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

```python
g2g = {
    'PBANAKA_000230': ['GO: 0003899'],
    'PBANAKA_000370': ['GO: 0016740'],
    'PBANAKA_010060': ['GO: 0030430'],
    'PBANAKA_010080': ['GO: 0008270'],
}
```

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

```python
import g2g

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:

```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?

(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>)
List Comprehensions Example

\[
\text{squares} = [x^2 \text{ for } x \text{ in } \text{xrange}(1, 20)]
\]

\[
\text{squares} = []
\text{for } x \text{ in } \text{xrange}(1, 20):
    \text{squares}.\text{append}(x^2)
\]
def greet():
    print 'Hello World'
print 'Still Here'
greet()
greet()
greet()
print 'Now here'
greet()
```python
def greet(name):
    print 'Hello {0}'.format(name)

greet('World')
greet('Luis')
greet('Kim')
```
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)
Lists II

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)

```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.
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
Learn about the built-in function `sum`

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
```

Luis Pedro Coelho (luis@luispedro.org) (EMBL) ⋆ More Python ⋆ November 14, 2014 (20 / 39)
numbers = set([1, 2])
for i in xrange(5):
    numbers.add(i)
print len(numbers)

This prints:

- 7
- 6
- 5
- 4
And now, for something completely different...
Procedural programming: organising programs around functions.
Object-oriented programming: organising programs around objects.
Object Oriented Programming

**OOP**

**Aggregation** organise functions & data into classes.

**Encapsulation** hide information inside methods.

**Polymorphism** re-use code for multiple types.

**Inheritance** re-use code from one class to build another.
User-Defined Types

Built-in Types

1. lists
2. dictionaries
3. strings
4. ...

More Python
November 14, 2014 (25 / 39)
What’s a Type

1. A domain of values
2. A set of methods (functions)
Examples of Types

List

1. Domain: lists
2. Functions: L.append(e), L.insert(idx, e), ...
3. Operators: L[0], 'Rita' in L

Luis Pedro Coelho (luis@luispedro.org) (EMBL)  ⋆  More Python  ⋆  November 14, 2014  (27 / 39)
Examples of Types

List

1. Domain: lists
2. Functions: \texttt{L.append(e)}, \texttt{L.insert(idx,e)}, ...
3. Operators: \texttt{L[0]}, 'Rita' in \texttt{L}

Integer

1. Domain: \ldots, \ldots, -2, 1, 0, 1, 2, \ldots
2. Operators: \texttt{A + B},...
User-defined Types

Object-oriented programming languages allow us to define new types.
Motivating Example

Simple Simulation

1. Boat goes around the ocean
2. You can move it around
We define a Boat class, with two values, latitude & longitude, and five methods:

1. move_north, move_south, move_east, move_west
2. distance
Using our Boat

```python
b = Boat()
b2 = Boat()
b.move_north(1.)
b2.move_south(2.)
print b.distance(b2)
```
A class aggregates data and functions that belong together.
## Boat Interface

### Interface

**Functions:**

1. **Constructor:** Takes the initial adaptation value and sigma.
2. **move_*:** Moves the boat.
3. **distance(b):** Computes the distance between two boats.

**Data elements:**

1. **latitude:** Current latitude.
2. **longitude:** Current longitude.
Calling Methods

Defining a method

```python
class Boat(object):
    def __init__(self, lat=0, long=0):
        self.latitude = lat
        self.longitude = long

    def move_north(self, dlat):
        self.latitude += dlat
```

Calling a Method

```python
obj = Boat()
obj.method(arg1, arg2)
```
Object Oriented Programming

**OOP**

**Aggregation** organise functions & data into classes.

**Encapsulation** hide information inside methods.

**Polymorphism** re-use code for multiple types.

**Inheritance** re-use code from one class to build another.
Duck Typing
Object Oriented Programming

**OOP**

Aggregation organise functions & data into classes.

Encapsulation hide information inside methods.

Polymorphism re-use code for multiple types.

Inheritance re-use code from one class to build another.
Object Oriented Programming

OOP

Aggregation organise functions & data into classes.
Encapsulation hide information inside methods.
Polymorphism re-use code for multiple types.
Inheritance re-use code from one class to build another.