python™

Argelander-Institut für Astronomie

Python for Astronomers

more on numpy

Exercise

The file ~rschaaf/mobydick.txt contains the complete text of Herman Melville's "Moby Dick" (Thanks to the Project Gutenberg www.gutenberg.org).

Find out which the 10 most frequent words in the novel are. Please make sure that your statistics is not affected by interpunction and letter case!

Last time: Identity

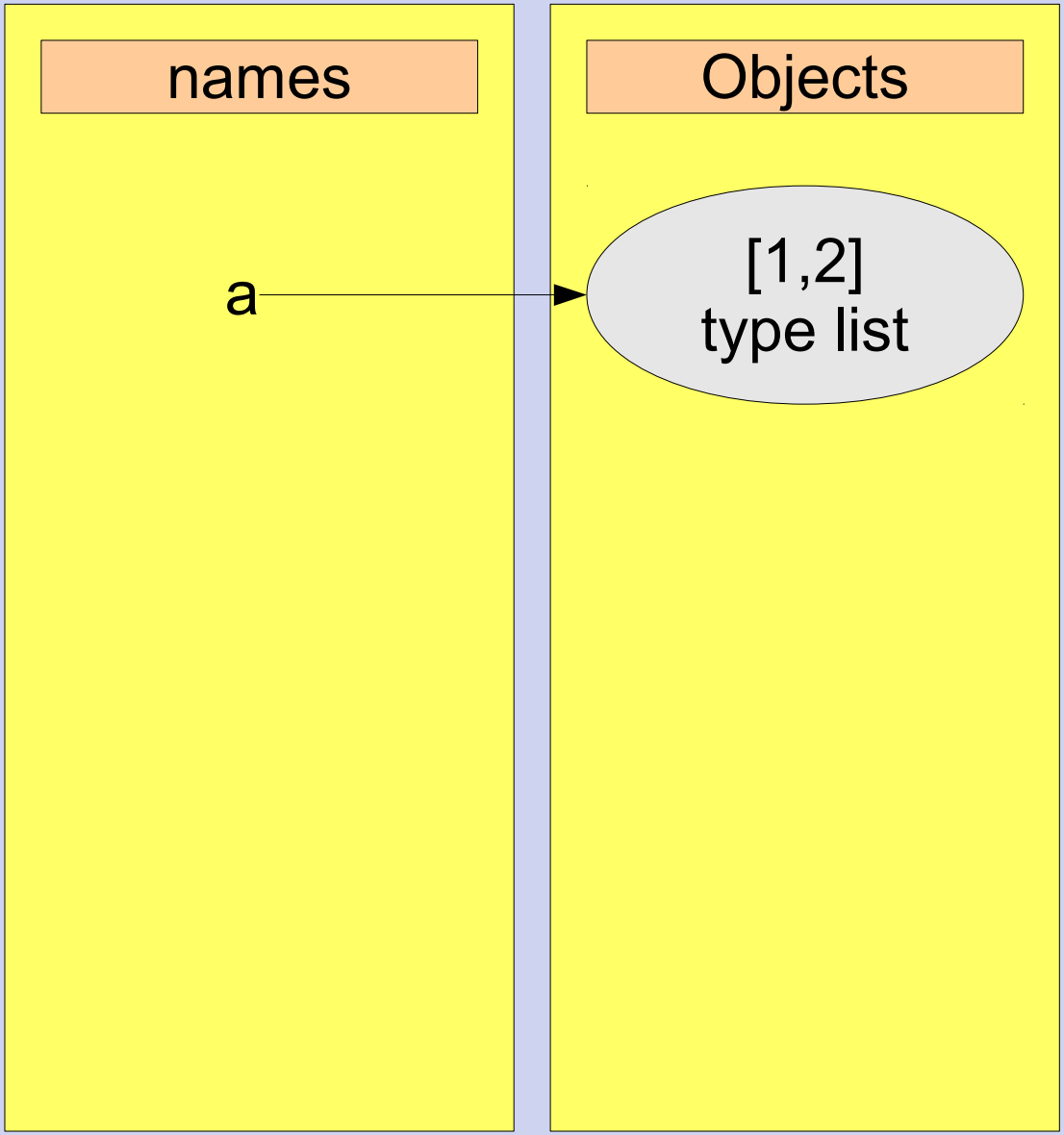
```
>>> a = [1,2,3]
>>> b = a[:]     # creates a copy of a
>>> a is b       # False
>>> a[0] = 42    # b not changes
>>>
>>> import numpy
>>> a = numpy.array([1,2,3])
>>> b = a[:]     # create a copy of a
>>> a is b       # False
>>> a[0] = 42    # b is changed,but why?
```

Identity & Copy

```
>>> a = [[1,2],[3,4]]
>>> b = a[:]          # create a copy of a
>>> a is b            # False
>>> a[0][0]=42        # b changed !???
```

