

Introduction to Data Science Programming

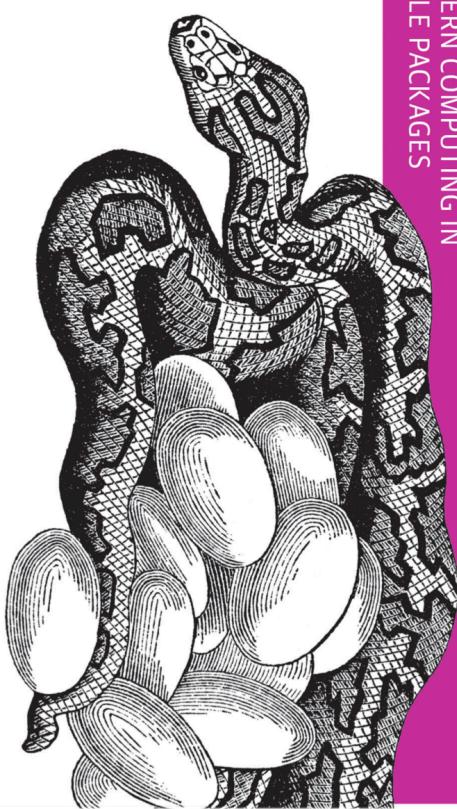
03: Sequences, Types & Dictionaries

- ❖ Checking In
- Sequences & Other Types
- Lists, Ranges, Tuples, Sets (activity 1)
- Dictionaries (activity 2)
- Mutability & Gotchas (activity 3)

