numpy

Ben Bolker

09:47 27 February 2017

```
x = [1,2,3]
type(x)
# <class 'list'>
a = np.array(x)
type(a)
# <class 'numpy.ndarray'>
a.dtype
# dtype('int64')
```

Variable types are `bool` < `int` < `float` < `str`. Mixed lists automatically get converted to the highest type:

```
y = [1, "a", 3]
np.array(y)
# array(['1', 'a', '3'], dtype='<U21')
np.array([1.0,2,4])
# array([ 1.,  2.,  4.])
np.array([1.0,2,4]).dtype
# dtype('float64')
```

Indexing and slicing:

```
z = np.array([1.0,2,3])
z[1]
# 2.0
z[:-2]
# array([ 1.])
len(z)
# 3
```

Vectorized arithmetic:

```
z
# array([ 1.,  2.,  3.])
z*3
# array([ 3.,  6.,  9.])
z + 1
# array([ 2.,  3.,  4.])
z += 1
z
# array([ 2.,  3.,  4.])
z + z
# array([ 4.,  6.,  8.])
```

In contrast, multiplying a list copies the whole list:

```
L = [1,2,3]
L*3
# [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

We get an error if we try to combine to unequally sized arrays:

```
z + np.array([1,2])
# Traceback (most recent call last):
#   File "<stdin>", line 1, in <module>
# ValueError: operands could not be broadcast together with shapes (3,) (2,)
```

Arrays are mutable:

```
z
# array([ 2.,  3.,  4.])
z[0] = 5
z
# array([ 5.,  3.,  4.])
```

When we stick items into arrays they can be cast to higher types, but trying to put a higher-type item into a lower-type array fails:

```
zz = np.array(["hello","goodbye"])
# zz[0] = 1.05
zz
# array(['1.05', 'goodbye'], dtype='<U7')
z[0] = "a"
# Traceback (most recent call last):
#   File "<stdin>", line 1, in <module>
# ValueError: could not convert string to float: 'a'
```