Class 2 - Starting with Python; sequence types and dictionaries

[w18] Python For Data Science - Summer 2018

Week 2 | Agenda

Week 1 Assignment and Polls

Expressions

Objects

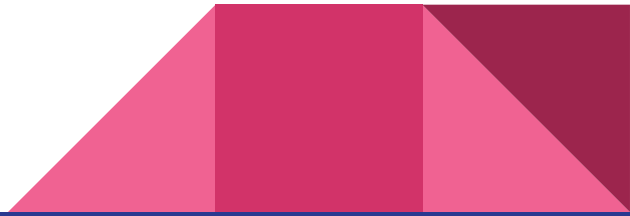
Variables

Strings - Activity 1

Control flow - Activity 2

Breakout - Activity 3

Homework 2



Course Content | First 8 Weeks - Programming

Unit 1 | Introduction, the Command Line, Source Control

Unit 2 | Starting Out with Python

Unit 3 | Sequence Types and Dictionaries

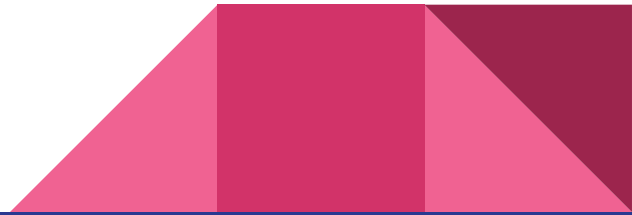
Unit 4 | More About Control and Algorithms

Unit 5 | Functions

Unit 6 | Modules and Packages

Unit 7 | Classes

Unit 8 | Object-Oriented Programming



logistics

Asynchronous, class meetings, and breakout sessions

Homeworks and assignments

https://github.com/MIDS-INFO-W18/assignments_upstream_summer18_SS

The Google group list

https://groups.google.com/forum/#!forum/w18-summer-session-2018

Using GitHub to get and submit your assignments

Important Locations | GitHub

Github-playground - Fun code and **student discovered resources**

Assignments_upstream_summer18 - The homework release site, as well as any supplemental class activities

FirstnameLastnameREPO - Your personal HW repository

- We capture this from your week 1 post - **if you change this let us know.**

Course-Syllabus - iPython Notebooks from async and related data

Drills - Additional exercises for fun

Week 1 Assignment | Check in

- Polls
  - How long did HW1 take?
  - How difficult was HW1 and the general setup?
  - How comfortable are you with the workflow?
  - Is python3/ github/ bash/ jupyter functional?
  - Any questions?

- Show SampleREPO structure

Quick Break | Breakouts

Say “Hi!” and tell your partner(s) why you enrolled in MIDS.

Also where you live and your favorite animal (clearly penguins).

Please make sure each member of your breakout room has the ability to “Share Screens”. If not, let your instructor know.

Please make sure each member of your room has “cloned” into all of the repositories listed on the previous screen. Then, please “git pull” the Class 2 Activity file from assignments-upstream.

Send the instructor (me) a message saying you have completed.

Running Python | Four Methods

1. Use the command line.
2. Write a .py script in a text editor. Run it from the command line.
    a. Mac users need to start their file with the location of python. Use the “which python” command in Terminal to find your path:
        i. #!/usr/bin/env
    b. Mac users also **may** need to make the file executable via the command line
        i. `chmod +x file.py`
3. Use a Jupyter Notebook.
4. Use an Integrated Development Environment (IDE) such as Spyder or PyCharm (not in this class).

Week 2 | Agenda

Week 1 Assignment and Polls

Expressions

Objects

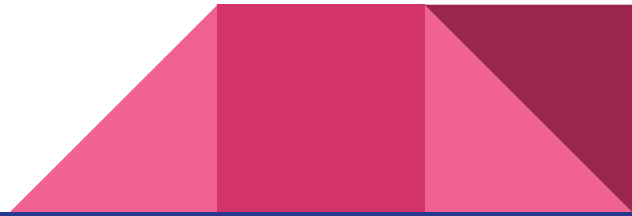
Variables

Strings - Activity 1

Control flow - Activity 2

Breakout - Activity 3

Homework 2



Expressions | Basic math operations

+,-, /,,***

== testing equality

// integer division

% modulus (remainder)

divmod(numerator, denominator)

**** can you think of an application for the modulus**

Expressions | repeated operation shorthand

Cumulative calculation

+=

/=

***=**

-=

...

