Introduction to Programming for Scientists

Lecture 2

Introduction

Datatypes

Lists and Tuples

```
>>> a=[1,2,3,4]                  # a list of numbers
>>> a
[1, 2, 3, 4]
>>> a[3]
4
>>> sum(a)
10
>>> a=["Hello",2,3]              # a list with a string and 2 numbers
>>> a
['Hello', 2, 3]
>>> sum(a)
Traceback (most recent call last):
  File "<pyshell#84>", line 1, in -toplevel-
    sum(a)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> a[0]=1
>>> sum(a)
6
>>> a
[1, 2, 3]
```

Lists and Tuples

```
>>> a=range(10)                    # makes a list of numbers from 0 thru 9
>>> a
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> a=range(5,20,2)                # makes a list of numbers starting at 5 ending before 20 step by 2
>>> a
[5, 7, 9, 11, 13, 15, 17, 19]
>>> a[2:5]
[9, 11, 13]

>>> a=(1,2,3,4)                    # a tuple, like a list, but elements cannot be changed
>>> a
(1, 2, 3, 4)
>>> a[2]=0
Traceback (most recent call last):
  File "<pyshell#109>", line 1, in -toplevel-a[2]=0
TypeError: object doesn't support item assignment
>>> b=list(a)
>>> b[2]=0
>>> b
[1, 2, 0, 4]
```

Dictionaries

```
>>> a={"a":0,"boat":4,"in":2,"the":3,"river":5}      # a dictionary. Associates items with
>>> a.keys()                                         # other items. The key is used for
['a', 'the', 'river', 'boat', 'in']                  # retrieving the value. The key must be
>>> a.values()                                       # immutable (number, string, tuple)
[0, 3, 5, 4, 2]
>>> a["a"]
0
>>> a["river"]
5
>>> a["a"]+a["boat"]+a["river"]
9
>>> a["tree"]="root"
>>> a
{'a': 0, 'tree': 'root', 'in': 2, 'the': 3, 'river': 5, 'boat': 4}
>>> a["a"]+a["boat"]+a["tree"]
Traceback (most recent call last):
  File "<pyshell#140>", line 1, in -toplevel-
    a["a"]+a["boat"]+a["tree"]
TypeError: unsupported operand type(s) for +: 'int' and 'str'

>>> str(a["a"])+str(a["boat"])+a["tree"]
(what do you think?)
```

Help!

Python reference manual online (also often installed with python):
http://docs.python.org/lib/lib.html

```
>>> help()
...
>>> help(str)
...
>>> help(str.join)
...
>>> help(list.sort)
```

Methods of Standard Types

Methods are functions applied to an object. For example:

a=[4,2,3,1]

a is now a 'list' object

a.sort()

applies the 'sort' method to the object 'a'.

Methods of Standard Types

Useful list methods:

```
>>> a.append(5)                # append adds a single item to a list
>>> a
[1,2,3,4,5]
>>> a.extend([1,2,3])          # extend adds a list to a list
>>> a
[1,2,3,4,5,1,2,3]
>>> a.count(3)                 # counts the number of 3's
2
>>> a.index(2)                 # finds the first 2 in the list
1
>>> a.remove(2)                # removes the first 2 from the list
>>> a
[1, 3, 4, 5, 1, 2, 3]
>>> 5 in a
True
```

More Fun with Strings

Useful string methods:

```
>>> a=”My name is Fred”
>>> a.capitalize()
'My name is fred'
>>> a.find("is")
8
>>> a.upper()
'MY NAME IS FRED'
>>> b=a.center(40)
>>> b
'            My name is Fred             '
>>> b.strip()
'My name is Fred'
>>> a.title()
'My Name Is Fred'
```

More Fun with Strings

```
>>> a="My,name,is,Steven,Ludtke"
>>> b=a.split(",")                        # chops a string into substrings with the given separator
>>> b
['My', 'name', 'is', 'Steven', 'Ludtke']
>>> c=" ".join(b)
>>> c
'My name is Steven Ludtke'
>>> b[2]="isn't"
>>> " ".join(b)
"My name isn't Steven Ludtke"
>>> b.sort()
>>> b
['Ludtke', 'My', 'Steven', 'is', 'name']
>>> " ".join(b)
'Ludtke My Steven is name'
>>> len(b)
5
>>> len(" ".join(b)
24
```

More Fun with Strings

If a string is the first item in a function definition, it documents the function

```
>>> def f(x):
        “takes a string input and sorts the letters”
        b=list(x)
        b.sort()
        return “”.join(b)
>>> f(“steven ludtke”)
'  deeeklnsttuv'

>>> help(f)
f(x)
        takes a string input and sorts the letters
```

Dictionaries are Fun Too

```
>>> a={"a":1,"b":2,"c":3,"d":4,"e":5}
>>> a.keys()
['a', 'c', 'b', 'e', 'd']
>>> a.values()
[1, 3, 2, 5, 4]
>>> a.items()
[('a', 1), ('c', 3), ('b', 2), ('e', 5), ('d', 4)]
>>> a.has_key(“b”)
True
>>> a[“z”]=26
>>> a
{'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4, 'z': 26}
>>> b={'e':5,'f':6}
>>> a.update(b)                 # change the values in one dict based on another
>>> a
{'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4, 'f': 6, 'z': 26}
```

Functions (not methods)

```
>>> a=[-10,1,2,3,4,5,99]
>>> max(a)                        # a function is of the form f(x) instead of x.f()
99
>>> min(a)
-10
>>> len(a)
7
>>> range(2,9,3)
[2, 5, 8]
>>> zip(('a','b','c'),(1,2,3))
[('a', 1), ('b', 2), ('c', 3)]
```

int(), float(), str(), list(), tuple() and dict() for type conversion

We Have Learned so Far

- 6 Main Python Datatypes:
  - int
  - float
  - str
  - tuple
  - list
  - dict
- How to define functions
- How to 'import' libraries of functions
- How to get help

Basic Syntax Reference

- Indent
- Numbers:

0 1.5 1.2e4 3+2j

- Strings:

“test string” 'this too'

“””multiple line

string”””

- Lists:

lst=[1,2,'abc',1+3j]

lst[0]

- Dictionaries:

dict={'key1':'value1','key2':'value2',3:'value3'}

dict['key2']

dict[3]

- import:

import os

from math import *

- print:

print 'x=',x,' y=',y

print 'x=%f y=%f'%(x,y)

- help:

help(str)

help(list.sort)

Homework

Email to me by noon Tuesday

- Write a function to return the name from the following sentence, irrespective of what the name is. The rest of the sentence won't change :

```
a=”My name (George Jones) is very nice.”
print yourfunction(a)
George Jones
print yourfunction(“My name (Fred) is very nice.”)
Fred
```

- Write a function to count the number of commas in a sentence:

```
a=”It is, very strange, to put, commas everywhere, like this”
print yourfunction2(a)
4
```