SKILL<S/ASH>

NUMPY CHEAT SHEET

Becoming Human.AI

SWIPE

WHAT IS NUMPY ?

The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

CREATING ARRAYS

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],dtype = float)
```

INITIAL PLACEHOLDERS

`>>> np.zeros((3,4))`	**Create an array of zeros**
`>>> np.ones((2,3,4),dtype=np.int16)`	**Create an array of ones**
`>>> d = np.arange(10,25,5)`	**Create an array of evenly spaced values (step value)**
`>>> np.linspace(0,2,9)`	**Create an array of evenly spaced values (number of samples)**
`>>> e = np.full((2,2),7)`	**Create a constant array**
`>>> f = np.eye(2)`	**Create a 2X2 identity matrix**
`>>> np.random.random((2,2))`	**Create an array with random values**
`>>> np.empty((3,2))`	**Create an empty array**

I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

INSPECTING YOUR ARRAY

`>>> a.shape`	**Array dimensions**
`>>> len(a)`	**Length of array**
`>>> b.ndim`	**Number of array dimensions**
`>>> e.size`	**Number of array elements**
`>>> b.dtype`	**Data type of array elements**
`>>> b.dtype.name`	**Name of data type**
`>>> b.astype(int)`	**Convert an array to a diferent type**

DATA TYPES

`>>> np.int64`	**Signed 64-bit integer types**
`>>> np.float32`	**Standard double-precision floating point**
`>>> np.complex`	**Complex numbers represented by 128 floats**
`>>> np.bool`	**Boolean type storing TRUE and FALSE**
`>>> np.object`	**Python object type values**
`>>> np.string_`	**Fixed-length string type**
`>>> np.unicode_`	**Fixed-length unicode type**

ASKING FOR HELP

```
>>> np.info(np.ndarray.dtype)
```

ARRAY MATHEMATICS

Arithmetic Operations

```
>>> g = a - b                                  Subtraction
array([[-0.5, 0. , 0. ], [-3. , -3. , -3. ]])
>>> np.subtract(a,b)                           Subtraction
>>> b + a                                      Addition
array([[ 2.5, 4. , 6. ], [ 5. , 7. , 9. ]])
>>> np.add(b,a)                                Addition
>>> a / b                                      Division
array([[ 0.66666667, 1. , 1. ], [ 0.25 , 0.4 , 0.5 ]])
>>> np.divide(a,b)                             Division
>>> a * b                                      Multiplication
array([[ 1.5, 4. , 9. ], [ 4. , 10. , 18. ]])
>>> np.multiply(a,b)                           Multiplication
>>> np.exp(b)                                  Exponentiation
>>> np.sqrt(b)                                 Square root
>>> np.sin(a)                                  Print sines of an array
>>> np.cos(b)                                  Element-wise cosine
>>> np.log(a)                                  Element-wise natural logarithm
>>> e.dot(f)                                   Dot product
array([[ 7., 7.],
```

COMPARISON

```
>>> a == b                                     Element-wise comparison
    array([[False, True, True],
           [False, False, False]], dtype=bool)
>>> a < 2                                      Element-wise comparison
    array([True, False, False], dtype=bool)
>>> np.array_equal(a, b)                       Array-wise comparison
```

AGGREGATE FUNCTIONS

```
>>> a.sum()
>>> a.min()
>>> b.max(axis=0)
>>> b.cumsum(axis=1)
>>> a.mean()
>>> b.median()
```

Array-wise sum
Array-wise minimum value
Maximum value of an array row
Cumulative sum of the elements
Mean
Median

COPYING ARRAYS

```
>>> h = a.view()
>>> np.copy(a)
>>> h = a.copy()
```

Create a view of the array with the same data
Create a copy of the array
Create a deep copy of the array

SORTING ARRAYS

```
>>> a.sort()
>>> c.sort(axis=0)
```

Sort an array
Sort the elements of an array's axis

SUBSETTING, SLICING, INDEXING

SUBSETTING

```
>>> a[2]
3
>>> b[1,2]
6.0
```

Select the element at the 2nd index

Select the element at row 1 column 2 (equivalent to b[1][2])

Slicing

```
>>> a[0:2]
array([1, 2])
```
Select items at index 0 and 1

```
>>> b[0:2,1]
array([ 2., 5.])
```
Select items at rows 0 and 1 in column 1

```
>>> b[:1]
array([[1.5, 2., 3.]])
```
Select all items at row 0
(equivalent to b[0:1, :])

```
>>> c[1,...]
array([[[ 3., 2., 1.],
[ 4., 5., 6.]]])
```
Same as [1,:,:]

```
>>> a[ : :-1]
array([3, 2, 1])
```
Reversed array a

BOOLEAN INDEXING

```
>>> a[a<2]
array([1])
```
Select elements from a less than 2

FANCY INDEXING

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]
array([ 4. , 2. , 6. , 1.5])
```
Select elements (1,0),(0,1),(1,2) and (0,0)

```
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]
array([[ 4. ,5. , 6. , 4. ],
[ 1.5, 2. , 3. , 1.5],
[ 4. , 5. , 6. , 4. ],
[ 1.5, 2. , 3. , 1.5]])
```
Select a subset of the matrix's rows and columns

ARRAY MANIPULATION

TRANSPOSING ARRAY

```
>>> i = np.transpose(b)
>>> i.T
```
Permute array dimensions
Permute array dimensions

Adding/Removing Elements

```
>>> h.resize((2,6))
>>> np.append(h,g)
>>> np.insert(a, 1, 5)
>>> np.delete(a,[1])
```

