numbers = set([1, 2, 5])
print 3 in numbers
numbers.add(4)
print numbers
numbers.add(1)
print numbers
print numbers | set(['Rita'])
print numbers - set([2, 3])

Output:
False
set([1, 2, 4, 5])
set([1, 2, 4, 5])
set([1, 2, 4, 5, 'Rita'])
set([1, 4, 5])
None object

None
Object Identity

- A is B
- A is not B
Exercise

A = []  
B = []  
A.append(1)  
B.append(1)

print (A == B)  
print (A is B)

This prints:
(a) True   (b) False   (c) False   (d) True
True   True   False   False   True
Consider the following code:

g2g = {
    'PBANKA_000230': ['GO: 0003899'],
    'PBANKA_000370': ['GO: 0016740'],
    'PBANKA_010060': ['GO: 0030430'],
    'PBANKA_010080': ['GO: 0008270'],
}

(In real life, this would have 2420 entries)
Consider the following code:

```python
g2g = {
    'PBANKA_000230': ['GO: 0003899'],
    'PBANKA_000370': ['GO: 0016740'],
    'PBANKA_010060': ['GO: 0030430'],
    'PBANKA_010080': ['GO: 0008270'],
}
```

(In real life, this would have 2420 entries)
How do you look up GO term for gene PBANKA_000230?

Consider the following code:

```
g2g = {
    'PBANKA_000230': ['GO: 0003899'],
    'PBANKA_000370': ['GO: 0016740'],
    'PBANKA_010060': ['GO: 0030430'],
    'PBANKA_010080': ['GO: 0008270'],
}
```

(In real life, this would have 2420 entries)

How do you look up GO term for gene PBANKA_000230?

(a) `g2g[0]`  
(b) `g2g['PBANKA_000230']`  
(c) `g2g[000230]`
name = [ <expr> for <name> in <sequence> if <condition> ]

maps to

name = []
for <name> in <sequence>:
    if <condition>:
        name.append(<expr>)
squares = [x*x for x in xrange(1,20)]

squares = []
for x in xrange(1,20):
    squares.append(x*x)
def greet():
    print 'Hello World'
print 'Still Here'

greet()
greet()
greet()
print 'Now here'
greet()
```
def greet(name):
    print 'Hello {0}'.format(name)

greet('World')
greet('Luis')
greet('Kim')
```
```python
def max(xs):
    
    M = max(xs)

    Returns the maximum of 'xs'
    
    M = xs[0]
    for x in xs[1:]
        if x > M:
            M = x
    return M
```

Multiple Assignment

\[ A, \ B = 1, 2 \]

Assign multiple elements at once.
def greet(name, greeting='Hello '):
    '''
    greet(name, greeting='Hello ')
    Greets person by name
    Parameters
    ----------
    name: str
        Name
    greeting: str, optional
        Greeting to use
    '''
    print greeting, name

ret = greet( 'World' )
for value in sequence:
...

Sequences

- Lists
- Tuples
- Sets
- Dictionaries
- ...
Goals for next 15 minutes

- A quiz
- Do a few exercises.
- Play around.
- You can work alone, in pairs, in triples,...
- Looking up answers on the internet is technique, not cheating!
How do you access the first element of a list? Assume list is a list:

1. `list[1]`
2. `list[0]`
3. `list[-1]`
4. `list(0)`
5. `list(-1)`
6. `list(1)`
How do you access the last element of a list?
Assume list is a list:

1. `list[1]`
2. `list(-0)`
3. `list[-1]`
4. `list(-1)`
5. `list(1)`
6. `list[-0]`
Exercises
What is the difference between the following two code examples:

A)

A = [1, 2, 3]
B = [1, 2, 3]

B)

A = [1, 2, 3]
B = A

Write a small piece of code (should be 2 or 3 lines) that behaves differently if you insert it after each of the two segments above.
What is the difference between the following two code examples:

A)  
```python
A = [1, 2, 3]
B = [1, 2, 3]
```

B)  
```python
A = [1, 2, 3]
B = A
```

Write a small piece of code (should be 2 or 3 lines) that behaves differently if you insert it after each of the two segments above.

```python
B[0] = 0
print A
```
1 Learn about the built-in function sum
2 Write an implementation of this function
1. Learn about the built-in function `sum`.

2. Write an implementation of this function.

```python
def sum(xs, start=0):
    
    s = sum(xs, start=0)

    Returns the sum of all values in `xs` + `start` (which defaults to 0)

    for x in xs:
        start += x
    return start
```
numbers = set([1, 2])
for i in xrange(5):
    numbers.add(i)
print len(numbers)

This prints:

- 7
- 6
- 5
- 4
Learning more

- Learn Python the Hard Way by Zed Shaw (online for free or pay money for hard copy)
- http://python.org
- Numeric (1995)
- Numarray (for large arrays)
- scipy.core (briefly, around 2005)
- numpy (2005)
Currently

- numpy 1.6
- *de facto* standard
- very stable
Basic Type

numpy.array or numpy.ndarray.