Week 2 | Agenda

Week 1 Assignment and Polls

Expressions

Objects

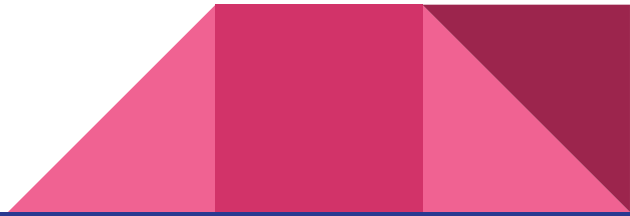
Variables

Strings - Activity 1

Control flow - Activity 2

Breakout - Activity 3

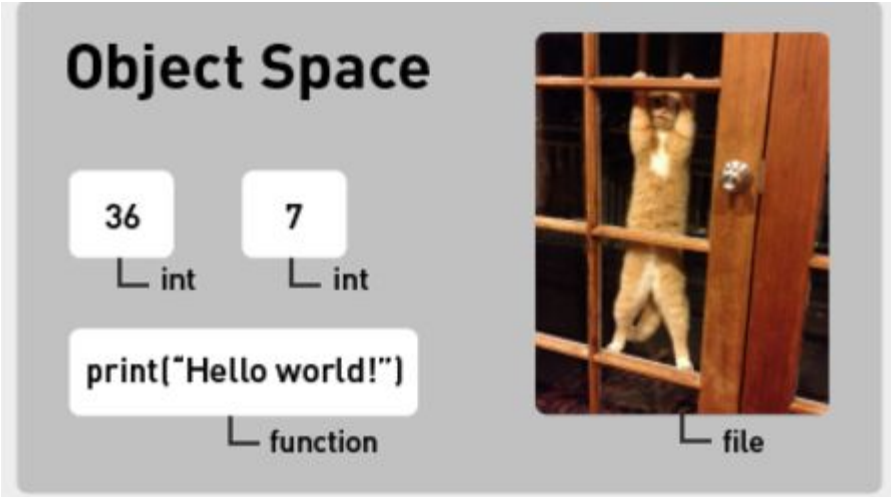
Homework 2



Objects | Basic types (classes)

types
restrict what can be done to an object

Everything in python is an object
every object has a type (class)



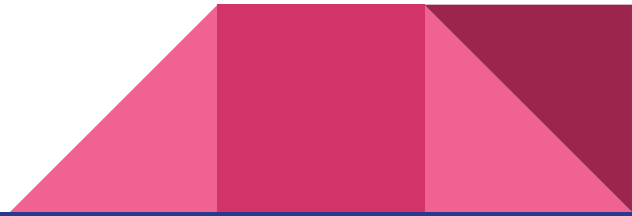
Objects | Basic types (classes)

boolean (bool) - True or False

integers (int) - 1,2,3..

