**Lab 01: Python Refresher.**

The programming assignments in this course will be written in Python, an interpreter, object-oriented language that shares some features with both Java and Scheme. This tutorial will walk through the primary syntactic constructions in Python, using short examples. We encourage you to type all python shown in the tutorial onto your own machine. Make sure it responds the same way. You may find the Troubleshooting section helpful if you run into problems. It contains a list of the frequent problems previous students have encountered when following this tutorial.

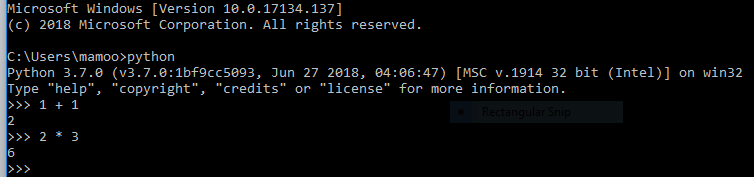
1. **Invoking the Interpreter**

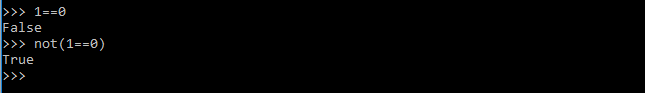
Python can be run in one of two modes. It can either be used *interactively*, via an interpeter, or it can be called from the command line to execute a *script*. We will first use the Python interpreter interactively. You invoke the interpreter by entering python at the command prompt.   
Note: you may have to type python2.4, python2.5, python2.6 or python2.7, rather than python, depending on your machine.

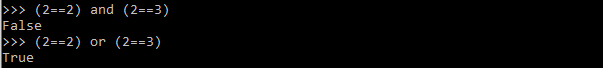
[Lab ~]$ python   
Python 2.6.5 (r265:79063, Jan 14 2011, 14:20:15)   
Type "help", "copyright", "credits" or "license" for more information.   
>>>

1. **Operators**

The Python interpreter can be used to evaluate expressions, for example simple arithmetic expressions. If you enter such expressions at the prompt (>>>) they will be evaluated and the result will be returned on the next line.

Boolean operators also exist in Python to manipulate the primitive True and False values.

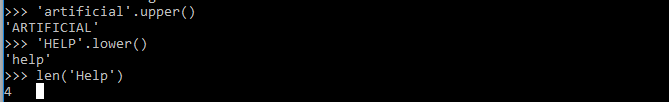




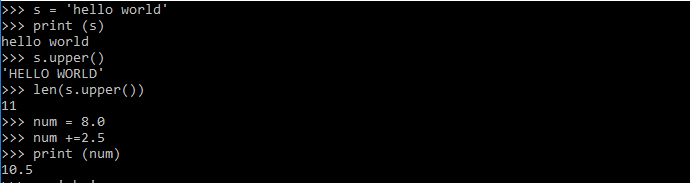
1. **Strings**

Like Java, Python has a built in string type. The + operator is overloaded to do string concatenation on string values.

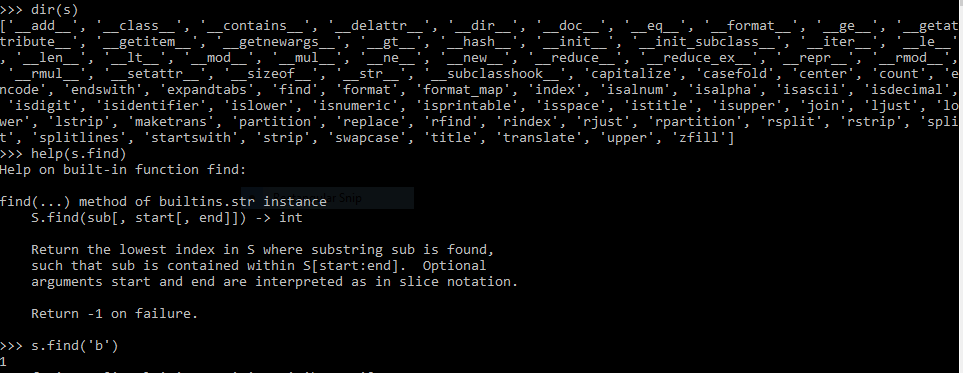
There are many built-in methods which allow you to manipulate strings.

