix Multiplication

vector: 1D $x = \text{np.array}([1,2,3])$

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

row vector: **col vector**

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

BOTH

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

Matrix multiplication:

$$A @ x = A \cdot \text{dot}(x) = \text{np}.\text{dot}(A, x)$$

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

$A @ x$ - A times col vector x

$x @ A$ - row vector x times A

Transpose A.T

```
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]])`

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

$A_{ijk}x_i$

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

3rd index

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

Transpose `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: col vector:

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

BOTH

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

Matrix multiplication:

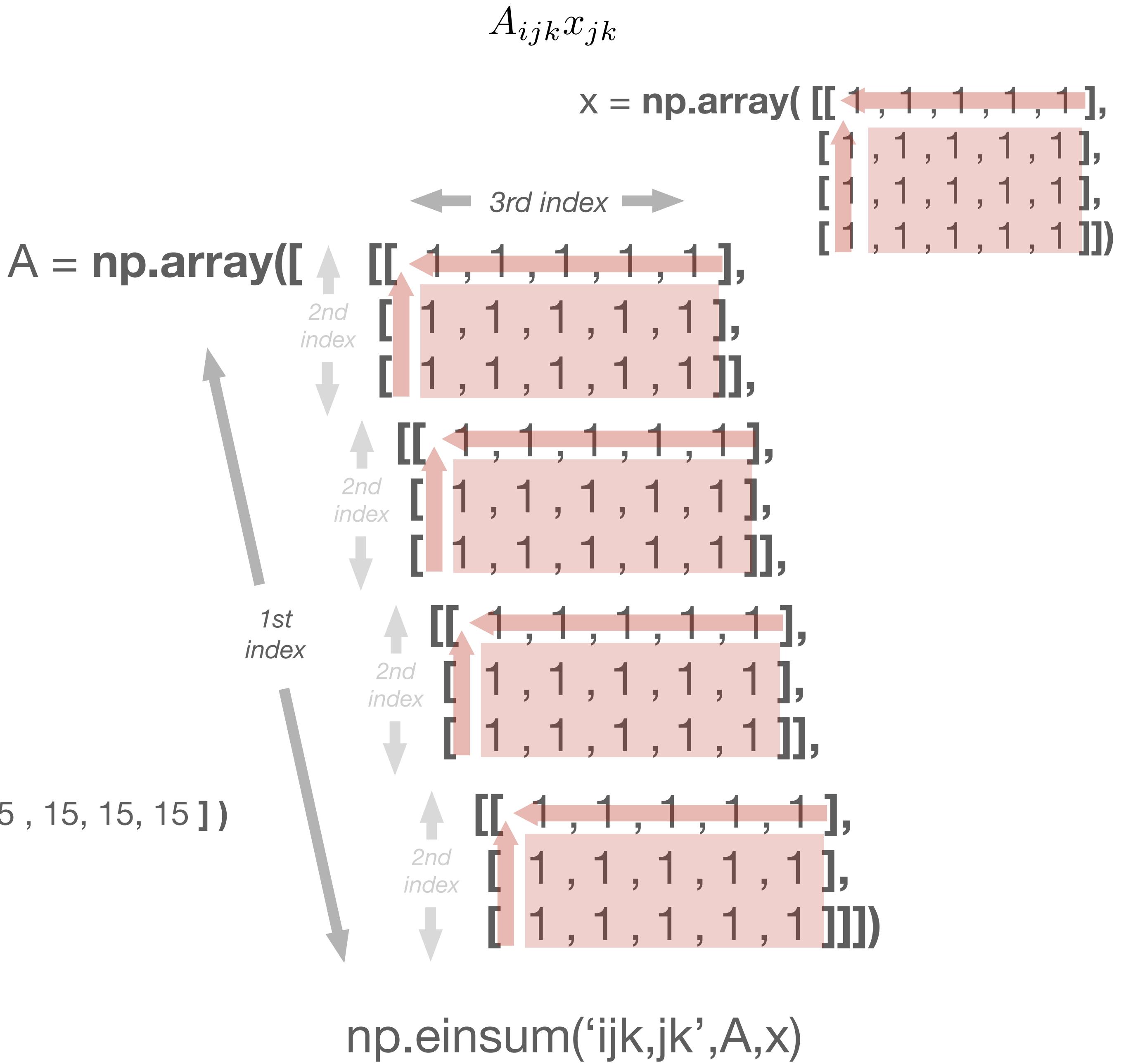
`A@x = A.dot(x) = np.dot(A,x)`

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

`A@x` - A times col vector x

`x@A` - row vector x times A

Transpose `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: col vector:

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

BOTH

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

Matrix multiplication:

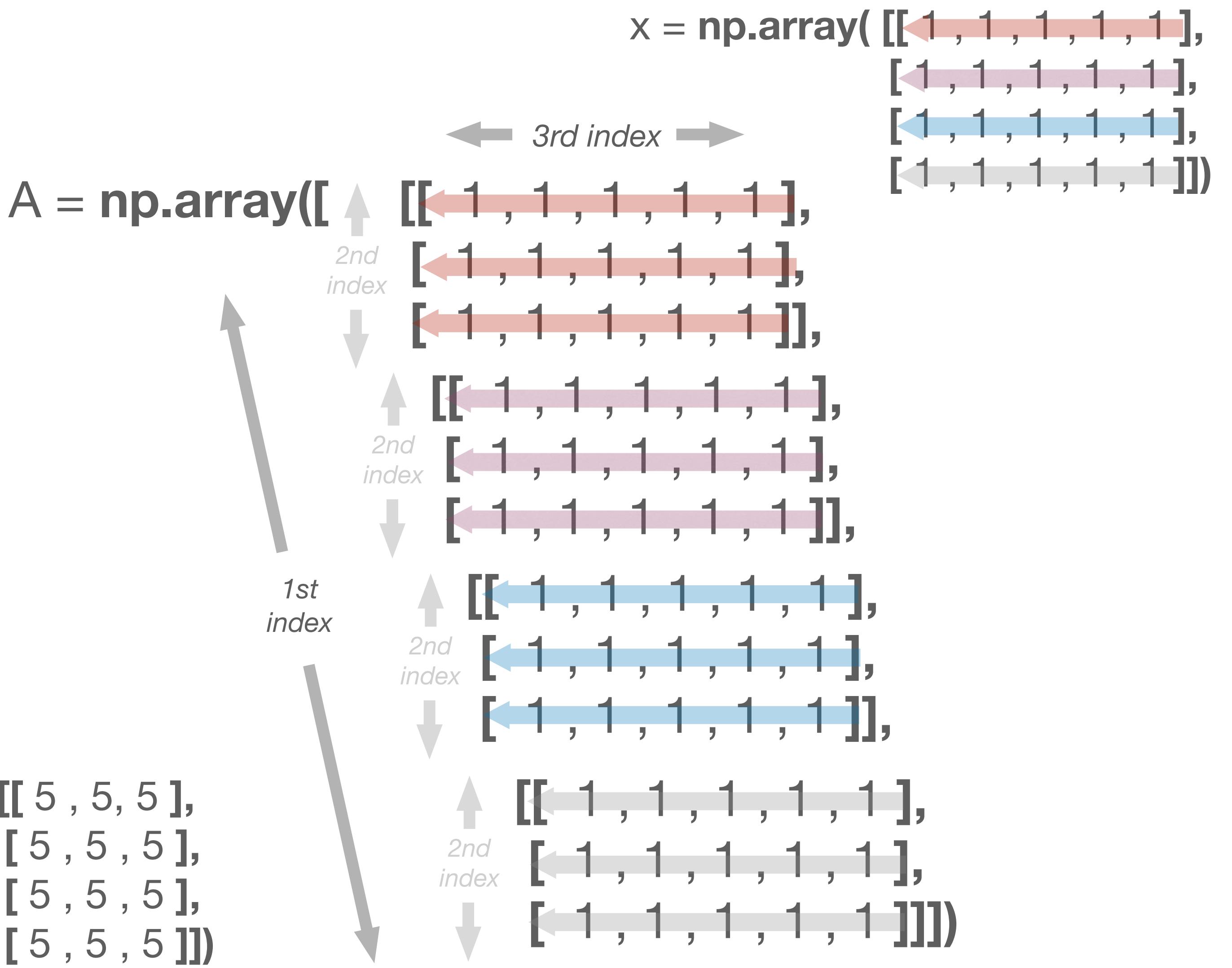
`A@x = A.dot(x) = np.dot(A,x)`

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

`A@x` - A times col vector x

`x@A` - row vector x times A

Transpose `A.T`



`np.einsum('ijk,ik->ij', A, x)`