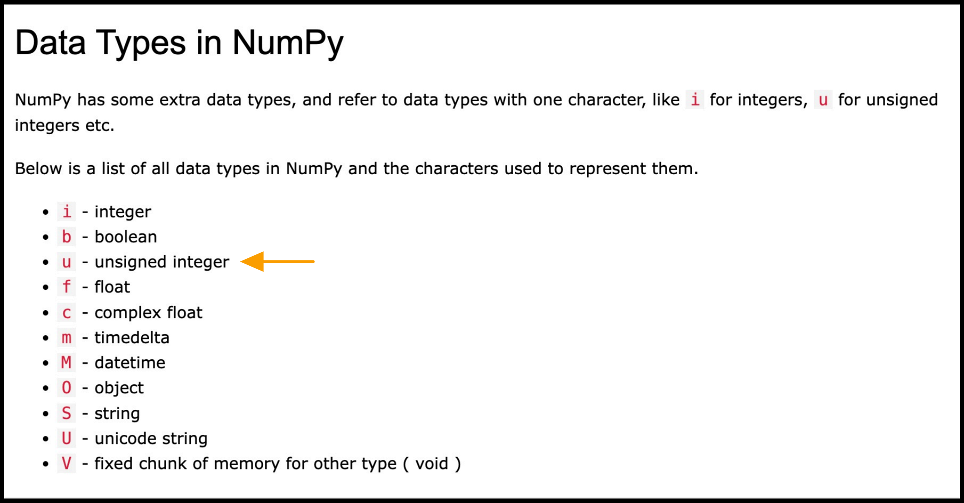
Numpy Library.

np. array ⟹ ndarray

1D, 2D, 3D



Data Types in NumPy

NumPy has some extra data types, and refer to data types with one character, like `i` for integers, `u` for unsigned integers etc.

Below is a list of all data types in NumPy and the characters used to represent them.

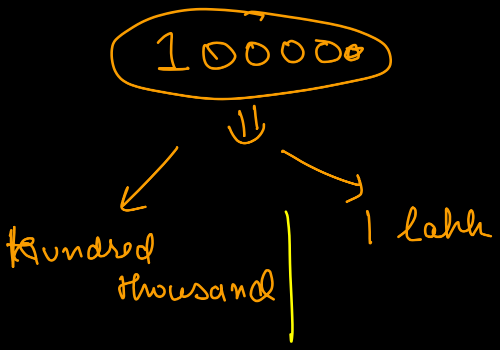
- `i` - integer
- `b` - boolean
- `u` - unsigned integer ←
- `f` - float
- `c` - complex float
- `m` - timedelta
- `M` - datetime
- `O` - object
- `S` - string
- `U` - unicode string
- `V` - fixed chunk of memory for other type (void)

Because, we are making our numpy array using numpy fn.

the type is also taken as that of numpy.int64 int

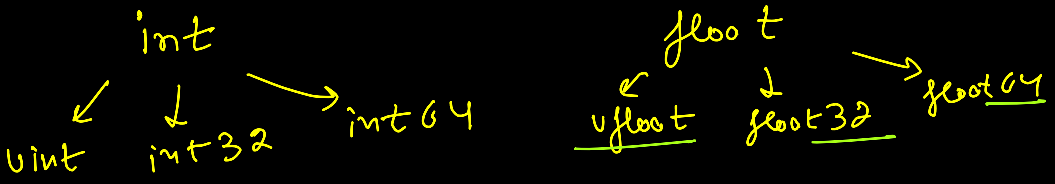
float

string

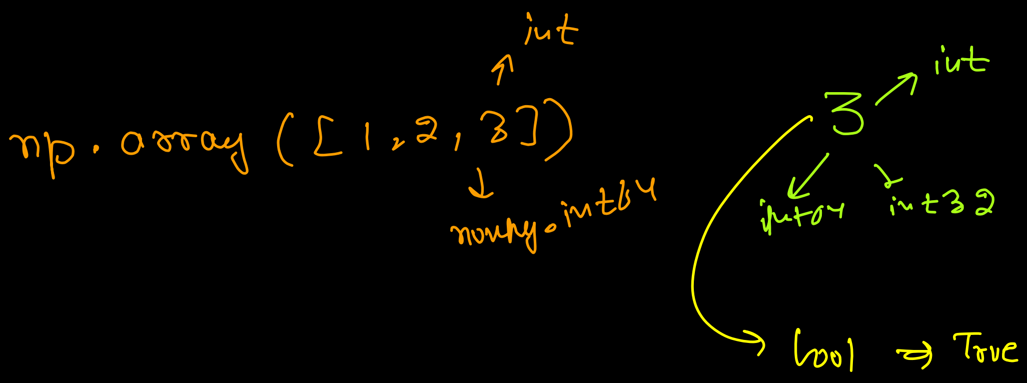


1 , 2 , 3

int
float



Learn about your 32, 64 & unsigned types?