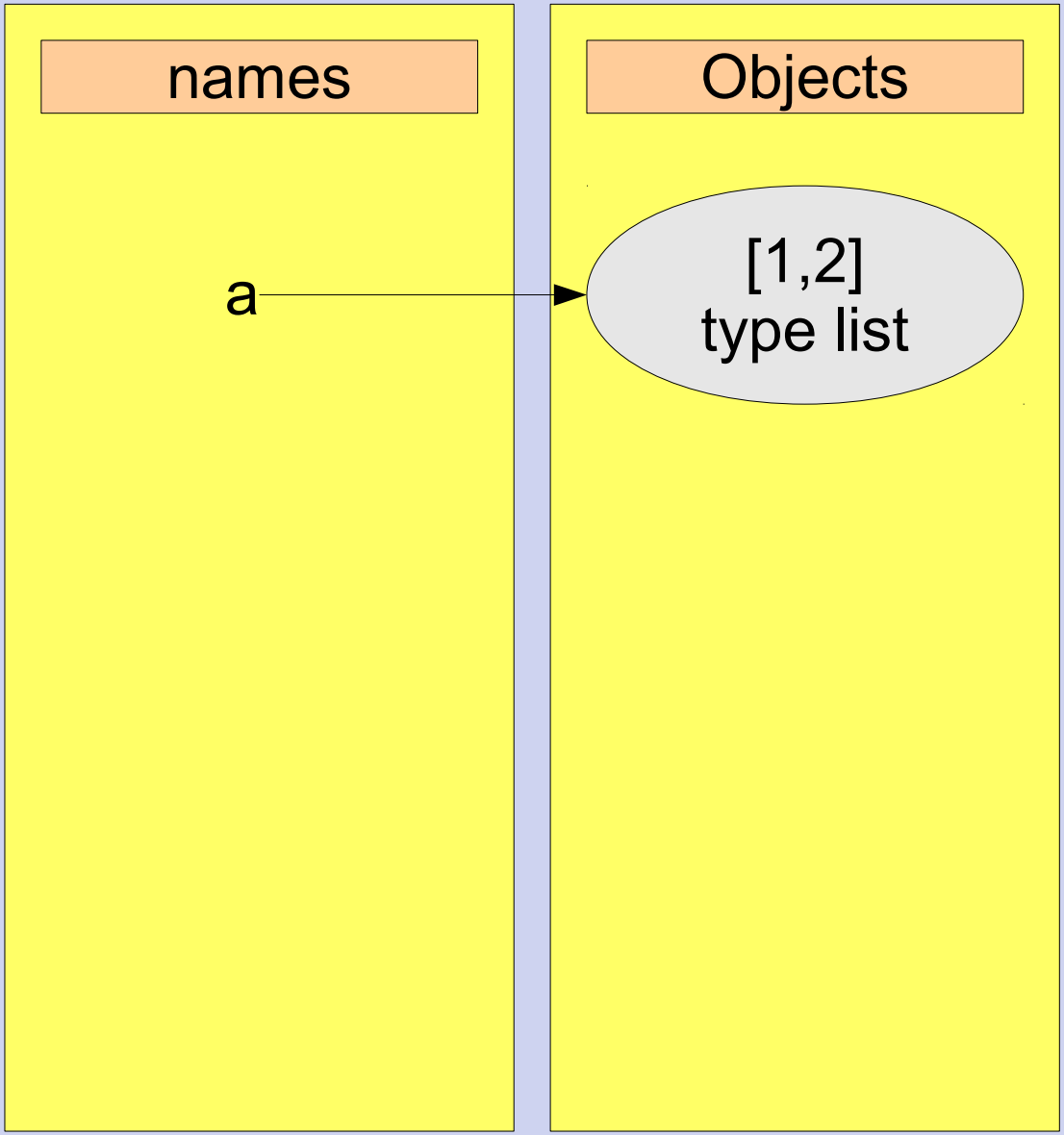
Identity & copy

```
>>> a = [1,2]
```



Identity & copy

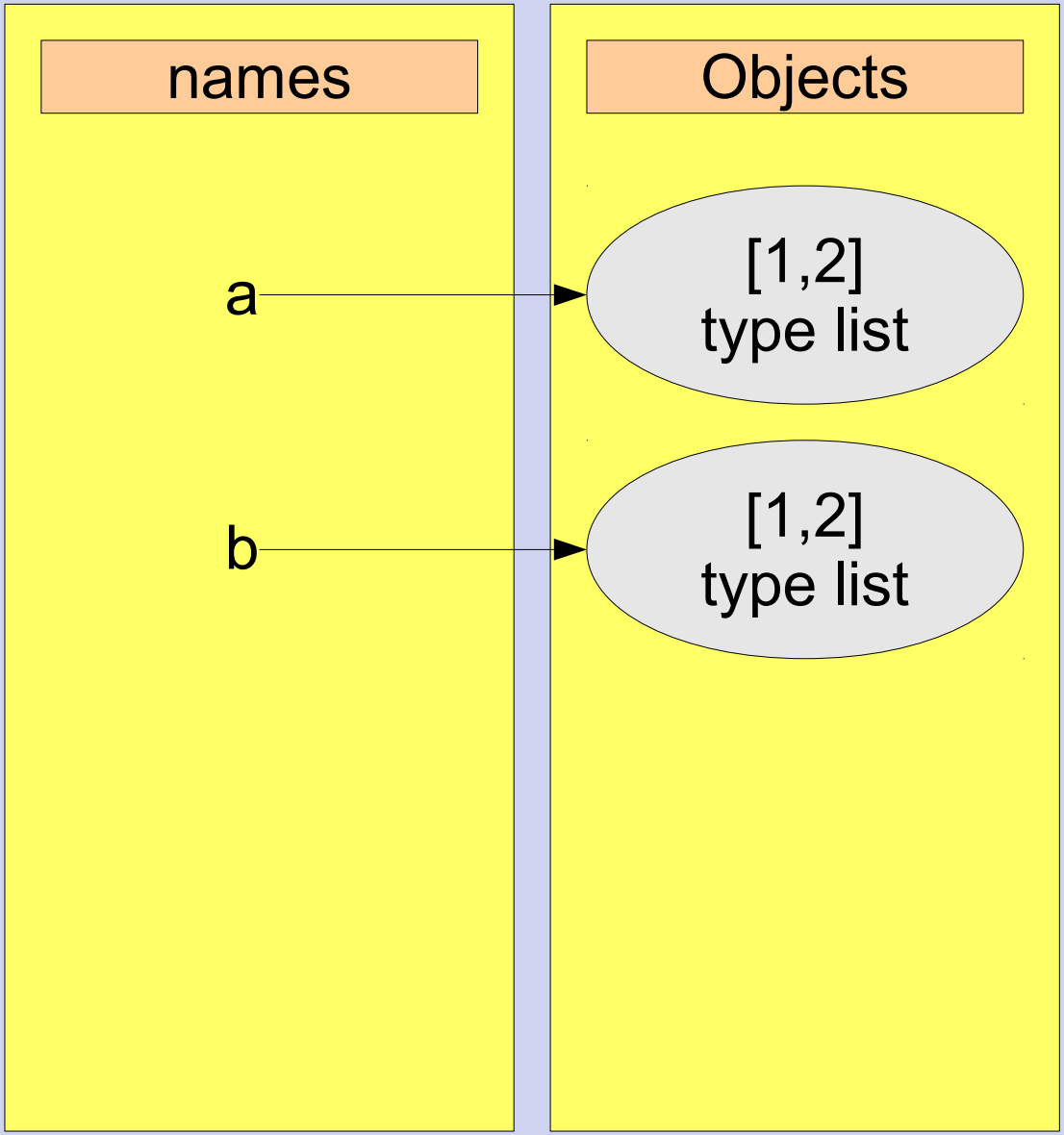