floats (float) - 1.34523

strings (str) - “this is a string”
sequence of characters



Week 2 | Agenda

Week 1 Assignment and Polls

Expressions

Objects

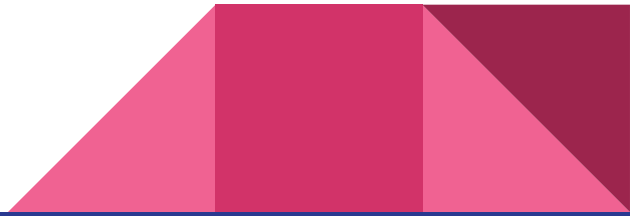
Variables

Strings - Activity 1

Control flow - Activity 2

Breakout - Activity 3

Homework 2

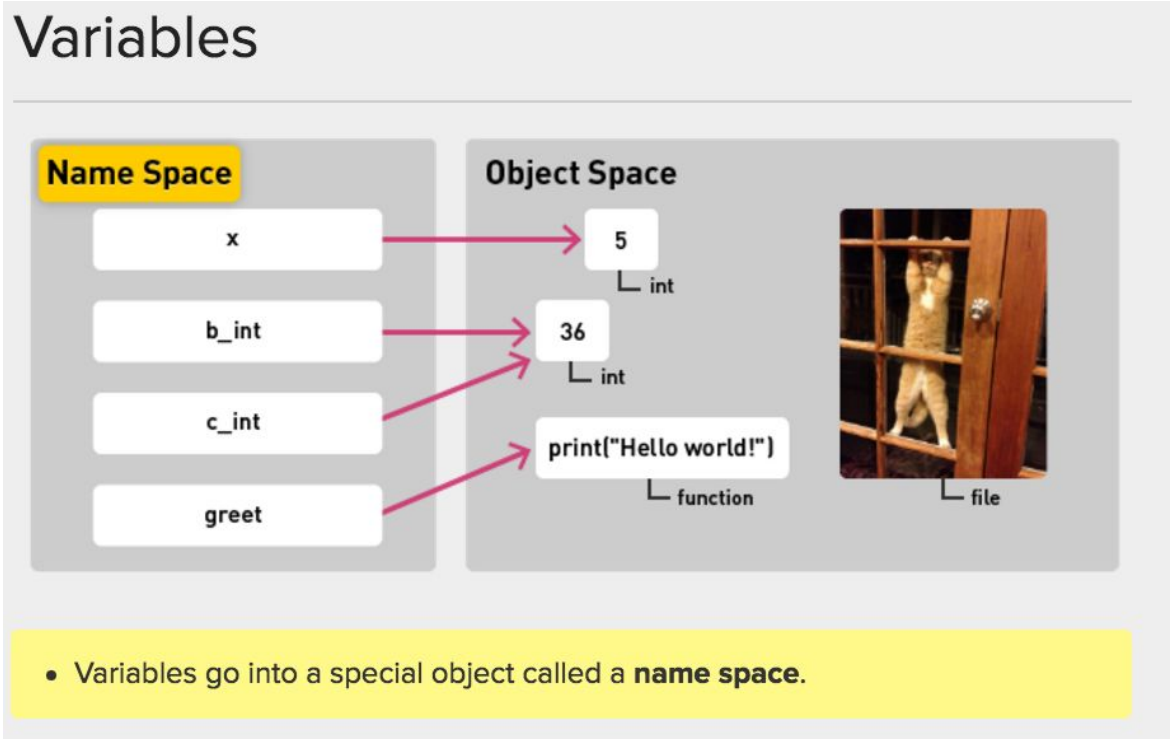


Variables | naming objects

Variables and objects are distinct

Distinct spaces

Objects have types variables do not



Week 2 | Agenda

Week 1 Assignment and Polls

Expressions

Objects

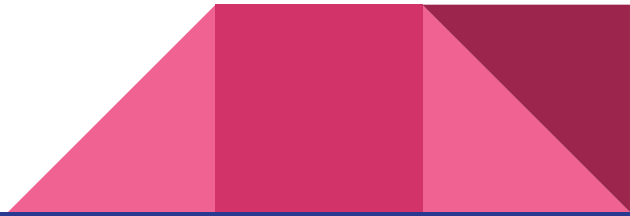
Variables

Strings - Activity 1

Control flow - Activity 2

Breakout - Activity 3

Homework 2



String objects | overview

- String is a sequence object
- We can use **extraction by index**
- **Index starts at 0**
  - think about it the index as an offset
- **Concatenation**
  ‘Cat ’ + ‘Dog’ -> ‘Cat Dog’
- **Multiplication**
  ‘Cat’ * 3 -> ‘CatCatCat’

String slicing | using indices

```
[0]      # the start
[-1]     # one from the end
[0:3]    # from the start to the THIRD letter (fourth is excluded)
[1:-1]   # 1 to the second to last
[1:5:2]  # 1 to 5 by 2s
[:-1]    # beginning to second to last
[:]      # whole thing
[::-1]   # start to end reversed
```

String | special characters etc.

‘ or " # to specify strings

“ “ “ # three quotes for block quotes

\ # escape

\n, \t, \” # escape use cases

print (‘somestring’, end= ‘ ‘) # override the newline at end with space

String | functions

<string>.upper()

<string>.lower()

var=input(“your message”) # note it saves strings

str() # type cast a string

String exercises | Try this

1) open your console and create a string variable that prints exactly

The "trouble with

Tribbles" that they

\\\EAT/// too many MREs.

2) using one line of Python code, slice your variable from part 1 to print ‘selbbirT ’ 300 x

selbbirT selbbirT selbbirT selbbirT ….

