RESEARCH TOOLS 2011
LECTURE 10
2011-Oct-04
Kurt Schwehr
http://schwehr.org
UNH CCOM/JHC

ipython and python data types

Our first full lecture covering python
I used the “Tom Boy”
Wednesday, October 5, 11

I tried using the emacs ERC client for IRC during this class. That was tough as I really do not have enough space up on the projector.

M-x erc
IRC server: researchtools.com.nh
port: 6667
set your nickname to something other than researchtools!!!!
/join #unhresearchtools
schwehrvm on irc.debian.org:6667 (+)

[Tue Oct 4 2011]
*** Welcome to the Internet Relay Network schwehrvm!~user@192.168.17.68
*** Your host is irc.debian.org, running version ngircd-15 (i486.pc/linux-gnu)
*** This server has been started Sun Sep 25 2011 at 16:05:06 (EDT)
*** irc.debian.org ngircd-15 aiob biKlmnoPstv
*** RFC2812 CASEMAPPING=ascii PREFIX=(ov)@+ CHANTYPES=###+
*** CHANMODES=bi,k,l,lmnpsf chanlimit=#&:10 are supported on this server
*** CHANELLEN=50 NICKLEN=9 TOPICLEN=490 AWAYLEN=127 KICKLEN=400 PENALTY are supported on this server
*** There are 2 users and 0 services on 1 servers
*** 2 channels formed
*** I have 2 users, 0 services and 0 servers
*** Current local users: 2, Max: 17
*** Current global users: 2, Max: 17
*** - irc.debian.org message of the day
*** - ***********************
*** - HELLO
*** - * Welcome on our new IRC server, running on: *
*** - * Debian GNU/Linux (www.debian.org) *
*** - ***********************
*** End of MOTD command

ERC> /
Lecture 11: ipython and python data structures

http://vislab-ccom.unh.edu/~schwehr/rt/11-ipython.html

Kurt Schwehr, 2011-Oct-4

Make sure you have this org file for class open in emacs from
~/class/11/11-ipython-matplotlib.org

Open a terminal and =cd ~/class/11=.

* Introduction

schwehrvm on #unhresearchtools (+,lag:0) UNH channel for ubuntu, bash, python, emacs

*** berrya (~chatzill@lab2.ccom.nh) has joined channel #unhresearchtools
*** sthein (~chatzill@lab9.ccom.nh) has joined channel #unhresearchtools

<sthein> Hi!
<berrya> good morning. Should we open this in Emacs?

<schwehrvm> yes. open the org file in emacs

*** gmasetti (~chatzill@192.168.8.241) has joined channel #unhresearchtools

<prasadh> hi
Here's a quick video preview: http://www.wolfram.com/common/includes/m8videos/quicktour.html

** UNH Signals

UNH has an Information Technology (IT) newsletter called Signals.

Vantagepoint: Inside UNH IT with Chief Information Officer Joanna Young

You can meet our CIO. If you read her answers to questions, you will see that I think she missed something *huge* with passwords: all your passwords should be different. I worry less about people changing their passwords and more about each password being unrelated to compartmentalize any damage. If someone grabbed your password for a site requiring your password, there is a non-zero chance that if you change your password that they will get that new password the same way.
** What was that strange command in the homework?

Did you try it with an echo command?

```sh
#BEGIN_SRC sh
echo ~/hw/03/log-$USER-$\(date +%Y%m%d\).org
#END_SRC
```

On my Mac laptop, I get:

```bash
#RESULTS:
```

---

** Lecture 11: ipython and tools**

- Ubuntu, bash, python, emacs, org-babel, and
  Python 3.4
- [2011/esci895-researchtools/src/]
  [11:10]
- #unhresearchtools
- Ben Hassan Prasad
  hwhevr @goatbar
  #unhresearchtools
  [11:12]
** Not Python 3

In this class we are using Python 2.7. You will want to avoid reference material for python 3. While python 3 is even better than python 2.x, there is still work to be done to get all of the add ons ready for python 3 and you will have trouble with getting examples to work. To reduce confusion, just avoid python 3 for now. If you learn python 2.7, the switch to python 3 will be very easy and there is even a program to automatically make the number of small changes required for code to work with python 3.

