NumPy basics

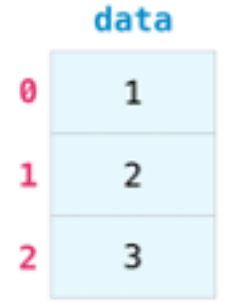
The lecture material is based on an excellent resource:

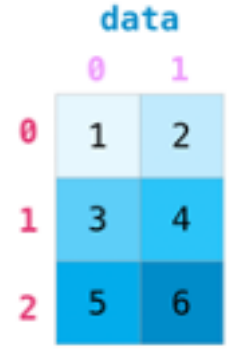
NumPy: the absolute basics for beginners
(https://numpy.org/doc/stable/user/absolute_beginners.html)

Also: (Required reading) **Array programming with NumPy**
(https://www.nature.com/articles/s41586-020-2649-2)

Memorization is not required. However, you must practice everything in this lecture note. The best way to learn these syntaxes is to use them multiple times.

Array initialization





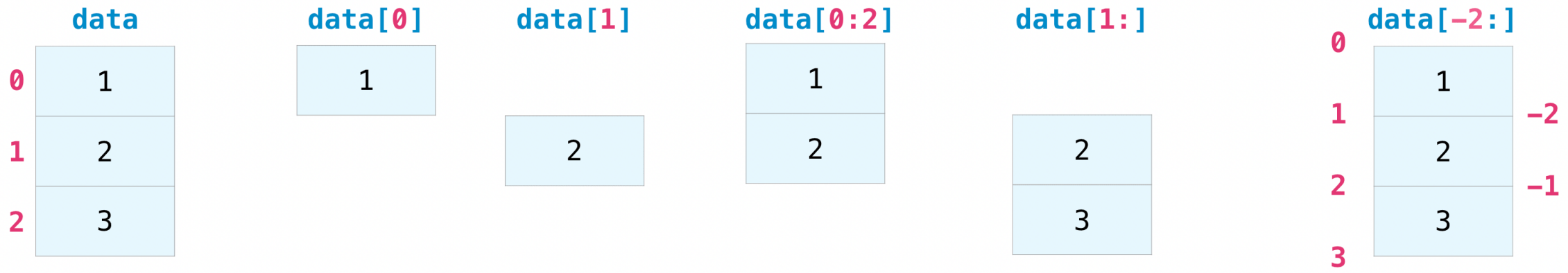

```
np.array()              a = np.array([1.0, 2, 3])
                        b = np.array([[1,2],[3,4],[5,6]])

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

np.zeros()              a = np.zeros(5)
np.ones()               a = np.ones(3)
np.empty()              a = np.empty(2)  # random initial

np.arange()             a = np.arange(3,20,3)

np.linspace()           a = np.linspace(5,20,num=10)

np.random.randint(low, high, size=(m,n)) # does not include 'high'
                        a = np.random.randint(3, 10, size=20)
```

Information about an array

```
ndarray.ndim          # number of axis (number of dimensions)

ndarray.shape         # number of elements in each dimension

ndarray.size          # total number of elements

ndarray.dtype         # type of elements

np.unique()
  a = np.random.randint(3, 10, size=(100))
  uq = np.unique(a)
  uq, idx = np.unique(a, return_index = True) # index of the first one
  uq, c = np.unique(a, return_counts = True)
  uq, idx, c = np.unique(a, return_index = True, return_counts = True)
```

Indexing and slicing

```
data = np.array([1, 2, 3])
b = data[0]
c = data[0:2]
d = data[1:]
e = data[-2:]
```



Conditional indexing

```
a = np.arange(6)
a = np.array([[1,2],[3,4],[5,6]])  # 2D array

print(a[a<4])

cond = (a>4)
print(cond)
print(a[cond])

b = a[ (a>2) & (a%2==0) ]
print(b)

d = np.nonzero( a<4 )
print(d)                    # prints index of a<4 elements
```

Creating (initializing) a new array from existing arrays

```
c = np.concatenate((a,b))

b = a[2:]              # Using slicing  (reference copy)

b = a[2:].copy()       # Shallow copy, but in NumPy, it works
                       # like deep copy, since there is no
                       # nested array.

b = a.view()           # A new array sharing the same memory

ex)
b = a.view(dtype=np.float32).reshape(3,-1)
a[3] = 100
print(a)
print(b)
```

Operators

```
+  -  *  /  ==  <    # element-wise operations
                     # broadcasting

ndarray.sum()        # summation
ndarray.sum(axis=1)  # summation over a certain dimension
                     # see Matrix operations in the
                     # following slides

ndarray.max()
ndarray.max(axis=0)

ndarray.min()
ndarray.min(axis=1)

np.sqrt()  np.exp()  np.log()  np.log2()  etc…
                    # all element-wise operations
```