Introduction to Programming for Scientists

LECTURE 2:
CONDITIONS, LOOPS & VARIABLES

Prof. Steven Ludtke
N410.07, sludtke@bcm.edu

Reminder

- Class material at:

http://blake.bcm.edu/IP14

- If you missed the first lecture, it is archived on the site above.

Expectations
(Yours, not mine)

- Concepts
- Syntax
- When to use each concept

Python

- **Data storage**
  - **‘simple’ types - numbers, strings, ...**
  - **compound types - lists, dictionaries, sets, ...**
- Operate on data
  - statements - a=b*10, print b*5+3, ...
  - functions - sin(a), len(x), ...
  - methods (functions on an object) - “abc”.count(“b”)
- Program Flow
  - for ... in ...
  - if, else
  - while ()
- Interact with the outside world
  - User interactions - raw_input()
  - Disk and other device access - file i/o

Python

- Data storage
  - ‘simple’ types - numbers, strings, ...
  - compound types - lists, dictionaries, sets, ...
- **Operate on data**
  - **statements - a=b*10, print b*5+3, ...**
  - **functions - sin(a), len(x), ...**
  - **methods (functions on an object) - “abc”.count(“b”)**
- Program Flow
  - for ... in ...
  - if, else
  - while ()
- Interact with the outside world
  - User interactions - raw_input()
  - Disk and other device access - file i/o

Functions vs. Methods

- Functions : sin(x), cos(y), len(s)
  - normally return a value
  - Not type-specific
- Methods : st.upper(), lst.append(5), lst.sort()
  - functions applied to a specific “object”
  - don't always return anything
  - methods are type-specific

Some Built-in functions

- int, float, str, list, tuple, set, dict - Converts between types
- range, xrange - makes a list (or iterator) covering a range
- input & raw_input
- len
- max,min
- reversed, sorted
- print (actually a statement in Python2)

Useful but more advanced:

- zip(list1,list2,...), zip(*zipped)
- enumerate

Methods of Strings

- Remember strings are immutable !
- upper, lower, title, capitalize
- count, find, rfind, index
- replace
- split
- join
- **in** (not really a method)

Lists

```
[item1,item2,item3,...]   # items can be anything
(item1,item2,item3,...)   # A tuple is an immutable list
a=[0,1,2,3,4,5,6]         # A list of 7 numbers
a[n]                      # nth element in list
a[n:m]                    # sublist elements n to m-1
a[-n]                     # nth item from the end
a[3] -> 3
a[1:4] -> [1,2,3]
a[-2] -> 5
a[2:-2] -> [2,3,4]
a[2]="x" -> [0,1,"x",3,4,5,6]
```

List Methods

- append, extend
- del, remove
- count
- index
- reverse, sort

Dictionaries

- keys must be immutable, values can be any type
- { k1:v1, k2:v2, k3:v3, ... }

  Example:

  a={ 1:2,2:3,”a”:”b”,2.0:3.2,(1,2):”really?” }

  a[1] -> 2

  a[(1,2)] -> “really?”

  a[2] -> 3.2
- Methods:
  - keys, values, items
  - has_key
  - set_default

Sets

- Sets have no order and elements are unique
- set([1,2,3,4,5])
- methods:
  - add, remove, discard, clear
  - issubset, issuperset
  - union, intersection, difference

Programs you can Run

- Do NOT use a word-processor like Word, Pages, etc. Use a 'text editor'. The built-in editor 'idle' is a good choice for beginners.
- Just type 'python program.py' -or- :
- use a '.py' extension for your programs
- for unix/mac, put:

  #!/usr/bin/env python

  on the first line of the file, and type:

  chmod a+x file.py
- NOTE: on windows, as soon as the program exits, the window showing the output will close. If you put a raw_input() at the end of your program, it will wait until you press enter before closing the window so you can see the output.

Writing Actual Programs

- How would you display a table of x vs sin(x) for x from 0 to 2π in steps of π/4 ?