Notice that we can use either single quotes ' ' or double quotes " " to surround string. This allows for easy nesting of strings.

We can also store expressions into variables.

In Python, you do not have declare variables before you assign to them.

1. **Dir and Help**

Learn about the methods Python provides for strings. To see what methods Python provides for a datatype, use the dir and help commands: Try out some of the string functions listed in dir (ignore those with underscores '\_' around the method name).

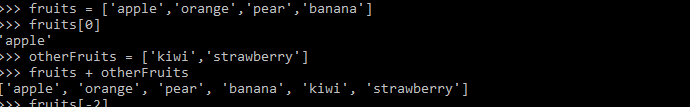
Built-in Data Structures

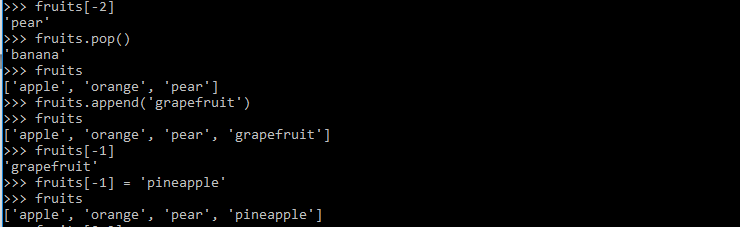
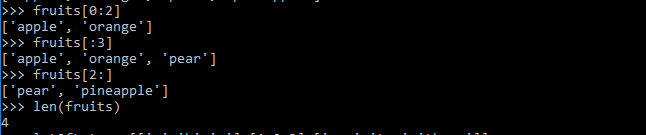
Python comes equipped with some useful built-in data structures, broadly similar to Java's collections package

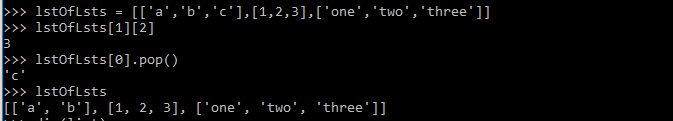
1. Lists

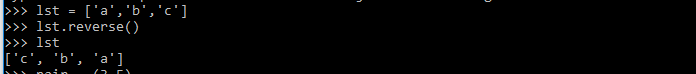
Lists store a sequence of mutable items:

We can use the + operator to do list concatenation:



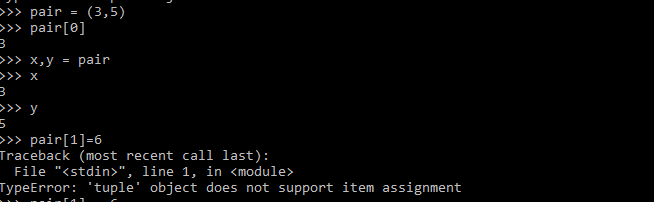
Python also allows negative-indexing from the back of the list. For instance, fruits[-1] will access the last element 'banana': We can also index multiple adjacent elements using the slice operator. For instance, fruits[1:3], returns a list containing the elements at position 1 and 2. In general fruits[start:stop] will get the elements in start, start+1, ..., stop-1. We can also do fruits[start:] which returns all elements starting from the start index. Also fruits[:end] will return all elements before the element at position end: The items stored in lists can be any Python data type. So for instance we can have lists of lists:

Exercise: Lists

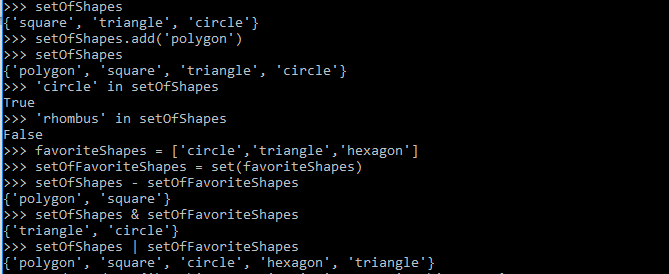
Play with some of the list functions. You can find the methods you can call on an object via the dir and get information about them via the help command:  

Note: Ignore functions with underscores "\_" around the names; these are private helper methods. Press 'q' to back out of a help screen.