Week 2 | Agenda

Week 1 Assignment and Polls

Expressions

Objects

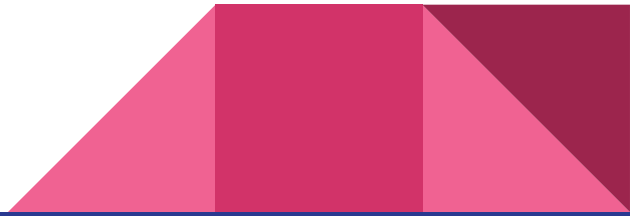
Variables

Strings - Activity 1

Control flow - Activity 2

Breakout - Activity 3

Homework 2



Flow control | conditionals and loops

Deviating from linear programming (scripting)

Flow control element ends with ":"

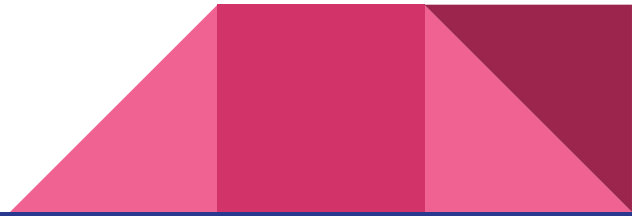
Code suites, noted by indentation

Indented by **4 spaces**

```
countdown = 5
while countdown > 0:
    print(countdown)
    countdown -= 1
print("Blast off!")
```

Conditionals | if, elif, else

```
if x > 2:
    print ( ‘x is greater than 2”)
elif x < 0 :
    print(’x is negative')
else:
    print (‘x is less than 2 but still positive')
```



Control flow exercise | fix this code

```
ans = input( 'do you have 8 legs?’)
if ans == "yes"
print (“you are a spider")
else
ans = input( 'do you have 4 legs?’)
if
print ('you are a quad')
else
if ans == "yes"
else:
print ('you are a bicycle')
```

While loops |

- repeat until condition is satisfied

```
countdown = 5
while countdown > 0:
    print(countdown)
    countdown -= 1
print("Blast off!")
```

```
5
4
3
2
1
Blast off!
```

Nested loops | to repeat an action

```
row = int(input("Enter an integer: "))

# while row >= 0:

j = 0
while j <= row:
    print(j, end=" ")
    j += 1
```

```
Enter an integer: 5
0 1 2 3 4 5
```

```
row = int(input("Enter an integer: "))

while row >= 0:

    # inner loop
    j = 0
    while j <= row:
        print(j, end=" ")
        j += 1

    print("")
    row -= 1
```

```
Enter an integer: 5
0 1 2 3 4 5
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0
```

Note the use of end = “ ”

Week 2 | Agenda

Week 1 Assignment and Polls

Expressions

Objects

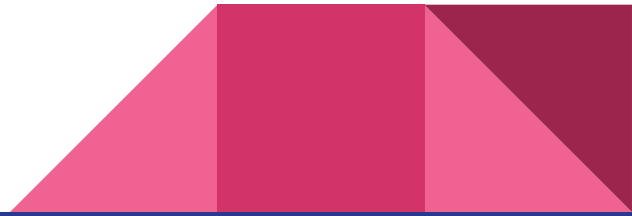
Variables

Strings - Activity 1

Control flow - Activity 2

Breakout - Activity 3

Homework 2



Breakout |

Make a calculator

Work with conditionals (if statements)

Save a .py file and execute it

HW preview |

Make .py files that are executed from the command line

You will ask for input and give output

Think about your user experience

- Provide easy to use menu with options spelled out
- Make readable (i.e. one dollar is “$1.00” not “1.0”)
- Anticipate and fix errors i.e. use <string>.lower()) to clean input

Unit 3 - Sequence Types and Dictionaries

[w18] Python For Data Science - Summer 2018

Unit 3 | Agenda

Sequences

Lists

Ranges, Tuples, and Sets - Activity 1

Dictionaries - Activity 2

Mutability Pitfalls - Activity 3

Assignment 1 Feedback

- How comfortable do people feel on github?
- Github folder structure – Please put week_01 homework files under SUBMISSIONS/week_01 folder (week_02 files would be under SUBMISSIONS/week_02, etc.)
- Reason to make file references relative – Some folks started with a cd to a long directory on their local computer. This won't run on a general user's computer since that user won't have that directory
- Capitalization matters! There is a programming difference between s1 and S1, which could cause errors if the two get mixed up.