Create Numpy Arrays

✓ np.array → with Type
without Type

✓ np.arange

✓ np.ones

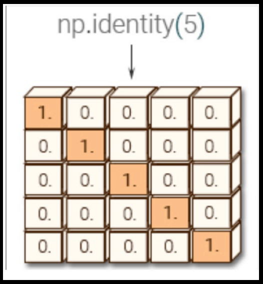
✓ np.zeroes

✓ np.random.random

✓ np.linspace

✓ np.identity

✓ np.eye


np.identity(5)

1.	0.	0.	0.	0.
0.	1.	0.	0.	0.
0.	0.	1.	0.	0.
0.	0.	0.	1.	0.
0.	0.	0.	0.	1.

$[\infty, \infty, \infty, \infty, \infty, \infty, \infty,$
$\infty, \infty]$

⇙

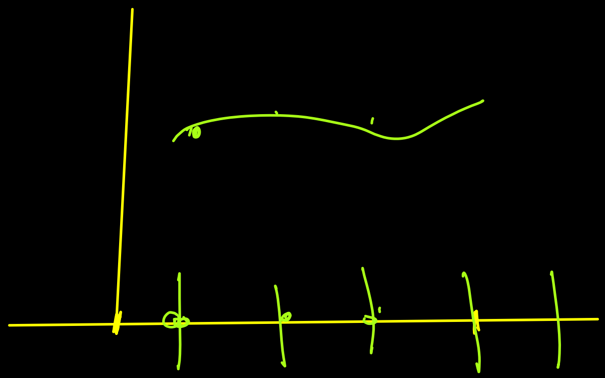
$[\infty \ \infty]$
$[\infty \ \infty]$

$[\infty \ \infty]$
$[\infty \ \infty]$

5 no. which are equidistant from each other b/w [-10 , 10] both included

3 → -10 , 0 , 10

5 → -10 , -5 , 0 , 5 , 10



Array attributes

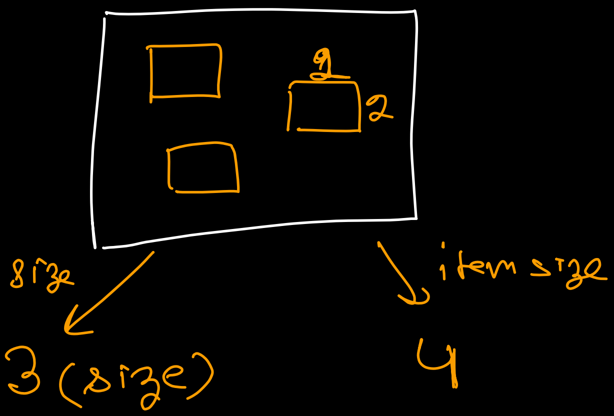
shape

n. dim

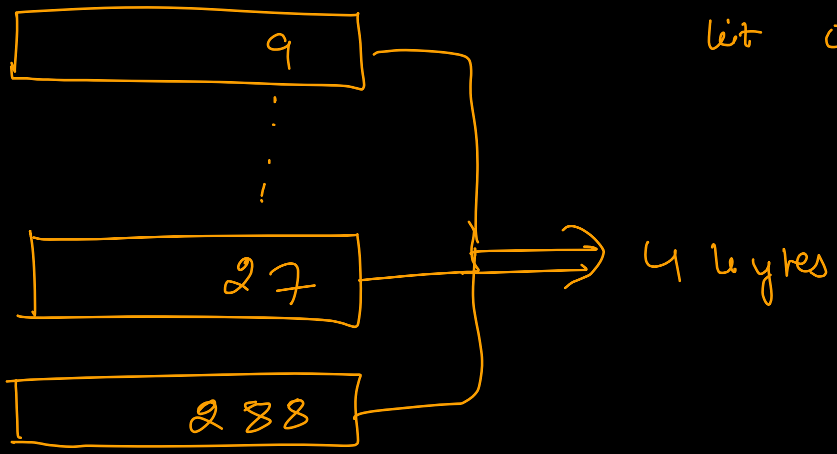
type

item type

Employee



bit 0,1



Changing Data Type

- (array).asType

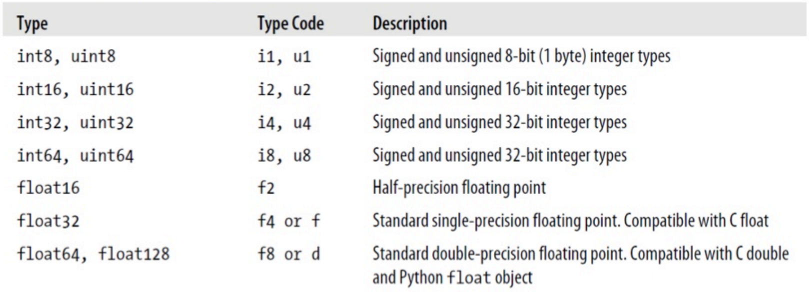


Table 4-2. NumPy data types