Multi-dimensional array of numbers.
numpy example

```python
import numpy as np
A = np.array([
    [0, 1, 2],
    [2, 3, 4],
    [4, 5, 6],
    [6, 7, 8]])
print A[0, 0]
print A[0, 1]
print A[1, 0]
```
Some Array Properties

```python
import numpy as np
A = np.array([
    [0, 1, 2],
    [2, 3, 4],
    [4, 5, 6],
    [6, 7, 8]]
)
print(A.shape)
print(A.size)
```
Some Array Functions

... 

\texttt{\textbf{print}} A.\texttt{max()} \\
\texttt{\textbf{print}} A.\texttt{min()}

\begin{itemize}
\item max(): maximum
\item min(): minimum
\item ptp(): spread (max - min)
\item sum(): sum
\item std(): standard deviation
\item ...
\end{itemize}
Other Functions

- `np.exp`
- `np.sin`
- ...

All of these work **element-wise**!
import numpy as np
A = np.array([0, 1, 2, 3])
B = np.array([1, 1, 2, 2])

print A + B
print A * B
print A / B
Broadcasting

Mixing arrays of different dimensions

```python
import numpy as np
A = np.array([
    [0, 0, 1],
    [1, 1, 2],
    [1, 2, 2],
    [3, 2, 2]
])
B = np.array([2, 1, 2])

print A + B
print A * B
```
Broadcasting

Special case: scalar.

```python
import numpy as np
A = np.arange(100)
print A + 2
A += 2
```

Luis Pedro Coelho (luis@luispedro.org) (EMBL) ⋆ More Python Types & Functions ⋆ June 10, 2014 (32 / 47)
**Data Types**

numpy.ndarray is a homogeneous array of numbers.

**Types**

- Boolean
- integers
- floating point numbers
- ...

Luis Pedro Coelho (luis@luispedro.org) (EMBL) ★ More Python Types & Functions ★ June 10, 2014 (33 / 47)
import numpy as np
A = np.array([0,1,1], float)
A = np.array([0,1,1], bool)
Reduction

```python
A = np.array([
    [0, 0, 1],
    [1, 2, 3],
    [2, 4, 2],
    [1, 0, 1]])

print A.max(0)
print A.max(1)
print A.max()
```

prints

```
[2, 4, 3]
[1, 3, 4, 1]
4
```

The same is true for many other functions.
import numpy as np
A = np.array([
    [0,1,2],
    [2,3,4],
    [4,5,6],
    [6,7,8]
])
print A[0]
print A[0].shape
print A[1]
print A[:,2]
Two minute break

- Talk to your neighbours
- Play around in Python
- Ask questions
Slices Share Memory!

```python
import numpy as np
A = np.array([
    [0, 1, 2],
    [2, 3, 4],
    [4, 5, 6],
    [6, 7, 8]])
B = A[0]
B[0] = -1
print A[0, 0]
```
Pass is By Reference

\begin{verbatim}
def double(A):
    A *= 2

A = np.arange(20)
double(A)
\end{verbatim}
def double(A):
    A *= 2

A = np.arange(20)
double(A)

A = np.arange(20)
B = A.copy()
Logical Arrays

\[ A = \text{np.array}([-1, 0, 1, 2, -2, 3, 4, -2]) \]
\[ \text{print} (A > 0) \]
A = np.array([-1, 0, 1, 2, -2, 3, 4, -2])
print((A > 0) & (A < 3)).mean()

What does this do?
Logical Indexing

\[ A[A < 0] = 0 \]

or

\[ A *= (A > 0) \]
Logical Indexing

```python
print 'Mean of positives', A[A > 0].mean()
```
Some Helper Functions

Constructing Arrays

\[
\begin{align*}
A &= \text{np.zeros}((10,10), \text{int}) \\
B &= \text{np.ones}(10) \\
C &= \text{np.arange}(100).\text{reshape}((10,10)) \\
&\ldots
\end{align*}
\]

Multiple Dimensions

\[
\begin{align*}
\text{img} &= \text{np.zeros}((1024,1024,3))
\end{align*}
\]
http://docs.scipy.org/doc/
Matplotlib is a **plotting library** for Python.

```python
import pylab
import numpy as np
X = np.linspace(-4, +4, 1000)
pylab.plot(X, np.exp(-X**2))
pylab.xlabel(r'$x$')
pylab.ylabel(r'$\exp(-x^2)$')
pylab.savefig('gaussian.pdf')
```

http://matplotlib.sf.net/
$$\exp(-x^2)$$