Jupyter Notebook Stops Working

- If jupyter notebook looks like this for a long period of time:
  - ```
    In [*]: # YOUR CODE HERE
    ```
    Notice the: [*]
  - The [*] means that block of code is running and either:
    - There is an infinite loop in that code somewhere so it never finishes, or
    - The calculations are taking a long time to do (which probably isn't correct either)
  - This will prevent you from running any other blocks of code in that notebook!
- To get out of this state:
  - 1) Try: (menu) Kernel -> Interrupt (only works sometimes)
  - 2) Try: (menu) Kernel -> Restart (pop-up) - Restart (works most of the time)
  - 3) Shutdown / exit out of Jupyter Notebook and manually restart
    (This could cause a loss of work if not saved recently.)
    ```

Jupyter Notebook Variable Space

- Variables in Jupyter:
  - For example: x =4; print(x) in Jupyter
  - If you delete the x=4; you can still print(x)
  - x is stored in the notebook memory even though it isn't defined anymore
- This is a problem:
  - When we re-run your code "x" is not in our notebook's memory
  - Code crashes with: "x is undefined" error
  - Please go to the Kernel menu - restart and clear output
  - Then re-run all of your code blocks before turning it in!

Unit 3 | Agenda

Sequences

Lists

Ranges, Tuples, and Sets - Activity 1

Dictionaries - Activity 2

Mutability Pitfalls - Activity 3

Sequences

What are sequences?

- Define
- Name some

What are some types that are not sequences?

Sequences

What are sequences?

Define

Name some

In Python, sequence is the generic term for an ordered group of objects. Examples include lists, tuples, and strings.

What are some types that are not sequences?

Any data type without an inherent order, such as dictionaries, sets, ints, floats.

Methods for Sequences | Part 1

- **index or slice with [ ]**
  - **Starts with 0**
  - **index is offset**
- **len()**
- **in** **# (e.g. “5 in list_X”)**
- **not in**
- **+** **# can ‘add’ to concatenate**

Methods for Sequences | Part 2

- **max()**
- **min()**
- **seqX.index(‘x’)** **# locate the first instance of ‘x’**
- **seqX.count(‘x’)** **# count how many times ‘x’ is in the sequence**

What are the purpose of the parentheses?

When do we use the “.” (dot) notation?

Methods for Sequences | Part 2

- **max()**
- **min()**
- **seqX.index(‘x’)** **# locate the first instance of ‘x’**
- **seqX.count(‘x’)** **# count how many times ‘x’ is in the sequence**

What are the purpose of the parentheses?

The parentheses are used to pass arguments to a function (e.g., ‘x’). Some functions do not need arguments.

When do we use the “.” (dot) notation?

The dot notation indicates that a function is defined within a specific object. In the example above, the object “seqX” has both “index()” and a “count()” functions associated with it. In Python, all sequence objects have these functions defined.

Unit 3 | Agenda

Week 2 Assignment and Polls

Sequences

Lists

Ranges, Tuples, and Sets - Activity 1

Dictionaries - Activity 2

Mutability Pitfalls - Activity 3

Lists

Lists are a particularly versatile type of sequence

Lists are **mutable**

What does it mean to be mutable?

What other types are mutable? Which are not?

Lists

Lists are a particularly versatile type of sequence

Lists are **mutable**

What does it mean to be mutable?

Mutability refers to the ability to modify the object, in place, in memory.

What other types are mutable? Which are not?

Dictionaries and sets are mutable. Tuples and strings are not, though tuples can hold mutable objects within them. Primitive data types such as int, and float are also immutable.

Lists

Lists are a particularly versatile type of sequence

Lists are **composite types**

What does it mean to be a composite type?

What other types are composite? Which are not?

Lists

Lists are a particularly versatile type of sequence

Lists are **composite types**

What does it mean to be a composite type?

Composite types are comprised of other types. Lists, for example, can contain any other object within them.

What other types are composite? Which are not?

Tuples, dictionaries and sets are all composite types. Strings are not. Primitive objects such as ints and floats are also not composite types.

Mutation Methods for Lists | Part 1

