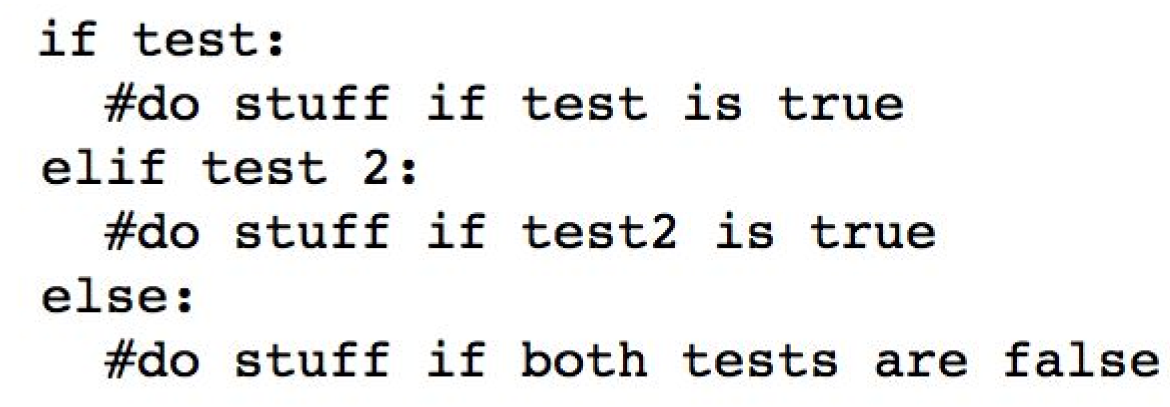
Mini-Review: Python Basics and Control Flow

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

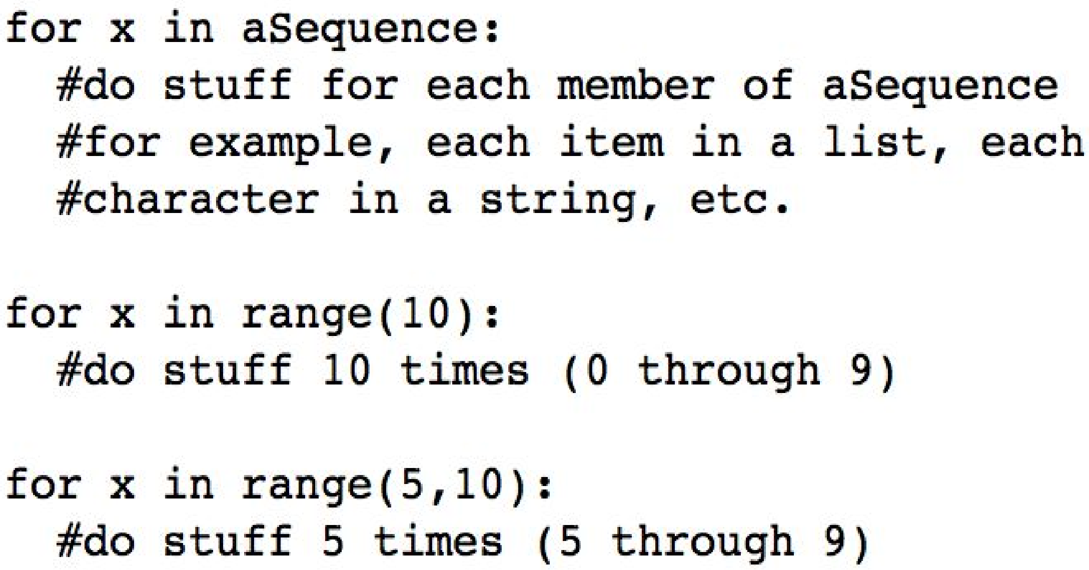


```
if test:
   #do stuff if test is true
elif test 2:
   #do stuff if test2 is true
else:
   #do stuff if both tests are false
```

Main things to think about:

- **All the conditions you need to consider for your end goal(s)**
- **The order of the conditions**