** See Also

If you like the concept of a double sided reference card, here some for python and ipython:

```bash
```

*** hminami (~chatzill@192.168.8.245) has joined channel #unhresearchtools
*** cengler (~chatzill@lab8.ccom.mh) has joined channel #unhresearchtools
```bash
eRC> /names
```

** Users on #unhresearchtools: hminami bwelton kjerram ben n hassan prasad
Gmasetti matt_w mohammad ygh2 sthein berrya mluke schwehrvm @goatbar
There are a number of very good free books to get you started. I've sorted them in the order that I think you might want to approach them.

- **Dive Into Python** by Mark Pilgrim
- **Python Quick Reference** by Richard Gruet
- **Think Python** by Allen Downey
- **Byte of Python** by Swaroop C H. Make sure to get the 2.x version. Not 3!
- **Data Structures and Algorithms with Object-Oriented Design Patterns in Python** by Bruno R. Preiss
- **The Programming Historian** by William J. Turkel and Alan MacEachern
  (for a research area focused on python)
At offset 10:30
At 11:47
Run pwd to make sure you are in ~/class/11
At 13:30
You can run any bash shell command by putting a “!” as the first character of the line.
At 15:05

ipython will put in parentheses if you leave them out
String Form: <built-in function open>
Namespace: Python builtin
Docstring:
open(name[, mode[, buffering]]) -> file object

Open a file using the file() type, returns a file object. This is the preferred way to open a file. See file.__doc__ for further information.

In [14]: open??
Type: built-in function or method
Base Class: <type 'builtin_function_or_method'>
String Form: <built-in function open>
Namespace: Python builtin
Docstring [source file open failed]:
open(name[, mode[, buffering]]) -> file object

Open a file using the file() type, returns a file object. This is the preferred way to open a file. See file.__doc__ for further information.

In [15]:

The main web page for python documentation is: http://docs.python.org/

Inside of python, there are a number of ways to get help.
First, you can directly ask for help. Here we are asking for help on the open "function":

```
#+BEGIN_SRC python
help open
#+END_SRC
```

You can also put a "?" after a bunch of text and it will try to tell you what it can about that string. You can put the "?" before or
In [16]: ?Exit
Type: Magic function
Base Class: <type 'instancemethod'>
String Form: <bound method InteractiveShell.magic_Exit of <IPython.iplib.InteractiveShell object at 0xb76e138c>>
Namespace: IPython internal
File: /usr/lib/python2.7/dist-packages/IPython/Magic.py
Definition: Exit(self, parameter_s='')
Docstring:
   Exit IPython without confirmation.

In [17]: Exit()
NameError
Traceback (most recent call last)
/home/researchtools/class/11/<ipython console> in <module>()
NameError: name 'Exit' is not defined

In [18]: Exit
researchtools@ubuntu:~/class/11$
At 17:01
I used S-<TAB> to collapse the whole document to the top level headings
Here we are faced with a little problem before we go on. I would like the examples to be runnable both in org-mode with `C-c C-c` and as something you can paste into ipython without modification. However, that is not possible. The setup for python in org-babel is that it ignores what we will print. So if I try a print statement in python and run it with org-babel:

```
#+BEGIN_SRC python
print 1
#+END_SRC
```

The results above are "None". Say what?!! It turns out that we have to "return" what we want org-babel to print.

```
#+BEGIN_SRC python
return 1
#+END_SRC
```

That is more like what we wanted. If you just paste the text without the return, all will be well. So, in ipython, it will look like this:

```
ipython
```

```python
Python 2.7.1+ (r271:86832, Apr 11 2011, 18:05:24)
```
In [1]: help open
-----> help(open)

In [2]: print 1
1

In [3]: 'this is a string'
Out[3]: 'this is a string'

In [4]: 'this is a string'.capitalize()
Out[4]: 'This is a string'

In [5]: 'this is a string'.upper()
Out[5]: 'THIS IS A STRING'

I will stick

#*results:[]
: This is a string

Or we can get fancier. The title method for a string makes it have each word capitalized.