```
ls_X.insert(index, value)

ls_X.pop(x)          # pops last value by default but can instead take index argument "x"

ls_X.remove()        # use remove command to remove first instance of value

ls_X.sort()          # this mutates the list

sorted(ls_X)         # this returns a new list

ls_X.reverse()       # reverses list
```

Note that the "sorted()" function is not called using the dot notation! It requires assignment:

```
list_2 = sorted(list_1)
```

Mutation Methods for Lists | Part 2

```
ls_x.append(x)          # adds x to end of list

ls_x.extend(list2)      # adds items from list2 to the end of the ls_x

ls_x[a] =               # swaps out the item at index [a] with whatever is provided

ls_x.clear()            # clears list

del(ls_x[a])            #  deletes item from index a
```

Unit 3 | Agenda

Sequences

Lists

Ranges, Tuples, and Sets - Activity 1

Dictionaries - Activity 2

Mutability Pitfalls - Activity 3

Tuples, Ranges, 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
>>>
```

Tuples | food for thought

Tuples are immutable but they can contain mutable data types!

What is 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
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Range Activity | Make these sequences

range(start, stop(exclusive), step)

[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]

Unit 3 | Agenda

Sequences

Lists

Ranges, Tuples, and Sets - Activity 1

Dictionaries - Activity 2

Mutability Pitfalls - Activity 3

Dictionaries | Define

- **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}
  - a = {‘names’: {‘fred’:1, ‘frank’:3, ‘ben’:1}} **# can be nested (JSON)**
- **Values can be any type**
- **Keys need to be hashable** **# should be immutable**

Dictionaries | Define

- **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}
  - a = {‘names’: {‘fred’:1, ‘frank’:3, ‘ben’:1}} **# can be nested (JSON)**
- **Values can be any type**
- **Keys need to be hashable** **# should be immutable**

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.

Dictionaries | Indexing

instantiation

- dict_x=dict(fred=1, frank=3, ben=1) # assign values to variables (no quotes)
- dict_x={‘fred’:1, ‘frank’:3, ‘ben’:1} # as a dict literal
- dict_x=dict ([(‘fred’:1),(‘frank’,3), (‘ben’, 1)]) # as a list of tuples (single object)
- dict_x=dict ([[‘fred’,1],[’frank’,3], [‘ben’, 1]]) # as a list of lists (single object)

Index by key to get value

Dict_x[‘fred’] # indexing by key name

Dict_x.fred # dot notation when there are no spaces

Dictionaries | More Methods

- **del**(dict_X['key']) # delete by key reference
- dict_X.**pop**('key', "default val") # pop the value for key from dictionary. If the key does not exist, the function will return the default
- dict_X.**get**('key', "default value')
- dict_X.**clear**()
- dict_X.**update**(dict2) # appends a second dictionary to the first
- dict_X.**keys**()
- dict_X.**values**()
- dict_X.**items**() # get the key:value pairs

List and 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.

