C# For Data Science

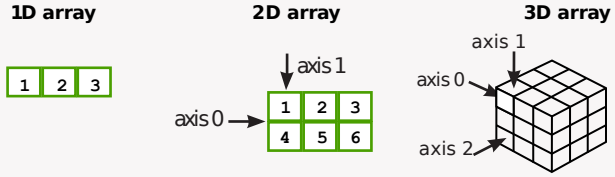
NumSharp Basics - Cheat Sheet

Brought to you by https://3rdman.de

(https://github.com/SciSharp/NumSharp)

NumSharp is a C# port of NumPy, targetting .NET Standard. It therefore works on multiple platforms and is a fundamental package needed for scientific computing with C#.

NumPy Arrays



Creating Arrays

```
NDArray a = new double[]{1,2,3};
NDArray b = new double[,]{{1,2,3},{4,5,6}};
NDArray c = new double[,,]{{{1.5,2,3}, {4,5,6}},
                           {{3,2,1}, {4,5,6}}};
```

Initial Placeholders

`var a = np.zeros((3,4));`	Create an array of zeros
`var a = np.ones((2,3,4), np.int16);`	Create an array of ones
`var d = np.arange(10,25,5);`	Create an array of evenly spaced values (step value)
`var b = np.linspace(0,2,9);`	Create an array of evenly spaced values (number of samples)
`var e = np.full(7, (2, 2));`	Create a constant array
`var f = np.eye(2);`	Create a 2X2 identity matrix
`NDArray a = np.random .randint(0, 100, (2,2)) .astype(typeof(double));`	Create an array with random int values
`var a = np.empty((3,2));`	Create an empty array

I/O

Saving & Loading On Disk

```
np.Save((Array)a, "my_array.npy");
NDArray z = np.Load<double[]>("my_array.npy");
var dict = new Dictionary<string, Array>();
dict.Add("A", (Array)a);
dict.Add("B", (Array)b);
np.Save_Npz(dict, "array.npz");
NpzDictionary<Array> load_dict =
                    np.Load_Npz<Array>("array.npz");
```

Data Types

`np.bool_, np.bool8, np.@bool`	System.Boolean
`np.@byte, np.uint8, np.ubyte`	System.Byte
`np.int16, np.uint16`	System.Int16, System.UInt16
`np.int32, np.uint32`	System.Int32, System.UInt32
`np.int_, np.int64, np.int0`	System.Int64
`np.uint64, np.uint0, np.@uint`	System.UInt64
`np.float32`	System.Single
`np.float_, np.float64, np.@double`	System.Double
`np.@decimal`	System.Decimal
`np.@char`	System.Char
`np.complex_, np.complex64, np.complex128`	System.Numerics.Complex

Inspecting Your Array

`c.shape`	Array dimensions
`c.shape[0]`	Length of array
`c.ndim`	Number of array dimensions
`c.size`	Number of array elements
`c.dtype`	Data type of array elements
`c.dtype.ToString()`	Name of data type
`d = c.astype(np.int16);`	Convert an array to a different type

Asking For Help

```
https://scisharp.github.io/NumSharp/api/index.html
```

Array Mathematics

Arithmetic Operations

```
NDArray a = new double[,]{{1,2,3},{4,5,6}};
NDArray b = new double[,]{{3,2,1},{6,5,4}};
NDArray c = new double[]{3,2,1};
```

`var g = a - b; g.ToString();` `[[-2, 0, 2], [-2, 0, 2]]`	Subtraction
`np.subtract(a,b);`	Subtraction
`(b + a).ToString();` `[[4, 4, 4], [10, 10, 10]]`	Addition
`np.add(b,a);`	Addition
`(a / b).ToString();` `[[0.3333333333333333, 1, 3], [0.6666666666666666, 1, 1.5]]`	Division
`np.divide(a,b);`	Division
`(a * b).ToString();` `[[3, 4, 3], [24, 25, 24]]`	Multiplication
`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
`a.dot(c).ToString();` `[10, 28]`	Dot product

Comparison

`(a == b).ToString();` `[[False, True, False], [False, True, False]]`	Element-wise comparison
a < 2, a > 3 etc.	Element-wise comparison ! NumSharp does not support all comparison operators. Check out the notebook (see end of the cheat sheet) for a potential workaround.
`np.array_equal(a, b);`	Array-wise comparison

Aggregate Functions

`a.sum();`	Array-wise sum
`a.min();`	Array-wise minimum value
`a.max(0);`	Maximum value of an array row
`b.cumsum(axis:1);`	Cumulative sum of the elements
`a.mean();`	Mean
`np.std(b);`	Standard deviation

Copying Arrays

`var h = a.view();`	Create a view of the array with the same data
`np.copy(a);`	Create a copy of the array
`var h = a.copy();`	Create a deep copy of the array

Sorting Arrays

NumSharp does not seem to support sort(). It provides argsort for one dimensional arrays however.

```
var x = np.array(new double[] {3.0, 1.0, 2.0});
var i = x.argsort<double>();
i.ToString()
                        [1, 2, 0]
```

Subsetting, Slicing, Indexing

```
NDArray a = new double[]{1,2,3};
NDArray b = new double[,]{{1.5, 2, 3}, {4, 5.2, 6}, {0, -2, 2.3}};
NDArray c = new double[,,]{{{1.5,2,3}, {4,5,6}}, {{3,2,1}, {4,5,6}}};
```

Subsetting

`a[2].ToString();` `3`	Select the element at the 2nd index
`b[1,2].ToString();` `6`	Select the element at row 1 column 2(same as `[1][2]`)

Slicing

`b["0:2"].ToString();` `[[1.5, 2, 3], [4, 5.2, 6]]`	Select items at index 0 and 1
`b["0:2,1"].ToString();` `[2, 5.2]`	Select items at rows 0 and 1, in column 1
`b[":1"].ToString();` `[[1.5, 2, 3]]`	Select all items at row 0
`c["1,..."].ToString();` `[[3, 2, 1], [4, 5, 6]]`	Same as `[1, :, :]`
`a["::-1"].ToString()` `[3, 2, 1]`	Reversed array a

Array Manipulation

Transposing Array

`np.transpose(b).ToString();` `[[1.5, 4, 0], [2, 5.2, -2], [3, 6, 2.3]]`	Permute array dimensions

Changing Array Shape

`b.ravel().ToString();` `[1.5, 2, 3, 4, 5.2, 6, 0, -2, 2.3]`	Flatten the array
`b.reshape(9,1).ToString();` `[[1.5], [2], [3], [4], [5.2], [6], [0], [-2], [2.3]]`	Reshape, but don't change data

Combining Arrays

```
NDArray a = new double[]{1,2,3};
NDArray b = new double[,]{{1.0, 1.1, 1.2},{2.1, 2.2, 2.3}};
NDArray c = new double[,]{{3.2, 3.3, 3.4},{4.3, 4.4, 4.5}};
```

`np.concatenate((a, new double[] {9, 8, 7})).ToString();` `[1, 2, 3, 9, 8, 7]`	Concatenate arrays
`np.vstack(c, b).ToString();` `[[3.2, 3.3, 3.4], [4.3, 4.4, 4.5], [1, 1.1, 1.2], [2.1, 2.2, 2.3]]`	Stack arrays vertically (row-wise)
`np.hstack(b, c).ToString();` `[[1, 1.1, 1.2, 3.2, 3.3, 3.4], [2.1, 2.2, 2.3, 4.3, 4.4, 4.5]]`	Stack arrays horizontally (column-wise)

Please visit the GitHub project site https://github.com/indy-3rdman/numsharp-cheatsheet for more details and to try the interactive notebook.