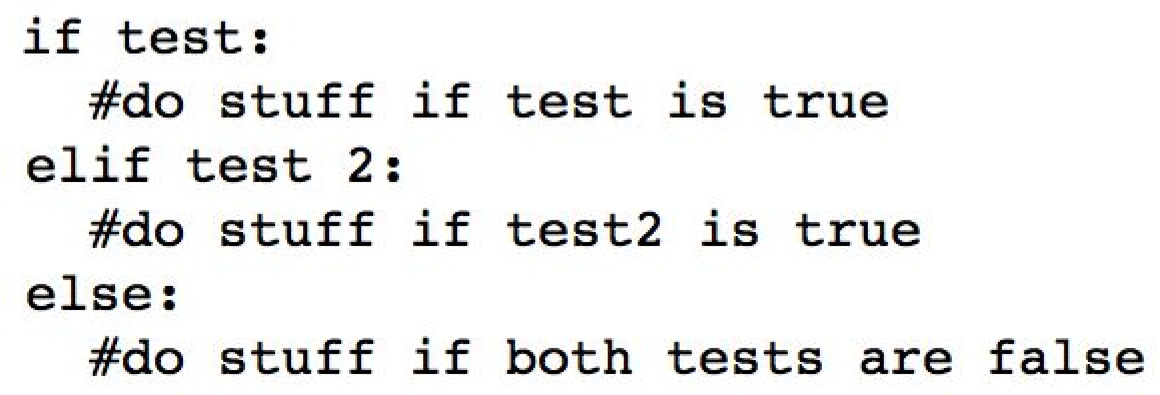
Mini-Review: Python Basics and Control Flow

Python Basics ⇒ Cheat Sheet

- Whitespace matters! Your code will not run correctly if you use improper indentation.
- Notice that this slicing and range() is 'exclusive' - it returns the 0th and the 1st element, but not the 2nd
- Other such notes...

If/Else



```
if test:
   #do stuff if test is true
elif test 2:
   #do stuff if test2 is true
else:
   #do stuff if both tests are false
```

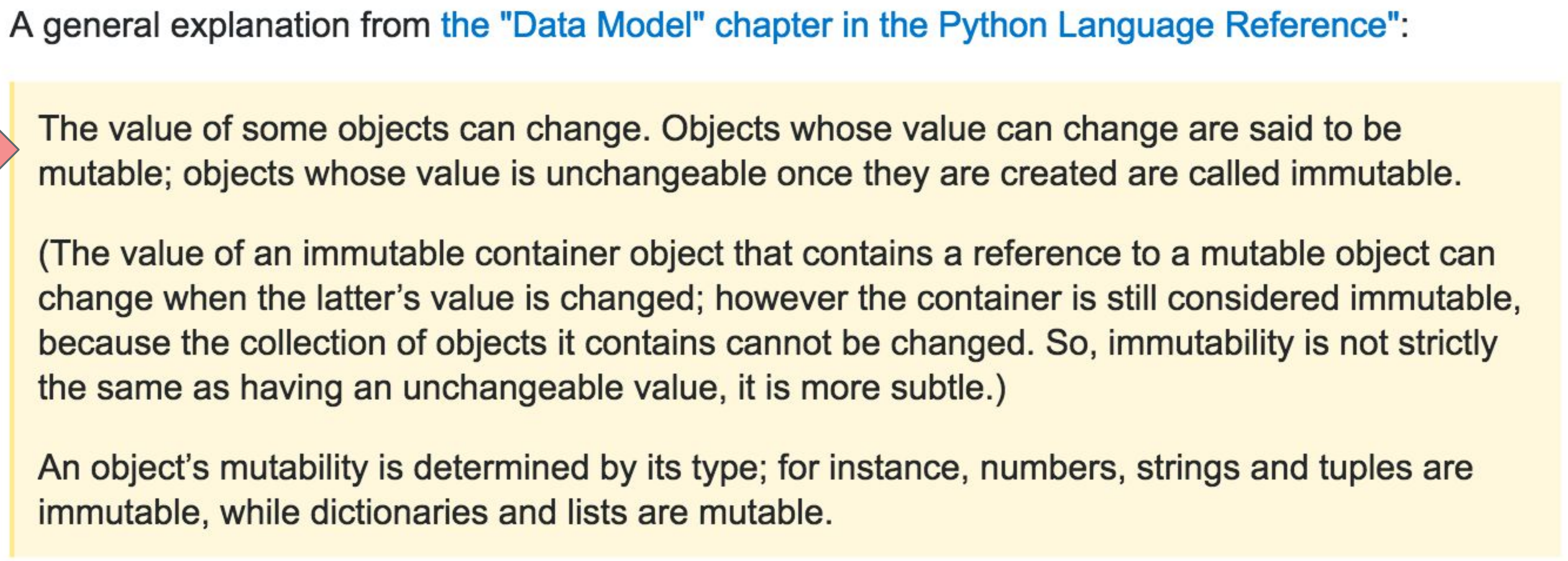
For Loop

```
for x in aSequence:
  #do stuff for each member of aSequence
  #for example, each item in a list, each
  #character in a string, etc.

for x in range(10):
  #do stuff 10 times (0 through 9)

for x in range(5,10):
  #do stuff 5 times (5 through 9)
```