Suboptimal, but functional

```
from math import *

print 0,sin(0)
print pi/4,sin(pi/4)
print pi/2,sin(pi/2)
print 3*pi/4,sin(3*pi/4)
print pi,sin(pi)
print 5*pi/4,sin(5*pi/4)
print 3*pi/2,sin(3*pi/2)
print 7*pi/4,sin(7*pi/4)
print 2*pi,sin(2*pi)
```

Umm... slightly better ?

```
from math import *
x=[0,pi/4,2*pi/4,3*pi/4,4*pi/4,5*pi/4,6*pi/4,7*pi/4,8*pi/4]

print x[0],sin(x[0])
print x[1],sin(x[1])
print x[2],sin(x[2])
print x[3],sin(x[3])
print x[4],sin(x[4])
print x[5],sin(x[5])
print x[6],sin(x[6])
print x[7],sin(x[7])
print x[8],sin(x[8])
```

Now What ?

- How would you display a table of x vs sin(x) for x from 0 to 2π in steps of π/64 ?

Python

- Data storage
  - ‘simple’ types - numbers, strings, ...
  - compound types - lists, dictionaries, sets, ...
- Operate on data
  - statements - a=b*10, print b*5+3, ...
  - functions - sin(a), len(x), ...
  - methods (functions on an object) - “abc”.count(“b”)
- **Program Flow**
  - **for ... in ...**
  - **if, else**
  - **while ()**
- Interact with the outside world
  - User interactions - raw_input()
  - Disk and other device access - file i/o

Program Flow

- **for** *i* **in** *list*:
- **if** *condition* :
  - Boolean operators
    - >, <, <=, >=, ==, !=, and, or, not, in
- **elif** *condition* :
- **else** :
- **while** *condition* :

for Loops

- Execute 'code' for each item in list, assigning the element to 'var' in each cycle:

  for *var* **in** *list:*

  code

  Example:

  a=[1,2,3,4,5]

  for i in a:

  print i,i*2

Umm... slightly better ?

```
from math import *
x=[0,pi/4,2*pi/4,3*pi/4,4*pi/4,5*pi/4,6*pi/4,7*pi/4,8*pi/4]

print x[0],sin(x[0])
print x[1],sin(x[1])
print x[2],sin(x[2])
print x[3],sin(x[3])
print x[4],sin(x[4])
print x[5],sin(x[5])
print x[6],sin(x[6])
print x[7],sin(x[7])
print x[8],sin(x[8])
```

Better !

```
from math import *
x=[0,pi/4,2*pi/4,3*pi/4,4*pi/4,5*pi/4,6*pi/4,7*pi/4,8*pi/4]

for i in x:
   print i,sin(i)
```

More improvement...

```
from math import *
x=range(9)

for i in x:
   j=i*pi/4
   print j,sin(j)
```

Would this work ?

```
from math import *
x=range(9)
for i in x: i=i*pi/4

for i in x:
   print i,sin(i)
```

Try this instead

```
from math import *
x=range(9)
for i in range(len(x)): x[i]*=pi/4

for i in x:
   print i,sin(i)
```

Works, but not very satisfying...

List Generators

- A **for** loop inside a list definition !
- [*x... **for** x **in** y*]

  example:

  a=[0,1,2,3,4,5,6,7]

  a=[i**2 for i in a]

  print a

  [0,1,4,9,16,25,36,49]

Much Better !

```
from math import *
x=[i*pi/4 for i in range(9)]

for i in x:
   print i,sin(i)
```

Writing Actual Programs

- How would you display a table of x vs sin(x) for x from 0 to 4π in steps of π/8, but only include values where sin(x) > 0 ?

Start with this

```
from math import *
x=[i*pi/8 for i in range(33)]

for i in x:
   print i,sin(i)
```

But what about the sin(x) > 0 requirement ?

The **if** statement

- Boolean operators
  - >, <, <=, >=, ==, !=, and, or, not, in
- **if** *condition* :
- **elif** *condition* :
- **else** :

Tack in our if

```
from math import *
x=[i*pi/8 for i in range(33)]

for i in x:
   if sin(i)>0 :
      print i,sin(i)
```

Comments

- Anything after ‘#’ on a line is a comment

Tack in our if

```
from math import *
# This generates our x-values
x=[i*pi/8 for i in range(33)]

for i in x:                 # loop over x-values
   if sin(i)>0 :            # only print if sin(x)>0
      print i,sin(i)
```