numerical operations on numpy arrays

Elementwise operations

Basic operation with scalars:

```
In [1]: # sum of a scalar
        import numpy as np
        a = np.array([1, 2, 3, 4])
        a + 1

Out[1]: array([2, 3, 4, 5])
```

```
In [2]: # division by a scalar
        a/2

Out[2]: array([ 0.5,  1. ,  1.5,  2. ])
```

```
In [3]: # exponentiation
        2**a, a**2

Out[3]: (array([ 2,  4,  8, 16]), array([ 1,  4,  9, 16]))
```

Elementwise operations (2)

All arithmetic operates elementwise:

```
In [4]: # difference of 2 arrays
        b = np.ones(4) + 1 # b = array([2.,2.,2.,2.])
        a - b

Out[4]: array([-1.,  0.,  1.,  2.])
```

```
In [5]: # multiplication of 2 arrays
        a * b

Out[5]: array([ 2.,  4.,  6.,  8.])
```

```
In [6]: # a more complex operation
        j = np.arange(5)
        2**(j + 1) - j

Out[6]: array([ 2,  3,  6, 13, 28])
```

Elementwise operations (3)

Warning: 2D *array multiplication is not matrix multiplication:*

```
In [7]: c = np.ones((3, 3))
        c * c                   # NOT matrix multiplication!
```

```
Out[7]: array([[ 1.,  1.,  1.],
               [ 1.,  1.,  1.],
               [ 1.,  1.,  1.]])
```

Matrix multiplication:

```
In [8]: c.dot(c)
```

```
Out[8]: array([[ 3.,  3.,  3.],
               [ 3.,  3.,  3.],
               [ 3.,  3.,  3.]])
```

Other elementwise operations

Comparisons:

```
In [9]: a = np.array([1, 2, 3, 4])
        b = np.array([4, 2, 2, 4])
        a == b
```

```
Out[9]: array([False,  True, False,  True], dtype=bool)
```

```
In [10]: a > b
```

```
Out[10]: array([False, False,  True, False], dtype=bool)
```

Other elementwise operations (2)

Logical operations:

```
In [11]: # the truth value of a OR b element-wise
         a = np.array([1, 1, 0, 0], dtype=bool)
         b = np.array([1, 0, 1, 0], dtype=bool)
         np.logical_or(a, b)
```

```
Out[11]: array([ True,  True,  True, False], dtype=bool)
```

```
In [12]: # the truth value of a AND b element-wise
         np.logical_and(a, b)
```

```
Out[12]: array([ True, False, False, False], dtype=bool)
```

Other elementwise operations (3)

Transcendental functions:

```
In [13]: a = np.linspace(1,10,10)
         np.sin(a)

Out[13]: array([ 0.84147098,  0.90929743,  0.14112001, -0.7568025 , -0.95892427,
                -0.2794155 ,  0.6569866 ,  0.98935825,  0.41211849, -0.5440211
         1])
```

```
In [14]: np.log(a)

Out[14]: array([ 0.        ,  0.69314718,  1.09861229,  1.38629436,  1.60943791,
                 1.79175947,  1.94591015,  2.07944154,  2.19722458,  2.3025850
         9])
```

```
In [15]: np.exp(a)

Out[15]: array([  2.71828183e+00,   7.38905610e+00,   2.00855369e+01,
                  5.45981500e+01,   1.48413159e+02,   4.03428793e+02,
                  1.09663316e+03,   2.98095799e+03,   8.10308393e+03,
                  2.20264658e+04])
```

Transposition:

```
In [16]: # An upper triangular array
         a = np.triu(np.ones((3, 3)), 1)   # see help(np.triu)
         a

Out[16]: array([[ 0.,  1.,  1.],
                [ 0.,  0.,  1.],
                [ 0.,  0.,  0.]])
```

```
In [17]: # Transpose of the array a
         a.T

Out[17]: array([[ 0.,  0.,  0.],
                [ 1.,  0.,  0.],
                [ 1.,  1.,  0.]])
```

Extrema

```
In [18]: x = np.array([1, 3, 2])
         x.min() # the minimum value

Out[18]: 1
```

```
In [19]: x.max() # the maximum value

Out[19]: 3
```

```
In [20]: x.argmin()  # index of minimum

Out[20]: 0
```

```
In [21]: x.argmax()  # index of maximum
```

```
Out[21]: 1
```

Logical operations:

```
In [22]: np.all([True, True, False])
```

```
Out[22]: False
```

```
In [23]: np.any([True, True, False])
```

```
Out[23]: True
```

Can be used for array comparisons:

```
In [24]: a = np.zeros((100, 100))
         np.any(a != 0)
```

```
Out[24]: False
```

```
In [25]: np.all(a == a)
```

```
Out[25]: True
```

Array comparisons: another example

```
In [26]: a = np.array([1, 2, 3, 2])
         b = np.array([2, 2, 3, 2])
         c = np.array([6, 4, 4, 5])
         ((a <= b) & (b <= c)).all()
```

```
Out[26]: True
```

Sorting data

Sorting along an axis:

```
In [27]: a = np.array([[4, 3, 5], [1, 2, 1]])
         a
```

```
Out[27]: array([[4, 3, 5],
                [1, 2, 1]])
```

In [28]:
```
# Sort inplace
a.sort(axis=1)
a
```

Out[28]:
```
array([[3, 4, 5],
       [1, 1, 2]])
```