Defining a Function

```
def myFunc(param1, param2):
    “””By putting this initial sentence in triple quotes, you can
    access it by calling myFunc.__doc___”””
    #indented code block goes here
    spam = param1 + param2
    return spam
```

Mutable vs Immutable: What’s the difference?



A general explanation from the "Data Model" chapter in the Python Language Reference":

> The value of some objects can change. Objects whose value can change are said to be mutable; objects whose value is unchangeable once they are created are called immutable.
>
> (The value of an immutable container object that contains a reference to a mutable object can change when the latter’s value is changed; however the container is still considered immutable, because the collection of objects it contains cannot be changed. So, immutability is not strictly the same as having an unchangeable value, it is more subtle.)
>
> An object’s mutability is determined by its type; for instance, numbers, strings and tuples are immutable, while dictionaries and lists are mutable.

Mutable vs Immutable: Why the heck do I care??

- Rule of thumb:
    - Primitive-like types (e.g., integers, floats) are probably immutable.
    - Container-like types (e.g. lists) are probably mutable.
- Mutability/Immutability matters for:
    - Dictionary keys
    - Function arguments
    - For loops
    - Memory consumption

Quick Check!

- What’s the difference between **.append()** and **.extend()**?
- Which container stores its items ordered?
- What does exclusive mean?

Hashability makes an object usable as a dictionary key and a set member, because these data structures use the hash value internally.

Using `try: except:` is preferred in cases where `if: else:` logic is more complicated. Simple is better than complex; complex is better than complicated; and it's easier to ask for forgiveness than permission.

lists

.append vs .extend

A container for holding items, or more formally, a Python data structure for holding python data types or other Python data structures. Items in a list are ordered and can be retrieved by the index of that item.

Explore the built in functions of lists:

- use .append(item) to add item to a list
- use .extend([item1, item2]) to join another list to the end of the list
- use .insert(index, item) to add an item to a list at an index
- use .remove(item) to remove an item from a list
- use .sort() to sort a list
- use .count(item) to count the number of times an item appears in a list
- use .index(item) to find the index of an element in a list
- use .pop() to extract (and remove) the last element of a list
- use .reverse() to reverse a list

dicts

A dictionary is a container that holds items using a key value pairs. **The items are stored unordered.** The items must be indexed by their key -- not by position.

Explore some of the built in functions for dictionaries:

- use **.pop(key)** to remove a key:value pair from the dictionary and return the value
- use **.get(key)** to get the value for a key
- use **.has_key(key)** to check if a key is in the dictionary
- use **.keys()** to get a list of the keys in the dictionary
- use **.items()** to get a list of the key:value pairs in the dictionary
- use **.update(other_dictionary)** to merge a 2nd dictionary into the current dictionary
- use **.clear()** to remove all key:value pairs from the dictionary

Conclusion:

- Lists and dictionaries are the two most popular data structures used in Python
- Lists and dictionaries are both mutable
- Lists are indexed by postion and dictionaries are indexed by keys