```
>>> a = [1,2]
```



Identity & copy

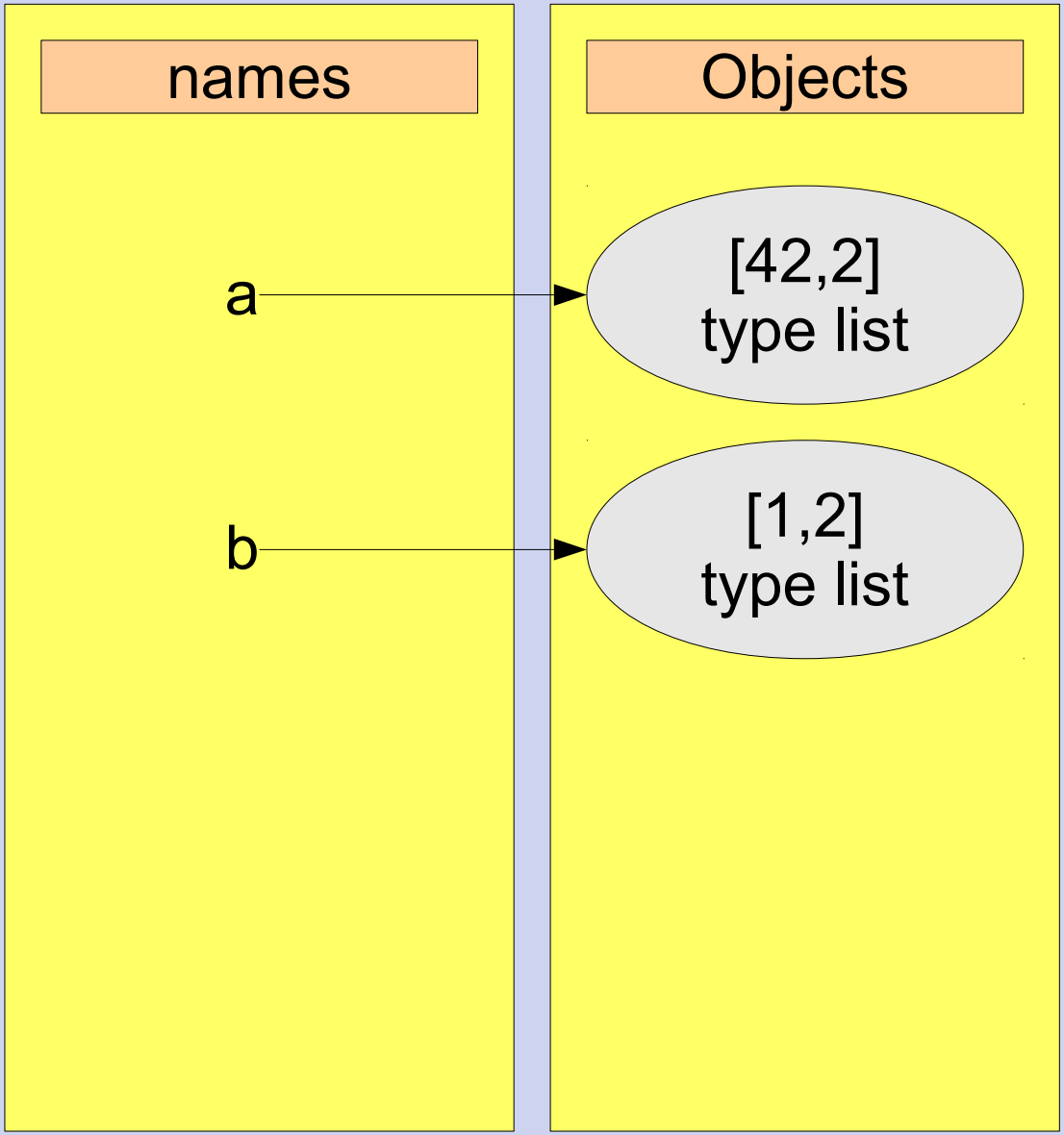
```
>>> a = [1,2]
```

```
>>> b = a[:]
```