O'REILLY®

Introducing Python

MODERN COMPUTING IN
SIMPLE PACKAGES



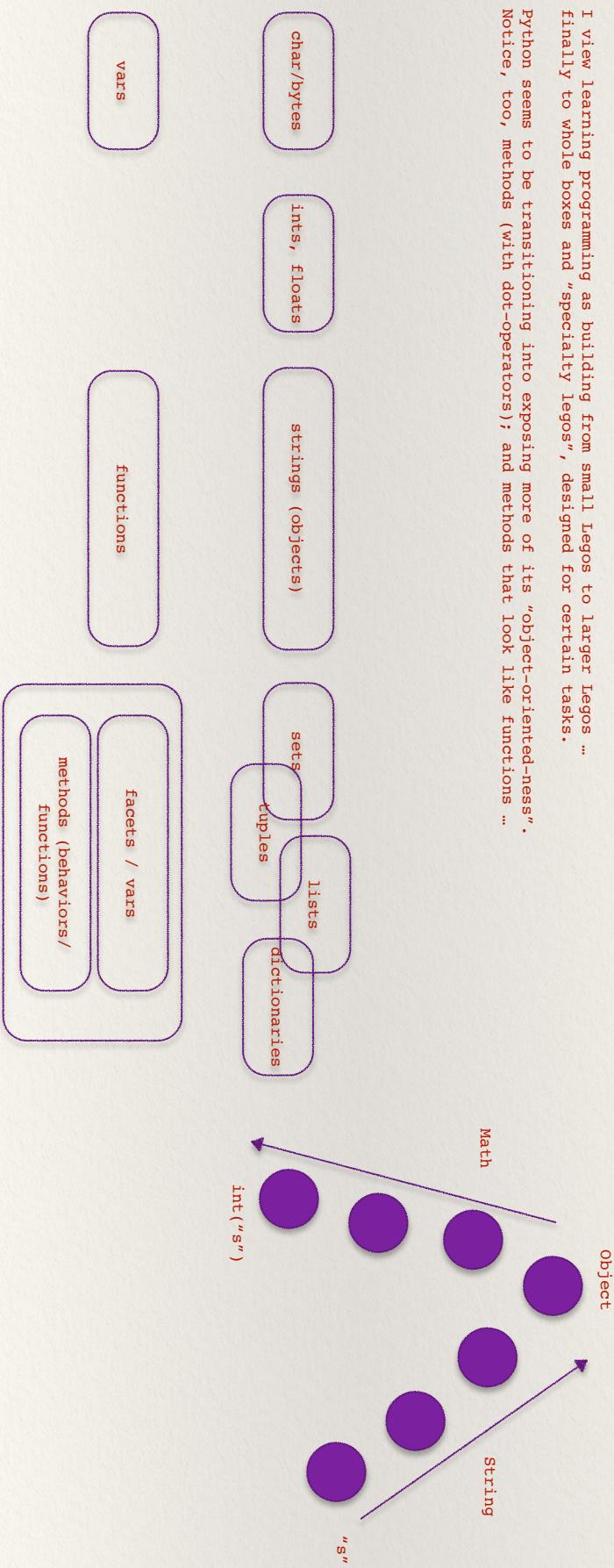
Bill Lubanovic

Checking In

- ❖ How is it going?
- ❖ What was the hardest part of the homework? The easiest?

I view learning programming as building from small Legos to larger Legos ... finally to whole boxes and "specialty legos", designed for certain tasks.

Python seems to be transitioning into exposing more of its "object-orientatedness". Notice, too, methods (with dot-operators); and methods that look like functions ...



Sequences & Other Types

- ❖ What are they?

In Python, a sequence is a generic term for objects with an ordered group.

Examples include



*Notice this string is a sequence,
read starting from the first character.*

lists,
tuples, and
strings.

- ❖ What's not a sequence?

Any data types without an inherent order, such as **dictionaries**, **sets**, **ints**, and **floats**.

Lists

- ❖ Sequences share a lot of common methods (but not all).
Include or slice with `[]`. The offset starts with 0.
E.g., `myList = ['Lars', 'Juan', 'Pierre', 'Marie', 'Tan']`
- ❖ How many elements in the list?
`"Lars' in myList"`
- ❖ I'm looking for "Lars" - is he in the list?
- ❖ How to add (concatenate) 2 lists?
`a = list()
a.append("tom")
a[0] = "tom"`

Lists

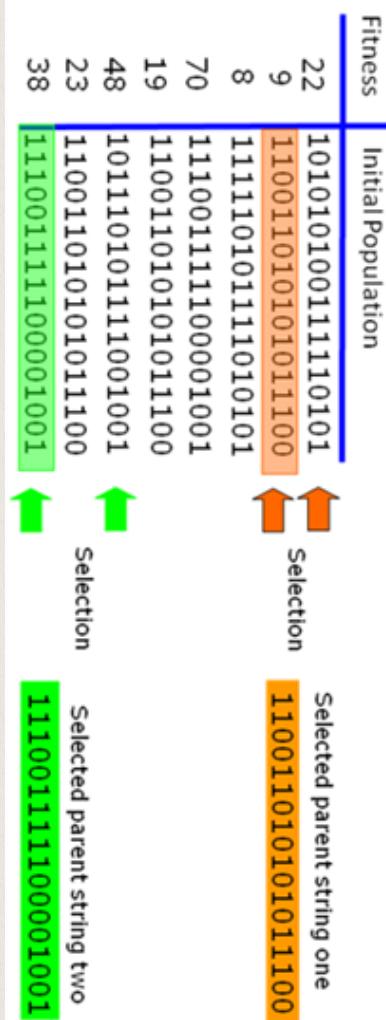
- ❖ Sequences are helpful! Their methods tell us about the size, min/max values, counts (occurrences) of an object, and more!
- ❖ **myList = [1,2,3,4]**
 - ❖ min(myList)
 - ❖ max(myList)
- ❖ **myList = ['a', 'b', 'c', 'd']**
 - ❖ myList.index('x') # locate the first instance of "x"
 - ❖ myList.count('x') # how many times of "x"
- ❖ What's up with the dot? And the parentheses?