Week 3 | Agenda

Week 2 Assignment and Polls

Sequences

Lists

Ranges, Tuples, and Sets - Activity 1

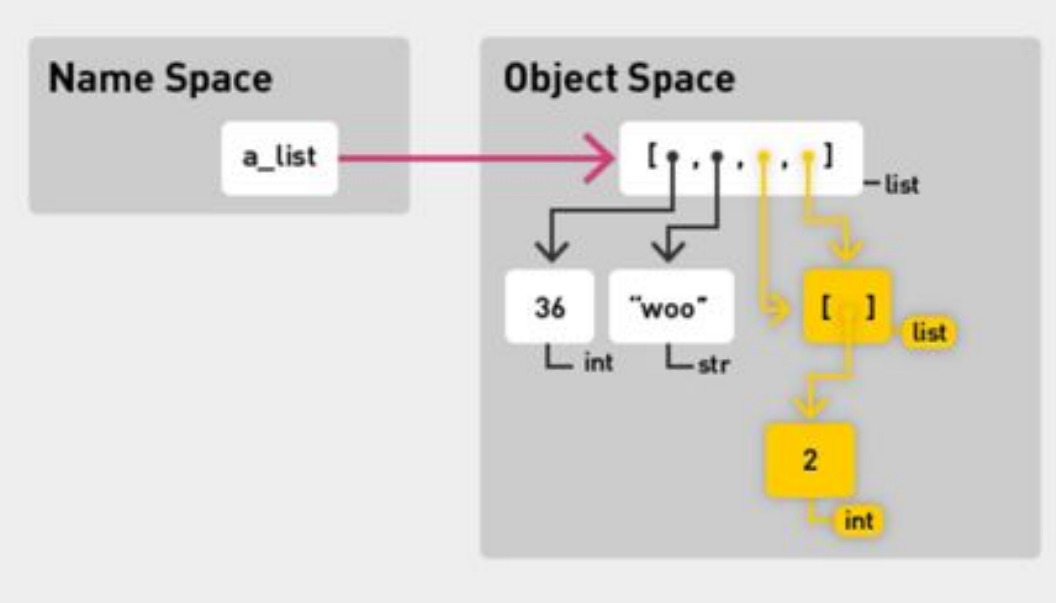
Dictionaries - Activity 2

Mutability Pitfalls - Activity 3

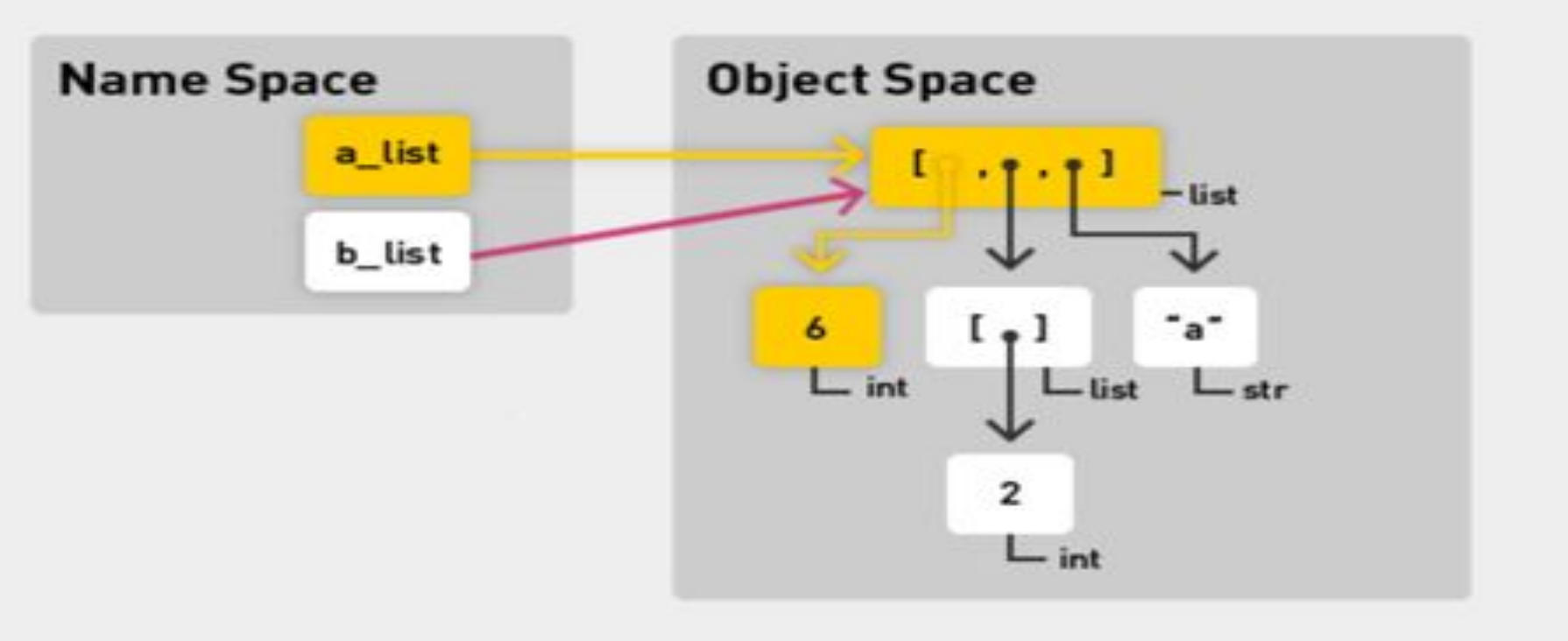
Mutability | Gotcha 1- this list is ***pointing to the same*** object

i.e. items 3 and 4 are the same object.