1. Tuples

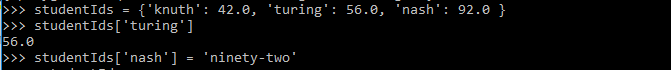
A data structure similar to the list is the tuple, which is like a list except that it is immutable once it is created (i.e. you cannot change its content once created). Note that tuples are surrounded with parentheses while lists have square brackets. The attempt to modify an immutable structure raised an exception. Exceptions indicate errors: index out of bounds errors, type errors, and so on will all report exceptions in this way.

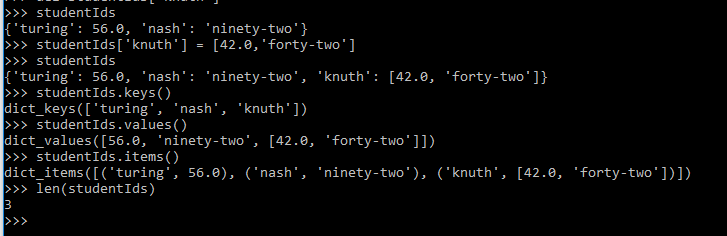
1. **Sets**

A set is another data structure that serves as an unordered list with no duplicate items. Below, we show how to create a set, add things to the set, test if an item is in the set, and perform common set operations (difference, intersection, union): **Note that the objects in the set are unordered; you cannot assume that their traversal or print order will be the same across machines!**

1. **Dictionaries**

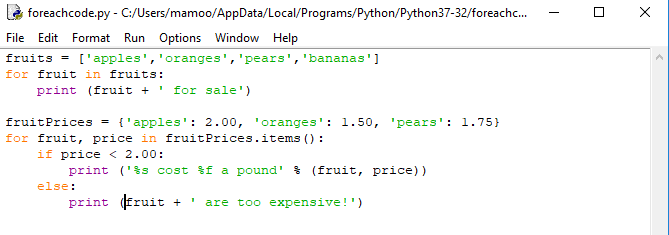
The last built-in data structure is the dictionary which stores a map from one type of object (the key) to another (the value). The key must be an immutable type (string, number, or tuple). The value can be any Python data type.

Note: In the example below, the printed order of the keys returned by Python could be different than shown below. The reason is that unlike lists which have a fixed ordering, a dictionary is simply a hash table for which there is no fixed ordering of the keys (like HashMaps in Java). The order of the keys depends on how exactly the hashing algorithm maps keys to buckets, and will usually seem arbitrary. Your code should not rely on key ordering, and you should not be surprised if even a small modification to how your code uses a dictionary results in a new key ordering.  

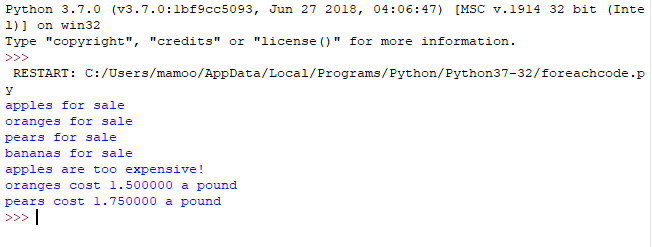


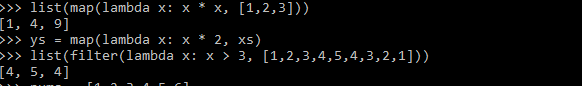
As with nested lists, you can also create dictionaries of dictionaries.