Type	Type Code	Description
`int8, uint8`	`i1, u1`	Signed and unsigned 8-bit (1 byte) integer types
`int16, uint16`	`i2, u2`	Signed and unsigned 16-bit integer types
`int32, uint32`	`i4, u4`	Signed and unsigned 32-bit integer types
`int64, uint64`	`i8, u8`	Signed and unsigned 32-bit integer types
`float16`	`f2`	Half-precision floating point
`float32`	`f4 or f`	Standard single-precision floating point. Compatible with C float
`float64, float128`	`f8 or d`	Standard double-precision floating point. Compatible with C double and Python `float` object

Array Operations

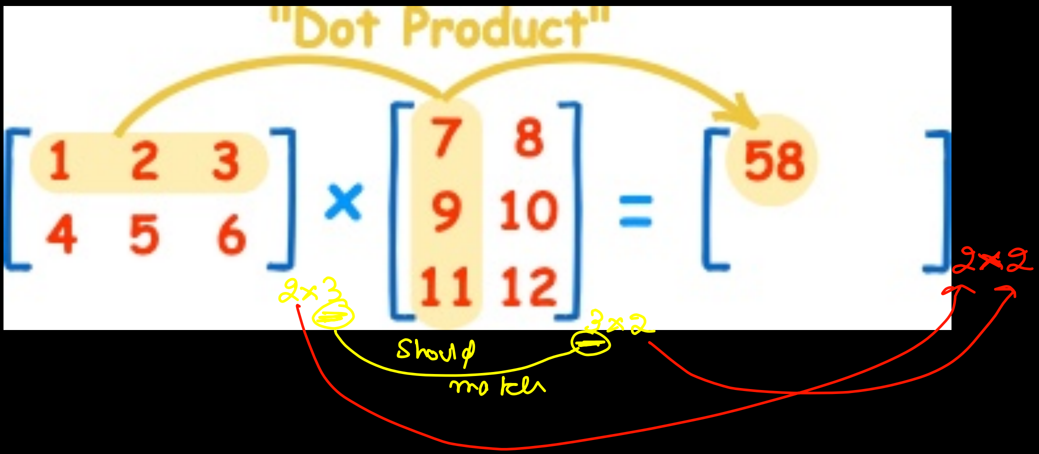
1. Slicing & Indexing

2. Scaler Operation

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} * 2 \Rightarrow \begin{bmatrix}2 & 4\\ 6 & 8\end{bmatrix}$$

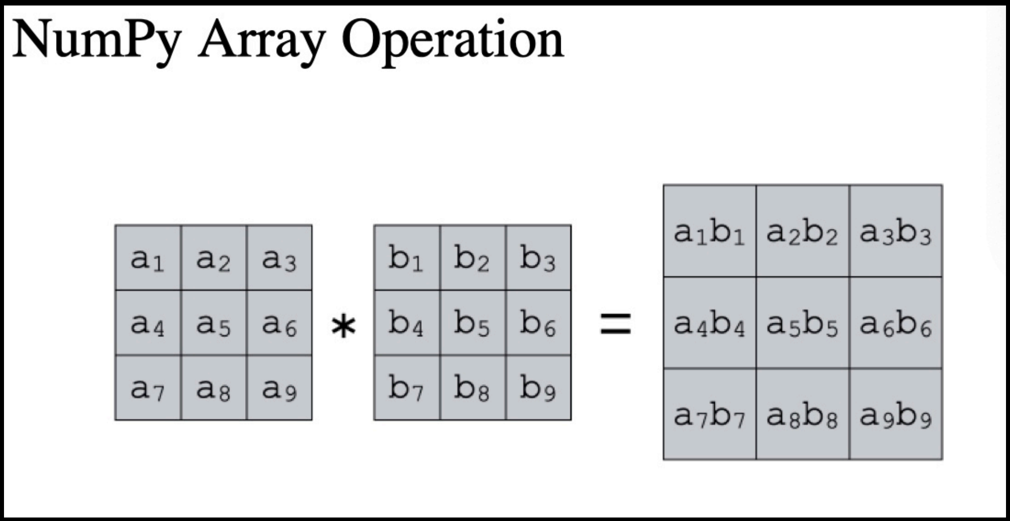
multiply , modulus
add , floor
subtract ,
pow