Parentheses hold the arguments we're passing to the function, e.g., len(myList).

Not all functions require arguments.

The dot notation indicates that a function is defined in the specific object before the dot.

Lists



While there are better techniques we'll encounter in NumPy and Pandas, we can imagine using the `<list>.index` and `<list>.count` to test for existence of an interesting value and then count them for fitness.

Data Mask
Paragraph/
Topic Signature
old-style IPv4

000010101000101111010100101010
000010101000101111010100101010

000010101000101111010100101010

Composites

- ❖ A **list** is a composite type - what does that mean?
- ❖ Other examples?
- ❖ A composite type comprises other types. Lists, tuples, dictionaries are composite because they can contain other objects. int, float, string are not composites.
- ❖ E.g.,

```
demo = [ 'cat', 'dog', 33, [ 'j', 'k', 'l' ] ]
```

The list has four elements:
 - 0 cat
 - 1 dog
 - 2 33
 - 3 ['j', 'k', 'l']A red bracket groups the last element, which is another list.
- ❖ `print(demo[3])` outputs 'j', 'k', 'l'

List Methods

- ❖ `myList.insert(index, value)`
- ❖ `myList.pop(x)` # pops last value by default but can take instead index argument "x")
- ❖ `myList.remove()` # use remove to eliminate the first instance of a value
- ❖ `myList.sort()` # mutate the list - sorting by default in ascending order
- ❖ `sorted(myList)` # this returns a new list
- ❖ `myList.reverse()` # reverses the list.

List Methods (continued)

- ❖ `myList.append("x")` # adds "x" to the end of the list
- ❖ `myList.extend(otherList)` # adds items from otherList to the end of myList
- ❖ `myList[a] = "z"` #swaps out the item at index [a] with whatever z stands for
- ❖ `myList.clear()` #clears out the list elements
- ❖ `del(myList[x])` # deletes item from index x

Ranges, Tuples, Sets

❖ Ranges

- ❖ a sequence
- ❖ need to be listed to yield the elements
- ❖ range(start, stop, step)

❖ Tuples

- ❖ a sequence
- ❖ like a list but immutable
- ❖ instantiate: `tup_x=(1,2,3)` or `tup(1,2,3)`
- ❖ Can use a tuple to create multiple objects

❖ Sets

- ❖ Unordered and mutable
- ❖ *Unique, keys only

```
>>> a=range(0,9)
>>> a
range(0, 9)
>>> list(a)
[0, 1, 2, 3, 4, 5, 6, 7, 8]
>>> type(a)
<class 'range'>
>>> type (list(a))
<class 'list'>
```

```
>>> low, high = 10,20
>>> print(low, high)
10 20
>>> █
```

More About Tuples

- ❖ Tuples are like lists - but they are immutable. What's happening here?

```
>>> a=[[1,2,3],2,3]
>>> type(a)
<class 'tuple'>
>>> a[0].append(5)
>>> a
[[1, 2, 3, 5], 2, 3]
>>> type(a)
<class 'tuple'>
>>> a[1]=10
>>> a[1]=10
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Activity 1

- ❖ range(start, stop(exclusive), step)
- ❖ Practice creating these outputs:
 - ❖ [1, 2, 3, 4, 5, 6, 7, 8, 9]
 - ❖ [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
 - ❖ [2, 4, 6, 8, 10, 12]
 - ❖ [2, 4, 6, 8, 10, 12, 13, 14, 15, 17, 19, 21]
 - ❖ [-1, 0, 1, 2, 3]
 - ❖ [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

Dictionaries

- ❖ Instantiation (many ways)

```
dict_x = {'fred':1, 'frank':3, 'ben':1}
```

As a dict literal

```
dict_y = dict()
```

```
dict_x = dict([('fred',1),('frank',3),('ben',1)])
```

As a list of tuples

Index by key to get value