#*BEGIN_SRC python
return 'this is a string'.title()
#*END_SRC

#*results:
: This Is A String
In [9]: at 20:15

Out[9]: 'This is a string'

In [5]: 'this is a string'.upper()
Out[5]: 'THIS IS A STRING'

In [6]: 'this is a string'.title()
Out[6]: 'This Is A String'

In [7]: 'this' + 'is' + 'a' + 'string'
Out[7]: 'thisisastring'

In [8]: 'this' + ' is' + ' a' + ' string'
Out[8]: 'this is a string'

In [9]: type ('this is a string')
Out[9]: <type 'str'>

In [10]: type 33
------> type(33)
Out[10]: <type 'int'>

In [11]:

```python
# BEGIN_SRC python
return type('my string')
# END_SRC
```

# results:

* Data types in python
* A little ipython before we go on
* Trying out the data types
* Working with files
* A for loop
* Making a function

: int: float: str: list: ...
: ipython: ...
: str: list: int: float: list: ...
: file: ...
: for: ...
: function: ...

---

Wednesday, October 5, 11

In [9]: at 20:15
There are several basic data types in Python:

- `str` - a character or string
- `int` - integers (aka whole numbers)
- `float` - real numbers
- `complex` - imaginary numbers
- `bool` - Booleans
- `list` - ordered sequence of items that can change
- `tuple` - ordered sequence that *cannot* change
- `dict` - a fast lookup table or "dictionary"
- `file` - you can read and write to files
- `None` - A special case

Note that `str`, `dict` and `file` also act as sequences of items. For example... Jumping ahead and using a for loop before I've explained the concept of a for loop. Sorry!

```python
# BEGIN_SRC python
for c in 'geology':
    print c
# END_SRC
```
* Data types in python

There are several basic data types in python.

- **str** = a character or string --> 'a' 'hello' "world"
  - lists with three quote characters can span multiple lines'
- **numbers**
  - **int** = integers (aka whole numbers) 1, 2, -1, 0
  - **float** = real numbers 3.1415, 0.0, -9e20
  - **complex** = imaginary numbers. complex(1,4)
- **bool** = Booleans. True or False
- **sequences of items**
  - **list** = ordered sequence of items that can change.
    - [1, -3, 1.3, 'hello', ['list', 'inside', 'a', 'list']
  - **tuple** = ordered sequence that can *not* change. (1,-3,'hi')
  - **set** = only one of each item
    - set([1,2,1,1]) -> set([1, 2])
  - **dict** = a fast lookup table or "dictionary"
    - { 1: 2, 99: 'second', 'third': 333 }
  - **file** = you can read and write to files
  - **None** = A special case

Note that **str**, **dict** and **file** also act as sequences of items. For example... Jumping ahead and using a for loop before I've explained the concept of a for loop. Sorry!

```python
#BEGIN_SRC python
for c in 'geology':
    print c
#END_SRC
```
Interactive namespace is empty.

Interactive namespace is empty.

shipname='R/V Cocheco'

shipname

shipname

del shipname

who

who

shipname

shipname

:ipython:

Try out those quit and start

We can also ask ipython to create a log file of our session.

```python
logstart
a = 1+2
b = 3+4
```
At 27:04
%logstart
history
%logstop
researchtools@ubuntu: ~/class/11

18: del shipename
19: __ip.magic("who ")
20: __ip.magic("whos ")
21: __ip.system("ls -F -l")
22: __ip.magic("logstart ")
23: __ip.magic("history ")

In [24]: ls -l
total 100
-rw-r--r-- 1 researchtools researchtools 21262 2011-10-04 10:28 11-ipython-matplotlib
-rw-r--r-- 1 researchtools researchtools 21269 2011-10-04 11:43 #11-ipython.org
-rw-r--r-- 1 researchtools researchtools 21269 2011-10-04 10:34 #11-ipython.org
-rw-r--r-- 1 researchtools researchtools 21269 2011-10-04 10:34 #11-ipython.org
-rw-r--r-- 1 researchtools researchtools 820 2011-10-04 11:47 ipython_log.py