Relational Operators



"Dot Product"

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \times \begin{bmatrix}7 & 8\\ 9 & 10\\ 11 & 12\end{bmatrix} = \begin{bmatrix}58 & \\ & \end{bmatrix}$$



NumPy Array Operation

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} * \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} a_1b_1 & a_2b_2 & a_3b_3 \\ a_4b_4 & a_5b_5 & a_6b_6 \\ a_7b_7 & a_8b_8 & a_9b_9 \end{bmatrix}$$

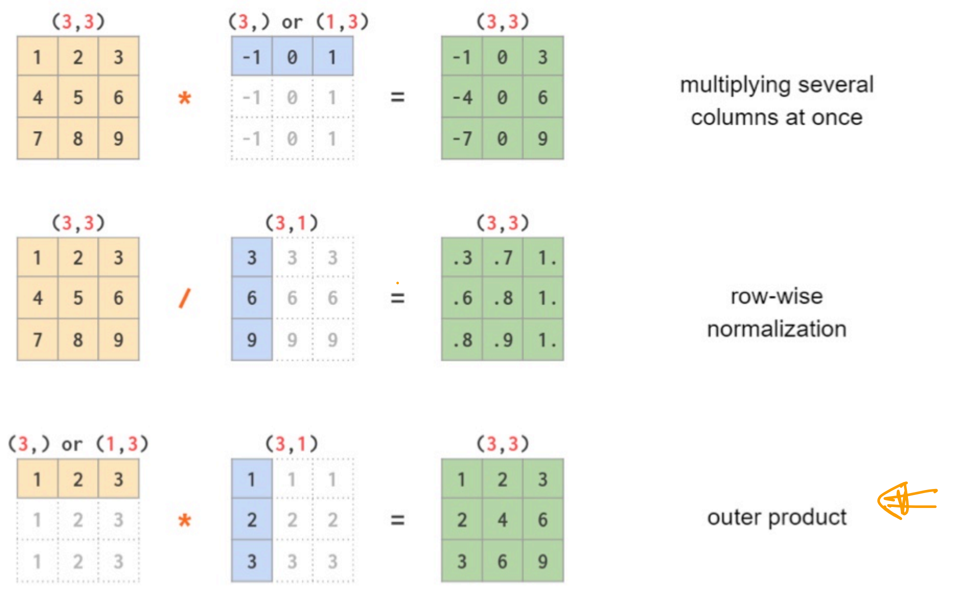
Brodcasting

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * \begin{matrix} [1 \, , 2] \\ [1 \, , 2] \end{matrix}$$

$$\Rightarrow \begin{bmatrix} 0 & 2 \\ & \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 0 & 2 \\ 2 & 6 \end{bmatrix}$$



$$\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix} * \begin{pmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{pmatrix} = \begin{pmatrix}-1&0&3\\-4&0&6\\-7&0&9\end{pmatrix}$$

multiplying several columns at once

$$\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix} / \begin{pmatrix}3&3&3\\6&6&6\\9&9&9\end{pmatrix} = \begin{pmatrix}.3&.7&1.\\.6&.8&1.\\.8&.9&1.\end{pmatrix}$$

row-wise normalization

$$\begin{pmatrix}1&2&3\\1&2&3\\1&2&3\end{pmatrix} * \begin{pmatrix}1&1&1\\2&2&2\\3&3&3\end{pmatrix} = \begin{pmatrix}1&2&3\\2&4&6\\3&6&9\end{pmatrix}$$

outer product ⇐

In dot produc

$$M_{a \times b} \quad N_{c \times d} \quad = O_{b \times d}$$

b shoud be equal to c

& output is b x d

$$\underset{M_{2\times 3}}{\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}} \cdot \underset{N_{3\times 2}}{\begin{pmatrix}4&7\\5&8\\6&9\end{pmatrix}} = \begin{pmatrix}1\cdot4+2\cdot5+3\cdot6 & 1\cdot7+2\cdot8+3\cdot9\\4\cdot4+5\cdot5+6\cdot6 & 4\cdot7+5\cdot8+6\cdot9\end{pmatrix} = \underset{O_{2\times 2}}{\begin{pmatrix}32&50\\77&122\end{pmatrix}}$$

Broadcasting

1. Same shape
2. dim^n is 1
3. The smaller array should be able to be padded on larger array