```
dict_x['fred']      # indexing by key name
```

Dictionary Methods

```
del(dict_x[ 'key' ]) # delete by key reference
dict_x.pop( 'key', 'default value') # pop the value for key from dictionary. If the
# key doesn't exist, the function will return the
# default
dict_x.get( 'key', 'default value')
dict_x.clear()
dict_x.update(dict2)
dict_x.keys()
dict_x.values()
dict_x.items()

# appends a second diction to the first
# get the key:value pairs
```

Sample .json structures

```
{ people
  [name, tom]
  [name,
fifi]
}

{
  "menu": [
    {
      "id": "file",
      "value": "File",
      "popup": [
        {
          "menuitem": [
            {
              "value": "New",
              "onclick": "CreateNewDoc()"
            },
            {
              "value": "Open",
              "onclick": "OpenDoc()"
            },
            {
              "value": "Close",
              "onclick": "CloseDoc()"
            }
          ]
        }
      ]
    }
  ]
}
```

Just a note in passing ...

JavaScript Object Notation (.json) files are used a lot to share data.

Notice that GitHub files (in "raw") mode show the underlying .json that's converted to a useful tool in Jupyter.

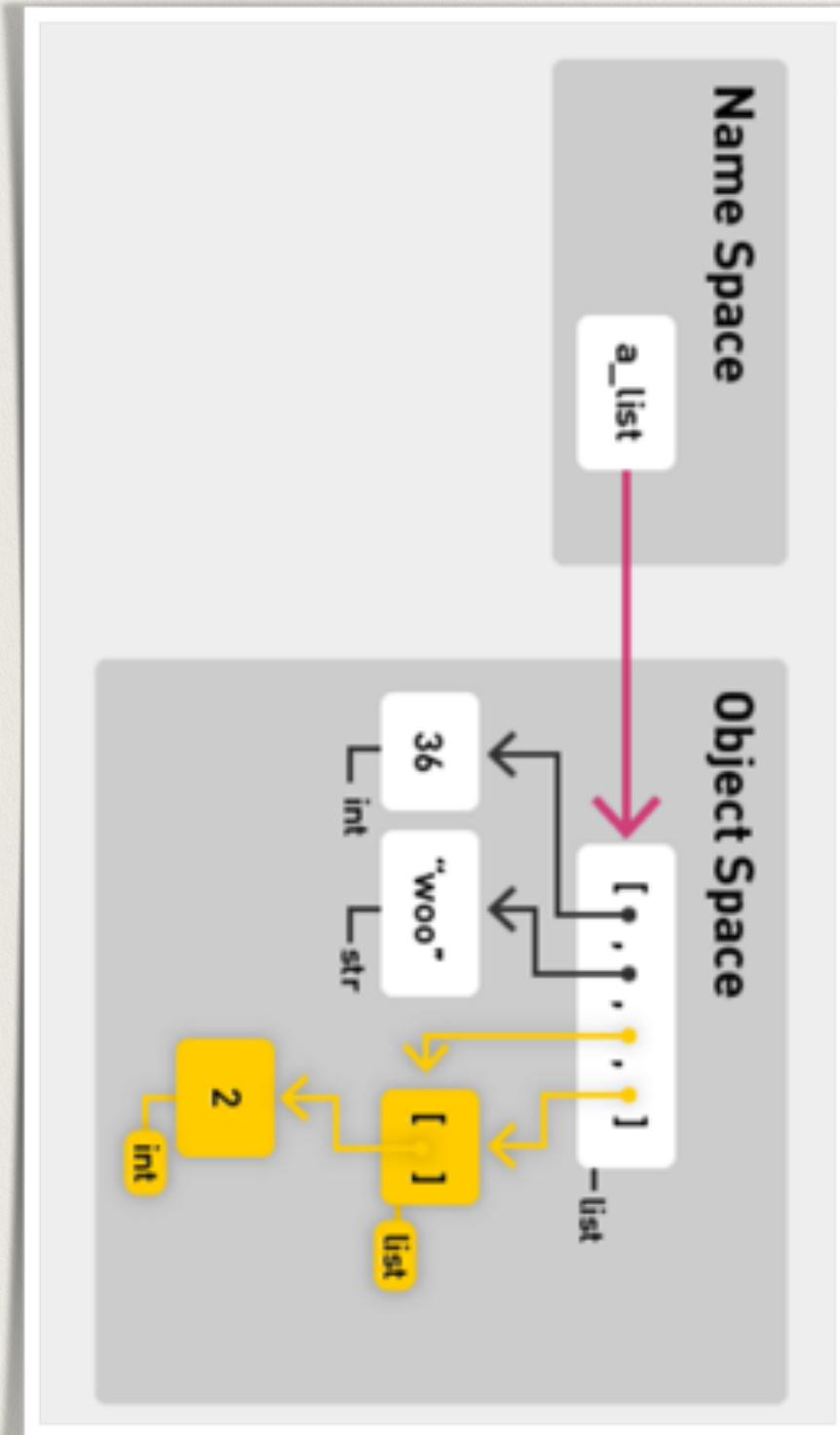
Activity 2 - Dictionaries

- ❖ Mutable, what does that imply?
- ❖ Not a sequence, what does that mean?
 - ❖ Maps keys to values # aka: map, key:value store
 - ❖ `a = {'fred':1, 'frank':3, 'ben':1}`
 - ❖ `book = {'changjing':'555-1212', 'jim':'333-234'}`
 - ❖ `a = {'names': {'fred':404, 'frank':3, 'ben':1}} # can be nested (JSON)`
- Python uses a hash function to quickly locate items stored in a dictionary. The key, when passed through the hash function, points to a unique place in the computer's memory. This makes finding the value extremely fast. Keys cannot be mutable, since if they were, the hash function would not return the same result.