Return a new array with shape (2,6)
Append items to an array
Insert items in an array
Delete items from an array

SPLITTING ARRAYS

```
>>> np.hsplit(a,3)
[array([1]),array([2]),array([3])] index
>>> np.vsplit(c,2) Split the array
[array([[[ 1.5, 2. , 1. ], [ 4. , 5. , 6. ]]]),
```

Split the array
horizontally at the 3rd
vertically at the 2nd index

CHANGING ARRAY SHAPE

```
>>> b.ravel()
>>> g.reshape(3,-2)
```

Flatten the array
Reshape, but don’t change data

COMBINING ARRAYS

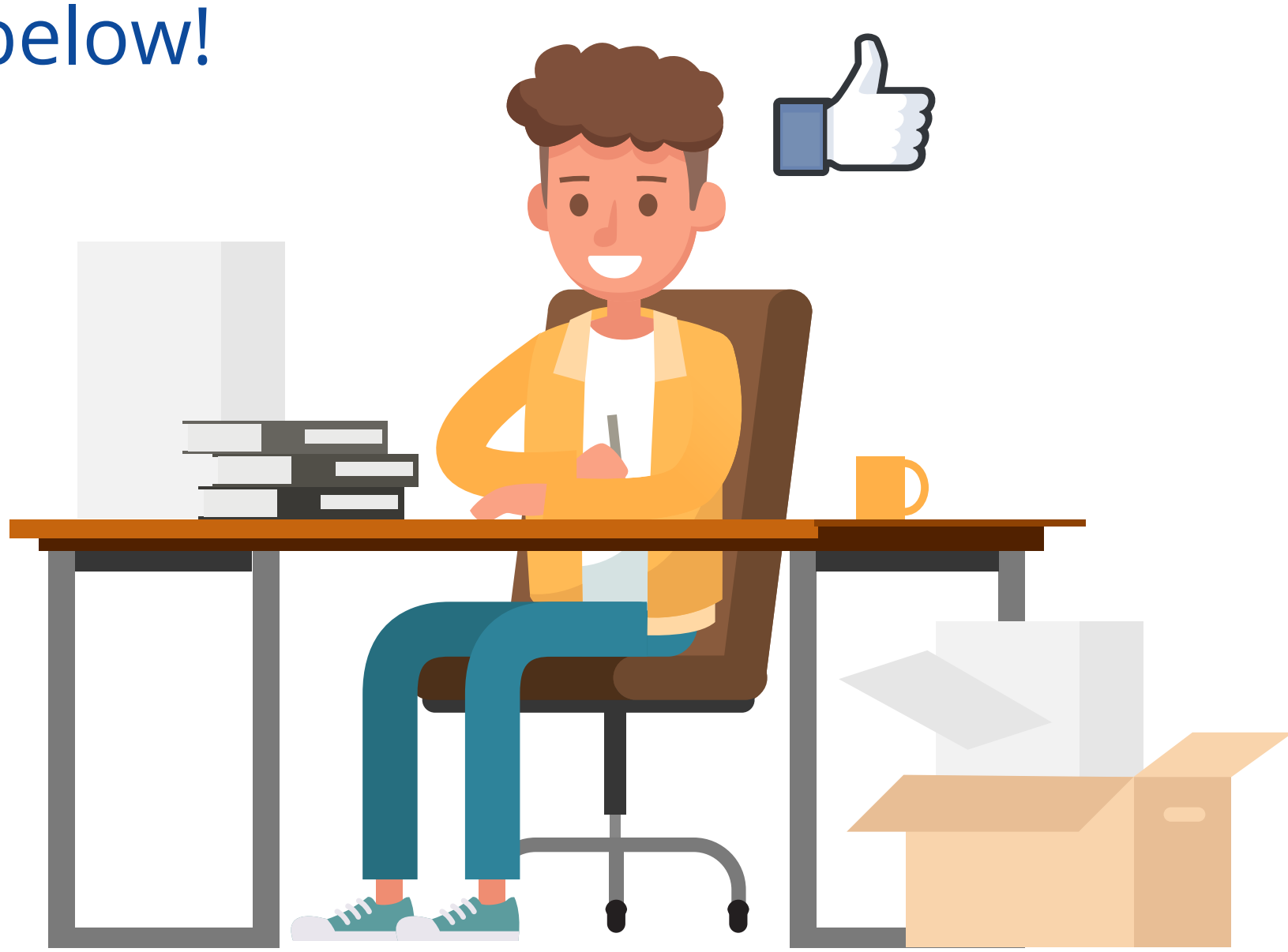
```
>>> np.concatenate((a,d),axis=0)
array([ 1, 2, 3, 10, 15, 20])
>>> np.vstack((a,b))
array([[ 1. , 2. , 3. ],
[ 1.5, 2. , 3. ],
[ 4. , 5. , 6. ]])
>>> np.r_[e,f]
>>> np.hstack((e,f))
array([[ 7., 7., 1., 0.],
[ 7., 7., 0., 1.]])
>>> np.column_stack((a,d))
array([[ 1, 10],
[ 2, 15],
[ 3, 20]])
>>> np.c_[a,d]
```

Concatenate arrays
Stack arrays vertically (row-wise)
Stack arrays vertically (row-wise)
Stack arrays horizontally (column-wise)
Create stacked column-wise arrays
Create stacked

SKILL<S/ASH>

Did you like this post?

Tell us in the comments below!



Drop a follow for more such valuable content!