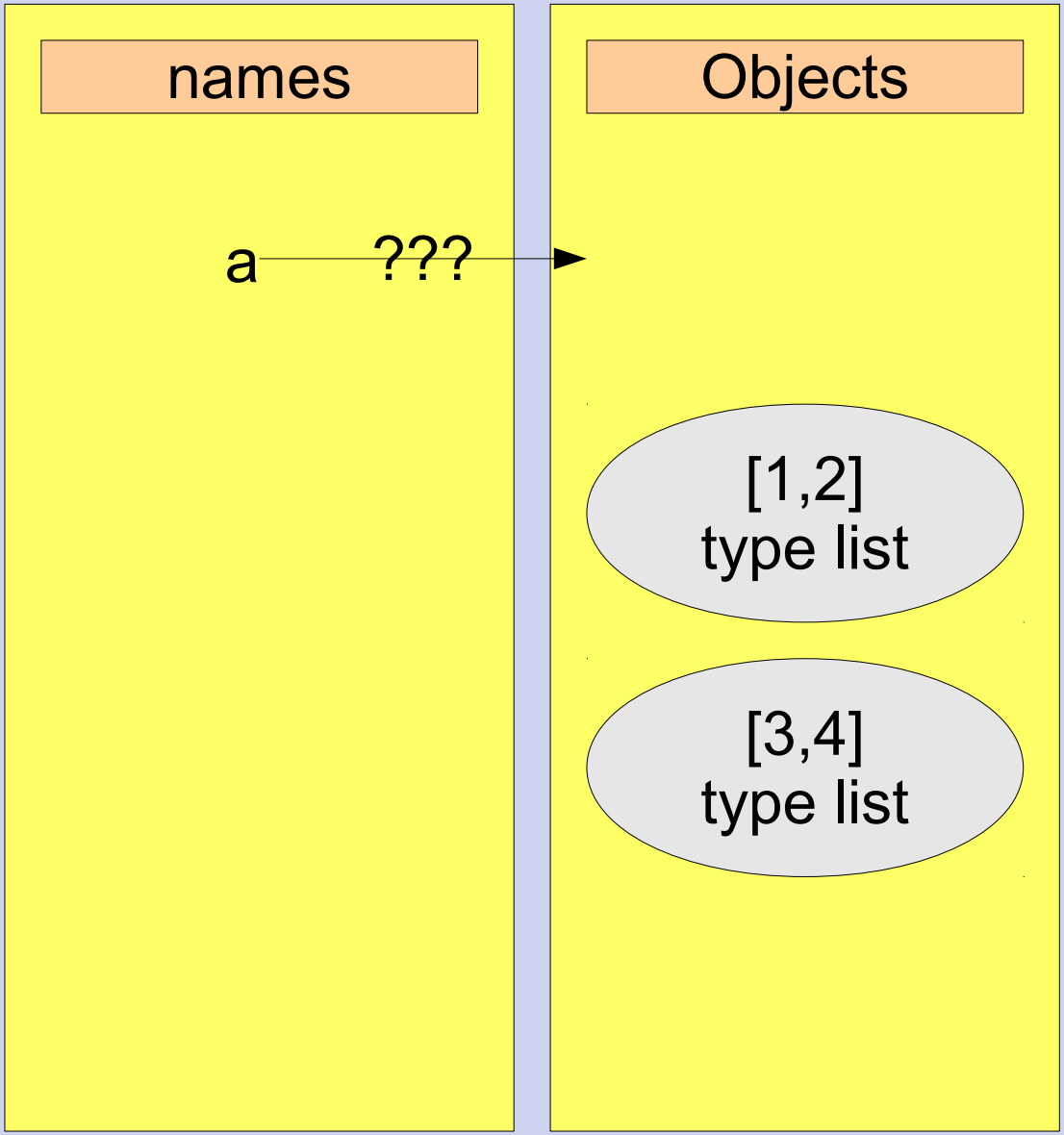
Identity & copy

```
>>> a = [1,2]

>>> b = a[:]

>>> a[0] = 42
```