For Loop

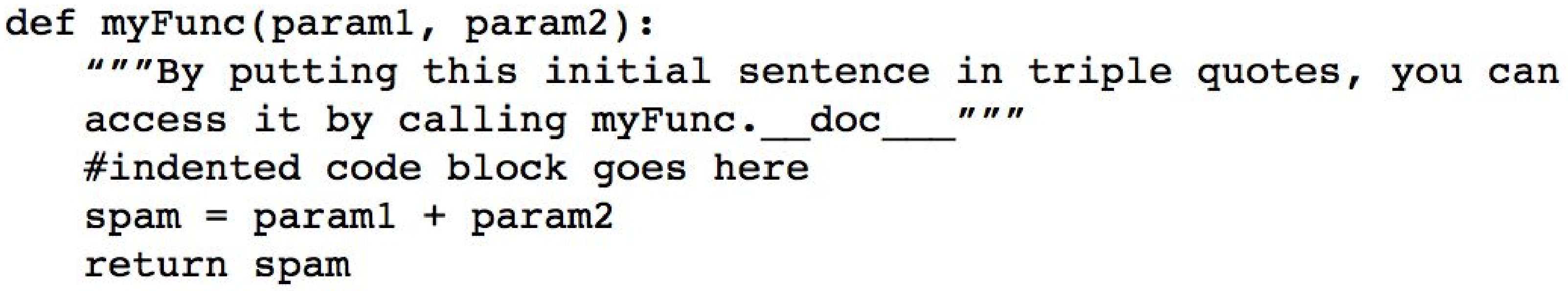


```
for x in aSequence:
  #do stuff for each member of aSequence
  #for example, each item in a list, each
  #character in a string, etc.

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)
```

Main things to think about:

- **Mutability vs. immutability**
- **When setting the temporary variable, think about what you are actually referring to.**

Defining a Function

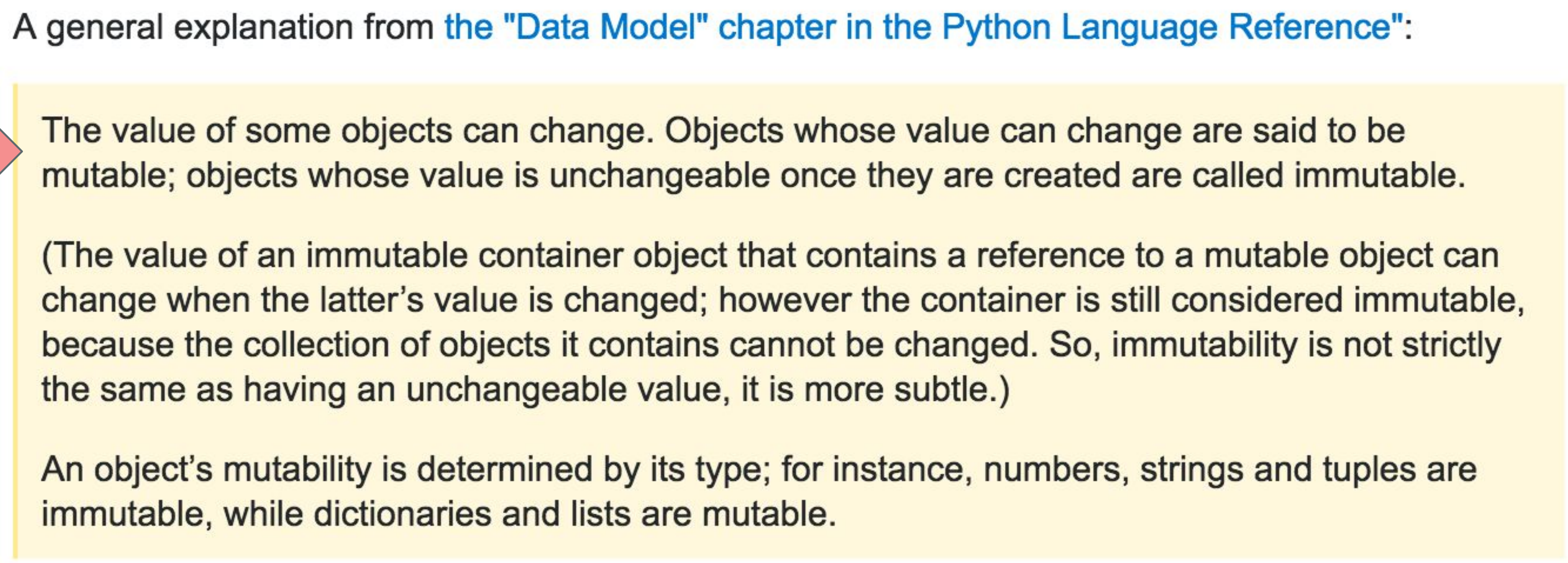


```
def myFunc(param1, param2):
    """By putting this initial sentence in triple quotes, you can
    access it by calling myFunc.__doc___"""
    #indented code block goes here
    spam = param1 + param2
    return spam
```

Main things to think about:

- **Start practicing the use of docs (denoted by 3 quotation marks: “““ ”””) or descriptive comments (preceded by #)**
- **Struggling about where to start? Break down the end goal of the function into steps.**

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?