- ❖ Values can be any type
- ❖ Keys need to be hashable # should be immutable

Mutability

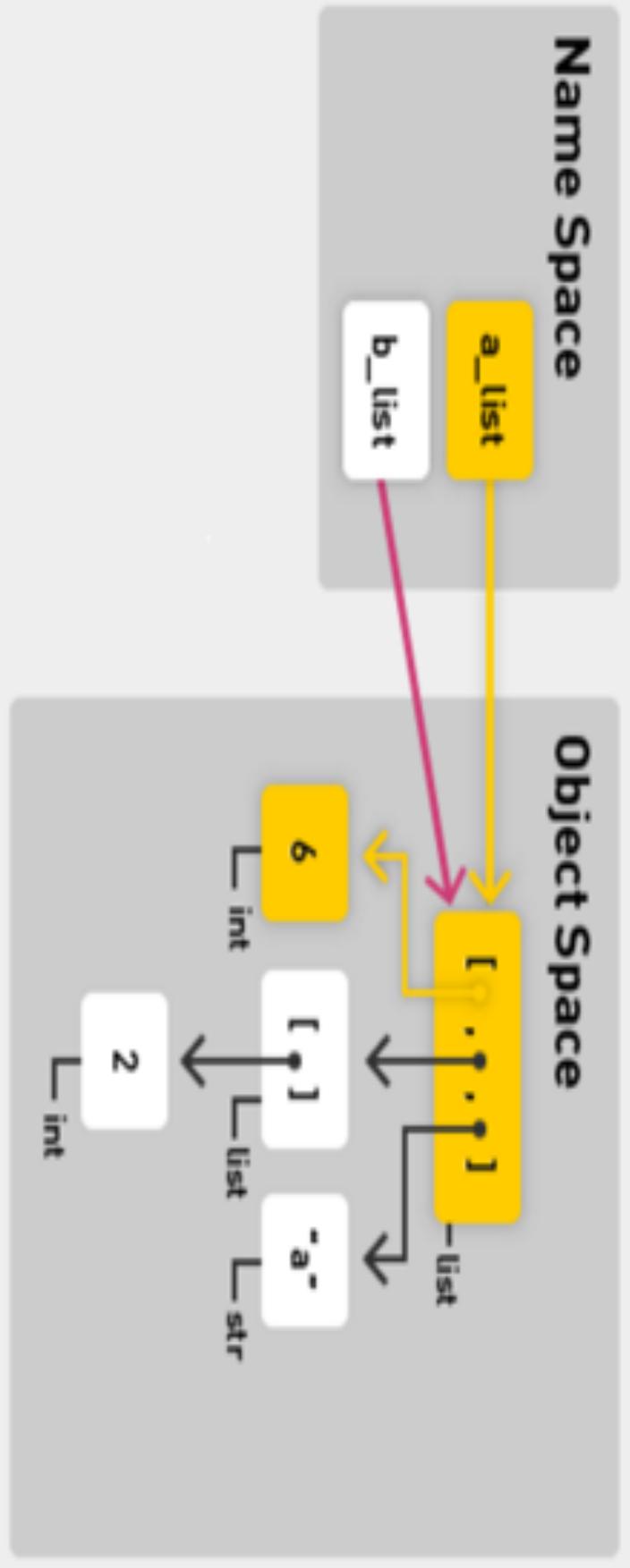
- ❖ Mutability: a list is mutable, meaning we can change the length and content. E.g.,
- ❖ `aList = []`
- ❖ `aList.append('cat')`
- ❖ `bList = ['a', 'b', 'c']`
- ❖ `aList.extend(bList)`
- ❖ `print(aList)`
- ❖ `['cat', 'a', 'b', 'c']`
- ❖ Which data types are mutable? Immutable?
- ❖ Mutability means the object can be changed;
- ❖ dictionaries, lists, and sets are mutable;
- ❖ tuples and strings are immutable.
- ❖ Primitive data types such as ints and floats are also immutable.

Mutability Gotchas

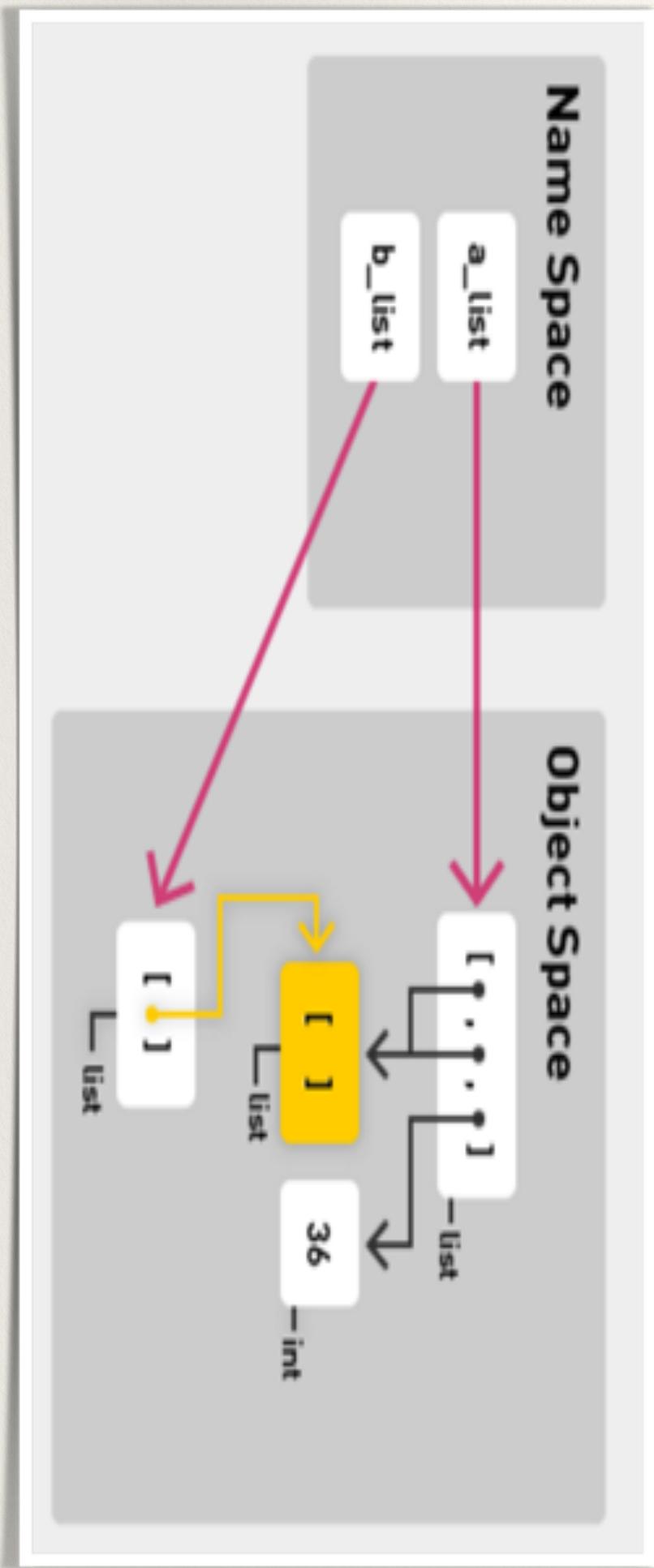


Note, items 3 and 4 are the same object: `[36, "woo", [2], [2]]`

Mutability Gotchas (con't)



Mutability Gotchas (con't)



Copy & Deepcopy

- ❖ Consider this snippet:

```
Ls_x = [ 1, 2, 3, ['Frank', 'Fred']]  
Ls_x_cp = Ls_x.copy()
```

```
from copy import deepcopy
```

```
Ls_x_deep = deepcopy(Ls_x)  
Ls_x[3][1] = 'Mufasa'
```

Copy & Deepcopy (con't)

- ❖ What is **copy**?
- ❖ Copy creates an independent copy of all list elements at the first level of the list.
- ❖ How does copy differ from **deepcopy**?
- ❖ Deepcopy creates an independent copy of all list elements at all levels.
- ❖ What is the final value of Ls_X_cp and Ls_X_deep?
- ❖ Lx_x_cp is [1,2,3,['Frank','Mufasa']]
- ❖ Lx_x_deep is [1,2,3,['Frank','Fred']]

List & Dictionary Activity

- ❖ We are now going to solve a very popular problem: *How do you count the words in a document?*
- ❖ While the solution here is simple, you will see in later courses that this is an excellent first problem when learning how to massively parallelize your code across a cluster of computers.
- ❖ The activity will guide you to the solution in a series of steps.
- ❖ As you will see next week, the “`while`” loop in this activity could be better represented by a “`for`” loop. For now, please work with the “`while`” loop.

Activity 3 - Mutability

- ❖ Mutability Activity:
- ❖ Read about and update a scoreboard reporting ranking and team color of contestants.
- Contestants = [{"name": "fred", "teamColor": "Red"},
 {"name": "Layla", "teamColor": "Yellow"},
 {"name": "Tammy", "teamColor": "Green"},
 {"name": "Buba", "teamColor": "Blue"}]
- ❖ Your job? Programmatically change the scoreboard as indicated. Hint! Use **copy** and/or **deepcopy** if required.

That's It

- ❖ Remember that scores & comments are listed in ISVC site.
- ❖ If you want to redo assignment 1, go ahead.
- ❖ Home works are very important - communicate with each other, with the instructors, etc. If you're spending too much time, step away and rest ... then tackle with fresh eyes.
- ❖ At this point in our studies we might slog thru some code ... There is almost always a more efficient way of doing things and we're going to encounter many of them in the coming weeks.