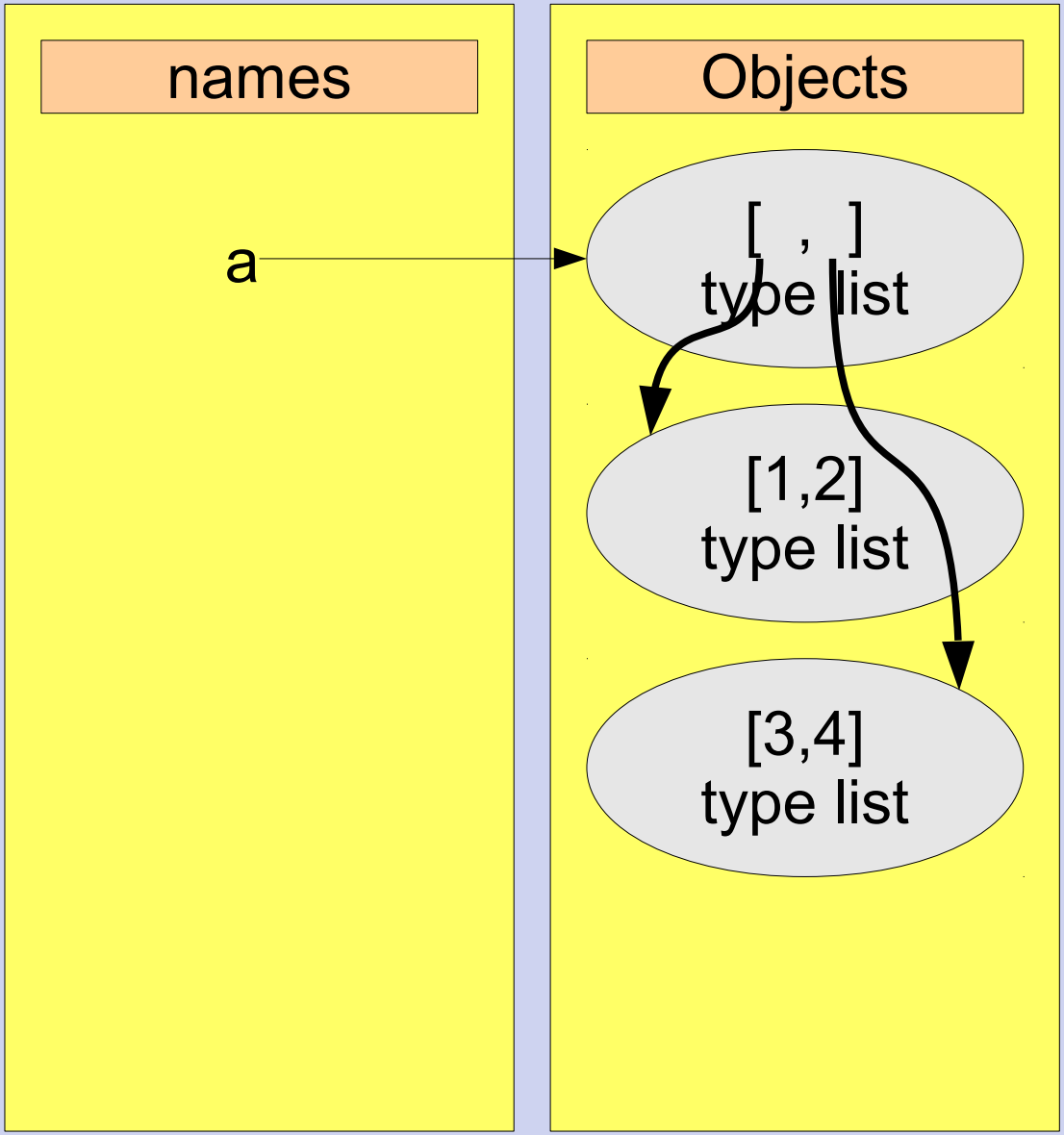
Identity & copy

```
>>> a = [[1,2],
...       [3,4]]
```



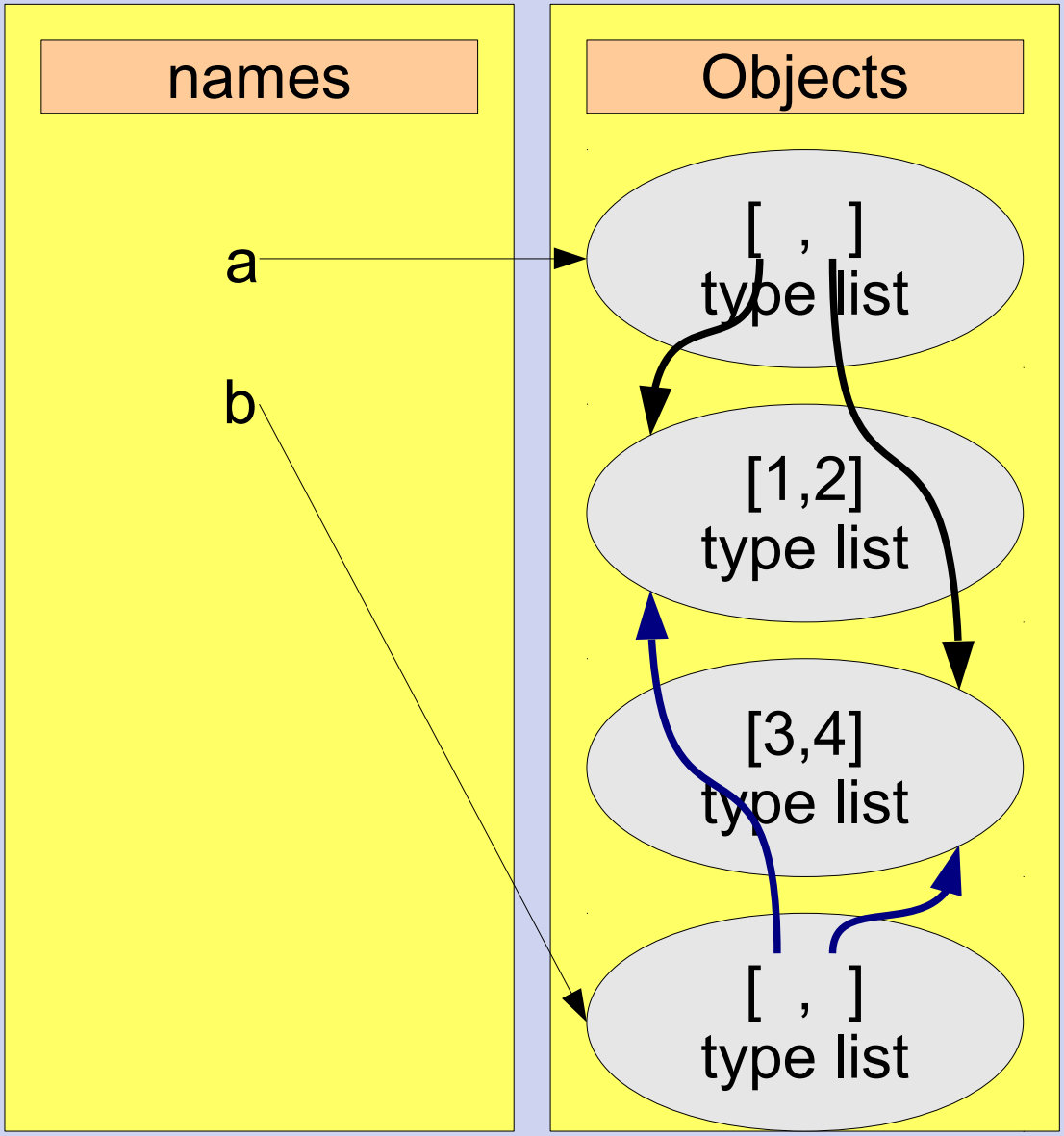
Identity & copy

```
>>> a = [[1,2],
...       [3,4]]
```