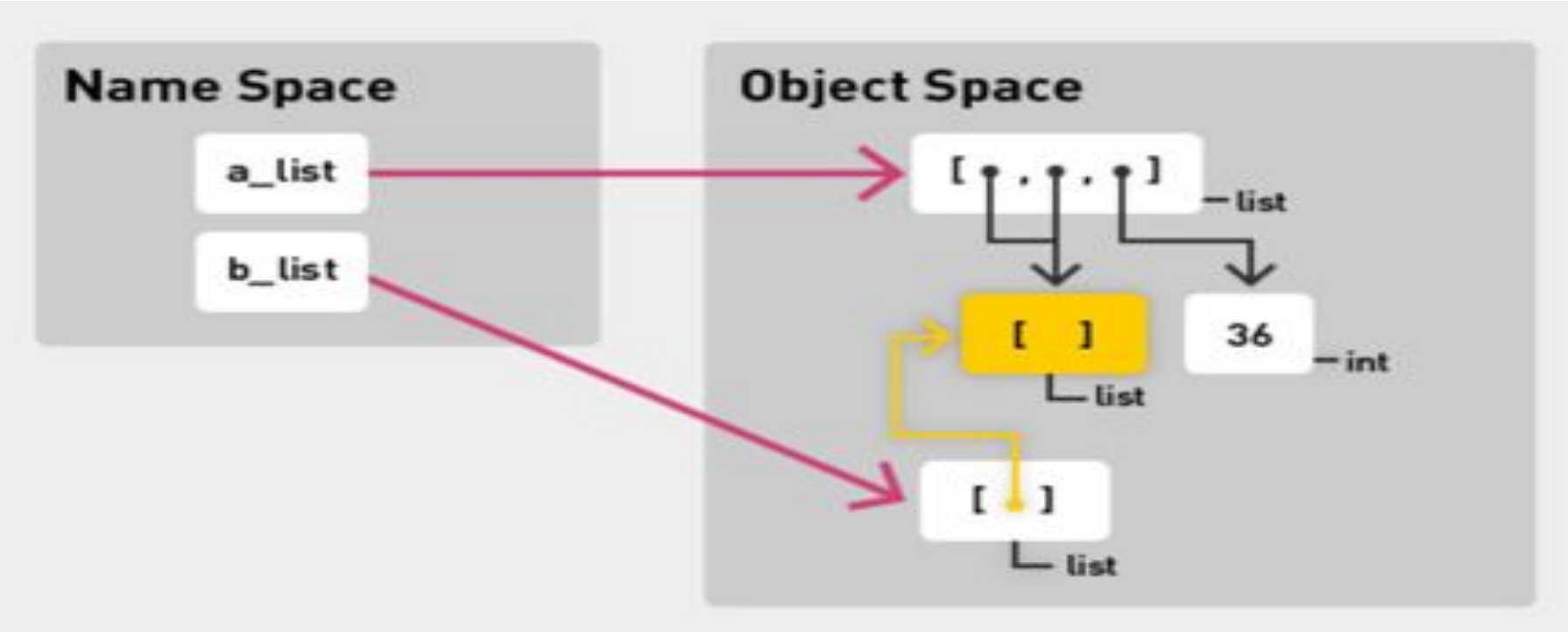
```
[36,"woo", [2], [2]]
```



Mutability | Gotcha 2- object has multiple names



Mutability | Gotcha 3 - object is in distinct lists



Copy and Deep Copy

Consider the code:

```
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’
```

What is copy?

How does copy differ from deepcopy?

What is the final value of Ls_x_cp and Ls_x_deep?

Copy and Deep Copy

Consider the code:

```
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’
```

What is copy?

Copy will create an independent copy of all list elements at the first level of the list

How does copy differ from deepcopy?

Deep copy will create an independent copy of all list elements at all levels

What is the final value of Ls_x_cp and Ls_x_deep?

Ls_x_cp is [1, 2, 3, [‘Frank’, ‘Mufasa’] Ls_x_deep is [1, 2, 3, [‘Frank’, ‘Fred’]

Mutability Activity

A score board reports the ranking and team color of contestants over a week long contest.

```
Contestants = [{"name":"fred", "teamColor":"Red"},
                {"name":"Layla", "teamColor":"Yellow"},
                 {"name":"Tammy", "teamColor":"Green"},
                {"name":"Buba", "teamColor":"Blue"}]
```

Your job is to programmatically change the score board as indicated in the exercise

Hint: use copy and/or deep copy if required