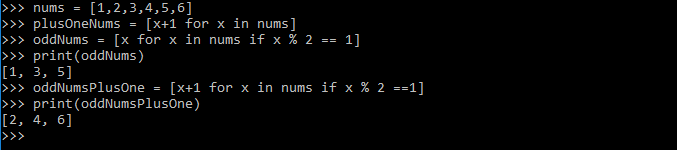
1. **Writing Scripts**

Now that you've got a handle on using Python interactively, let's write a simple Python script that demonstrates Python's for loop. Open the file called foreach.py and update it with the following 

save the file in foreach.py, and then run :

Remember that the print statements listing the costs may be in a different order on your screen than in this tutorial; that's due to the fact that we're looping over dictionary keys, which are unordered.

If you like functional programming you might also like map and filter: 

The next snippet of code demonstrates Python's list comprehension construction: **Beware of Indendation!**

Unlike many other languages, Python uses the indentation in the source code for interpretation. So for instance, for the following script:

if 0 == 1:

print 'We are in a world of arithmetic pain'

print 'Thank you for playing'

will output

Thank you for playing

But if we had written the script as

if 0 == 1:

print 'We are in a world of arithmetic pain'

print 'Thank you for playing'

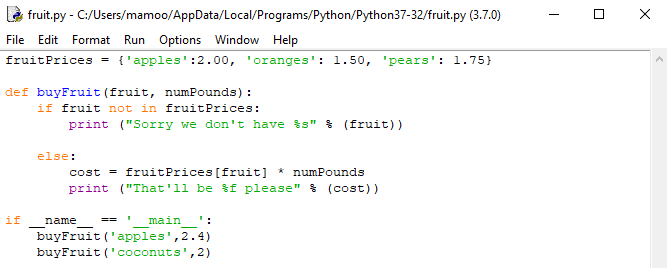
there would be no output. The moral of the story: be careful how you indent! It's best to use four spaces for indentation -- that's what the course code uses.

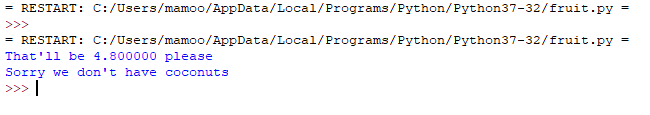
1. **Tabs vs Spaces**

Because Python uses indentation for code evaluation, it needs to keep track of the level of indentation across code blocks. This means that if your Python file switches from using tabs as indentation to spaces as indentation, the Python interpreter will not be able to resolve the ambiguity of the indentation level and throw an exception. Even though the code can be lined up visually in your text editor, Python "sees" a change in indentation and most likely will throw an exception (or rarely, produce unexpected behavior).

This most commonly happens when opening up a Python file that uses an indentation scheme that is opposite from what your text editor uses (aka, your text editor uses spaces and the file uses tabs). When you write new lines in a code block, there will be a mix of tabs and spaces, even though the whitespace is aligned.

1. Writing Functions

As in Java, in Python you can define your own functions: Rather than having a main function as in Java, the \_\_name\_\_ == '\_\_main\_\_' check is used to delimit expressions which are executed when the file is called as a script from the command line. The code after the main check is thus the same sort of code you would put in a main function in Java.

Save this script as fruit.py and run it: 

**Exercises:**

1. Write a list comprehension which, from a list, generates a lowercased version of each string that has length greater than five.
2. Let we have a dictionary D which contain roll number of student as key and its values shall be a list. The list has tuples as its item. Each tuple represent the name of the subject and gpa of that subject. You can visualize the data structure as of following:

|  |  |
| --- | --- |
| **key** | **Value** |
| **2016-CS-700** | **[(DSA,3),(Algo,2.5),(AI,3)]** |
| **2016-CS-701** | **[(LA,3),(Algo,3.5),(PF,2.8)]** |
| **…** | **…** |
| **…** | **..** |
| **2016-CS-710** | **[(OOP,3),(DB,3.5),(PF,2.8)]** |

1. **Write a program which shall print GPA of all students. Assume all subject have 3 credit hours.**
2. **Write a program that print the highest number obtained in DSA**
3. **Write a program that print the total number of student those have less than 2.5 GPA in AI?**