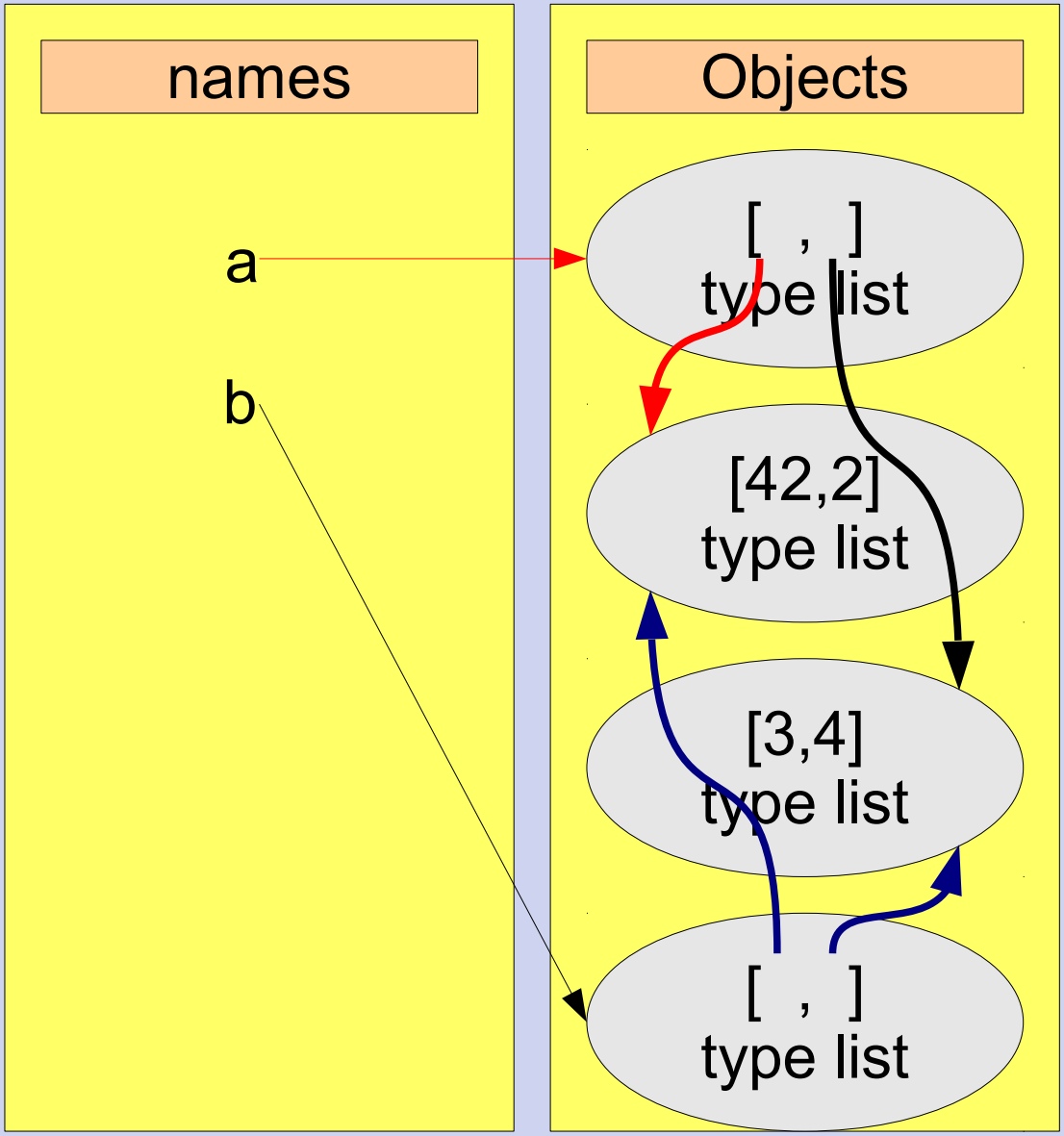
Identity & copy

```
>>> a = [[1,2],
...       [3,4]]

>>> b = a[:]
```



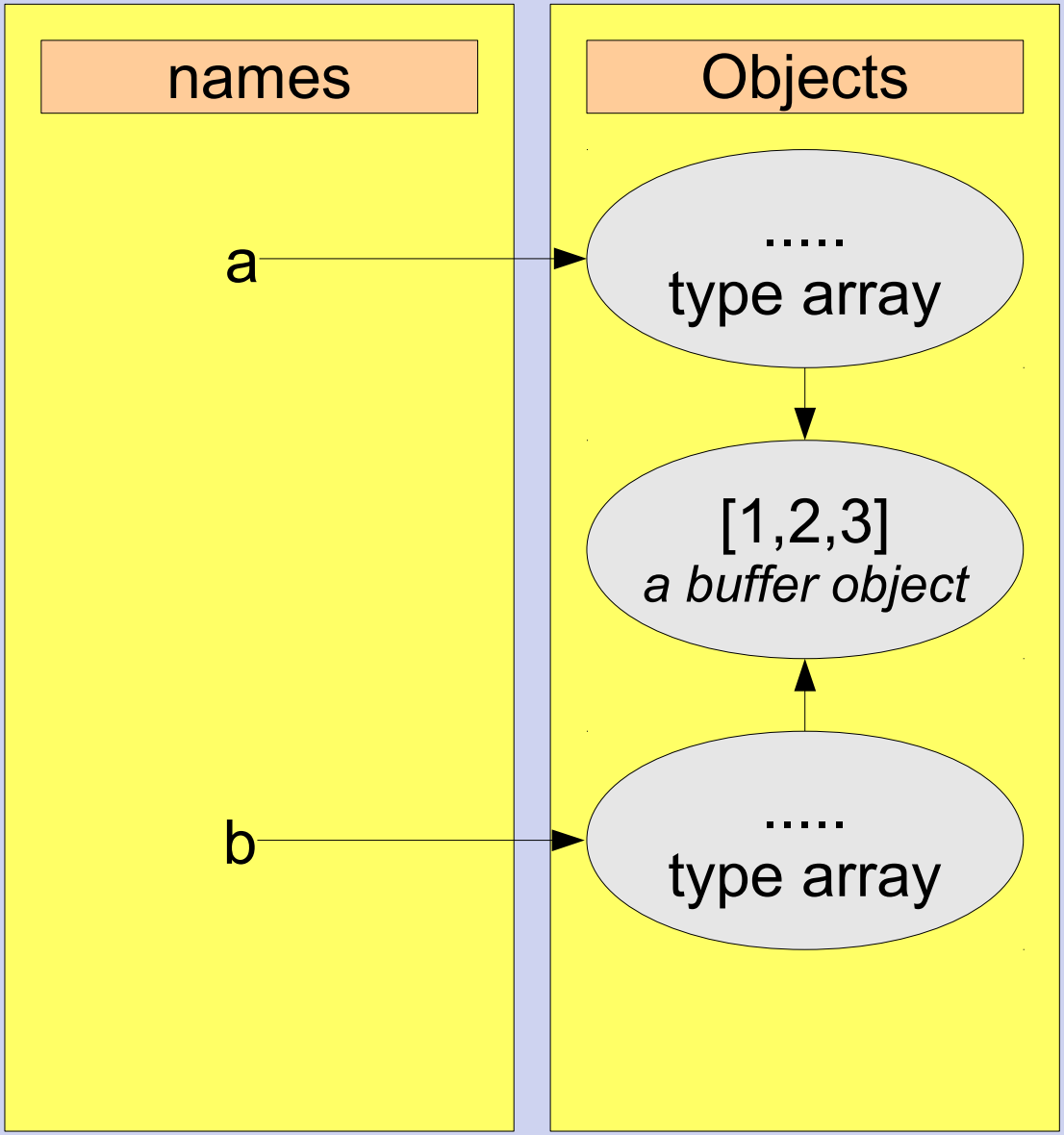
Identity & copy

```
>>> a = [[1,2],
...       [3,4]]

>>> b = a[:]

>>> a[0][0] = 42
```



Identity & Copy

```
>>> # simplified !
>>>
>>> a = numpy.array(
>>>            [1,2,3])
>>> b = a[:]
```



Creating arrays

```
>>> import numpy
>>> a1 = numpy.array([1,2,3])
>>> a2 = numpy.zeros(10)
>>> a3 = numpy.ones((2,2))
>>> a4 = numpy.empty((3,3,3))
```

Creating arrays

```
>>> a0 = numpy.zeros((3,4),numpy.float)
>>>
>>> a2 = numpy.zeros_like(a0)
>>> a3 = numpy.ones_like(a0)
>>> a4 = numpy.empty_like(a0)
>>> # no array_like method, but...
```

Creating arrays

```
>>> d = numpy.array(a)
>>>              # equivalent to  a.copy()
>>> a is d
>>> a[3]=126
>>>
>>> e = numpy.asarray(a)
>>> a is e
>>> a[4]=168
```

Creating arrays

```
>>> d = numpy.array(a)
>>>              # equivalent to  a.copy()
>>> a is d
>>> a[3]=126
>>>
>>> e = numpy.asarray(a)
>>> a is e
>>> a[4]=168
```

Creating arrays

```
>>> l = range(10)
>>> a = numpy.array( range(10) )

>>> a = numpy.arange(10)
>>> # integers 0-9 excluding last value
>>> a = numpy.arange(0,10,0.3)
```

Creating arrays

```
>>> b = numpy.linspace(-2.5,1.5,17)
>>> # 17 equally spaced numbers
>>> # from -2.5 to 1.5
>>> # including the boundary values !
>>> c = numpy.arange(-2.5,1.51,0.25)
>>>
>>> x = numpy.linspace(0, 3.14, 100)
>>> y = numpy.sin(x)
```

Creating arrays

```
>>> # a special feature for Astronomers
>>> m = numpy.logspace(-2,3,9)
```

indexing and slicing

```
>>> l = [[0]*4]*4
>>> # consider as 4x4 matrix
>>> # try to access the lower-right
>>> # 2x2 submatrix
>>> l[2:4][2:4]   # ☹
>>> # assign value to the first row
>>> l[0] = 1      # ☹
>>> l[0] = [1]*4  # hm...
```

indexing and slicing

```
>>> import numpy
>>> a = numpy.zeros( (4,4) )
>>> a[2:4][2:4]   # ☹ as for lists
>>> a[2:4,2:4]    # ☺
>>> a[2:4,2:4] = 1
>>> a[0] = 2
>>> a[:,1:3] += 3
>>> # the array shape is unaltered by
>>> # these operations
```

indexing and slicing

```
>>> # accessing rows and columns
>>> a[0,:]      # the first row
>>> a[0]   = 4 # the first row
>>> # read a[i] as
>>> # a[i,<as many :, as needed>]
>>> # sidemark: specific numpy notation
>>> a[0,...]
>>>
>>> a[:,1] = 5  # the second column
>>> a[:,-1] = 6 # the last column
```

Exercise

- Create a cubic array (3x3x3) of type integer, init the lower 2x2x2 array with some data.
- Calculate the sums along the axes and store the result in the last rows / columns of the array.
- Test different ways to access these last rows / columns.
- with a = numpy.zeros((3,3)) try a[2:3,:] and a[2,:]. Surprised ?

array shapes

```
>>> a = numpy.zeros( (4,4) )
>>> print a.shape
>>> print a.ravel() # flatten the array

>>> b = a.ravel()
>>> b[9] = 9
>>> print a
```

array shapes

```
>>> a = numpy.arange(12)
>>> b = a.reshape((4,3)) #4 rows,3 cols
>>> b[2,1] = -7
>>> print a
>>>
>>> c = b.reshape((12,))
>>> # is equivalent to
>>> d = b.ravel()
```

array shapes

```
>>> a = a.reshape((3,4))
>>> a.shape = (3,4)
>>>
>>> a = a.reshape((2,5)) # shape
>>> a.shape = (9,7)      # checking
>>>
>>> a.shape = (2,-1)
```

array shapes

```
>>> a = numpy.arange(12)
>>> a.resize((6,2))        # in place !!!
>>> print a
>>>
>>> a.resize((4,2))
>>> print a
>>>
>>> a.resize((2,8))
>>> print a
```

array shapes

```
>>> a.resize((6,2))    # in place !!!
>>> a
>>> a.resize((4,2))
>>> a                  # oooops
```

_ in interactive session

```
>>> print _
>>> # special identifier "_" (a single
>>> # underscore in interactive session
>>>
>>> 1+1
>>> print _
>>> a.resize((4,2))
>>> print a
```

Exercise numpy

```
>>> # time to play around with shape,
>>> # reshape and resize, e.g.
>>>
>>> e = numpy.arange(80)
>>> # - convert to a 3D array with
>>> #   various shapes.
>>> # - resize to a 5x5x4 array
>>>
```

numpy ASCII File I/O

```
>>> fname = '/home/mmetz/testn.dat'
>>> # have a look at the file in
>>> # the shell using “less“
>>>
>>> data = numpy.loadtxt(fname)
>>> # inspect properties of the data
>>> # check datatype & dimension ?
```

numpy ASCII File I/O

```
>>> c24 = numpy.loadtxt(fname,
            usecols=(1,3))
>>> # to select columns
>>>
>>> n,w,x,y,z = numpy.loadtxt(fname,
...         unpack = True)
>>>
>>> w,y = numpy.loadtxt(fname,
...         usecols=(1,3), unpack=True)
```

numpy ASCII File I/O

```
>>> fname2 = '/home/mmetz/testn2.dat'
>>> # again have a look at the file
>>>
>>> data2 = numpy.loadtxt(fname2,
...   delimiter='/', comments='!')
>>>
>>> # the default for comments is '#'
```

numpy ASCII File I/O

```
>>> # skip leading rows
>>> # might e.g. be a table header
>>> data2 = numpy.loadtxt(fname2,
...   delimiter='/', skiprows=3,
...   dtype = numpy.int)
>>> print data2.dtype
```

numpy ASCII File I/O

```
>>> data = numpy.loadtxt(fname)
>>> newfname = 'result.dat'
>>> res = numpy.empty((len(data),2),
...   numpy.float)
>>> res[:,0] = data[:,0]
>>> res[:,1] = (data[:,1] + data[:,2])*
...             data[:,3]
>>> numpy.savetxt('result.dat', res)
>>> #have a look at the file result.dat
```

numpy ASCII File I/O

```
>>> numpy.savetxt('result.dat', res,
...     fmt='%.6f')
>>>
>>> numpy.savetxt('result.dat', res,
...     fmt='%.6f', delimiter='\t')
>>>
>>> numpy.savetxt('result.dat', res,
...     fmt='%d :\t %.6f')
```

Exercise numpy File I/O

- The file '/home/mmetz/py2008/hip_sample.dat' (mounted on the cip-servers !!!) contains the Hipparcos catalogue-data for 114820 stars. Extract the Hipparcos number (column 2), and the BT and VT magnitudes (columns 33,35) from the catalogue. Calculate BT-VT colours and write the to a file with the following 3 columns: HipNo, BT-VT, VT
  (can later be used to create a colour-magnitude diagram).

numpy ufunc

- numpy provides universal functions (ufunc) that operate on ndimensinal arrays

```
>>> a = numpy.arange(12).reshape((4,3))
>>> a.sum()
>>> numpy.sum(a)
>>> a.sum(axis=0)
>>> a.sum(axis=1)
>>>
>>> a.prod(axis=0)
```

numpy ufunc

```
>>> a.mean(axis=-1)
>>> a.std(axis=1)
>>>
>>> numpy.average(a, axis=1)
>>> # equivalent to numpy.mean, but
>>> a.average(weights=w)
>>>
>>> numpy.median(a)        # hm !?
```

numpy ufunc

```
>>> a = numpy.arange(4)+1
>>> b = numpy.array([3,1,2,4])
>>>
>>> a.min()                # numpy.min(a)
>>> numpy.min( [a,-b] ) #
>>> numpy.minimum(a,b)  # elementwise
>>>
>>> numpy.max(a)
>>> numpy.maximum(a,b)
```

very useful

```
>>> a = numpy.arange(12).reshape((4,3))
>>> # consider a as Nx3 matrix with
>>> # Cartesian vectors in its rows.
>>> # What calculates the following
>>> # expression ?
>>> numpy.sqrt(numpy.sum(a**2, axis=1))
```

Docs

- Have a look at the website

  http://www.scipy.org/Numpy_Example_List_With_Doc