In [25]: logst
%logstart %logstate %logstop

In [25]: %logstop

In [26]: logstop

logstart
a = 1+2
b = 3+4
who
logstop
less ipython_log.py
#+END_SRC

For the logging commands, type "%log" in ipython and then press <TAB>

#+BEGIN_EXAMPLE
In [1]: %log
%logoff %logon %logstart %logstate %logstop

---**- 11-ipython.org 70% L480 (Org) ---**
In [28]: shipname='Coastal Surveyor'

In [29]: len(shipname)
   ---> len(shipname)
Out[29]: 16

In [30]: shipname[0]
Out[30]: 'C'

In [31]: shipname[1]
Out[31]: 'o'

In [32]: shipname[5:8]
Out[32]: 'al '

In [33]: shipname.find('S')
Out[33]: 8

In [34]: shipname[8:]
Out[34]: 'Surveyor'

** str - strings

```python
#BEGIN_SRC python
shipname='Coastal Surveyor'
len(shipname)
shipname[0] # Count from zero
shipname[5:8]
shipname.find('S') # returns 8
shipname.find('x') # returns -1 ... not found
shipname[8:] # from position 8 to the end
shipname[-4:] # last 4 characters
#END_SRC
```
Out[40]: 1.1

In [41]: type 1.1
    ---> type(1.1)
Out[41]: <type 'float'>

In [42]: float('3.1415')
Out[42]: 3.1415

In [43]: type float('3.1415')
    ---> type(float('3.1415'))
Out[43]: <type 'float'>

In [44]: float('hi mom')
ValueError
Traceback (most recent call last)
/home/researchtools/class/11/ipython console in <module>()
ValueError: could not convert string to float: hi mom

In [45]: import math

type(1)
1.1
type(1.1)
str(1.1)
float('3.1415')
import math
math.pi
math.sin(math.pi/2)
math.radians(180)
math.degrees(2*math.pi)
math. # then press the <TAB> key to get a list
complex(1,4j)

---* 11-ipython.org  75% L539  (Org) ---
ValueError Traceback (most recent call last)
/home/researchtools/class/11/<ipython console> in <module>()

ValueError: could not convert string to float: hi mom

In [45]: import math
In [46]: math.pi
Out[46]: 3.141592653589793
In [47]: math.sin(math.pi/2)
Out[47]: 1.0
In [48]: math.radians(180)
Out[48]: 3.141592653589793
In [49]: math.radians(180*2)
Out[49]: 6.283185307179586
In [50]: math.de
In [47]: math.sin(math.pi/2)
Out[47]: 1.0

In [48]: math.radians(180)
Out[48]: 3.141592653589793

In [49]: math.radians(180*2)
Out[49]: 6.283185307179586

In [50]: math.degrees(2*math.pi)
AttributeError
Traceback (most recent call last)
/home/researchtools/class/11/<ipython console> in <module>(0)
AttributeError: 'module' object has no attribute 'degress'
In [51]: math.degrees(2*math.pi)
Out[51]: 360.0

In [52]: type(1)
1.1
type(1.1)
str(1.1)
float('3.1415')
import math
math.pi
math.sin(math.pi/2)
math.radians(180)
math.radians(2*math.pi)
math.degrees(2*math.pi)
math. # then press the <TAB> key to get a list
complex(1,4j)

---**- 11-ipython.org 75% L539 (0rg)---**
import math
math.pi
math.sin(math.pi/2)
math.radians(180)
math.degrees(2*math.pi)
math. # then press the <TAB> key to get a list

complex(1,4j)
In [52]: math.pow?
Type:       builtin_function_or_method
Base Class: <type 'builtin_function_or_method'>
String Form: <built-in function pow>
Namespace:  Interactive

Docstring:

    pow(x, y)

    Return x**y (x to the power of y).

In [53]: complex(1,4j)
Out[53]: (-3+0j)
In [54]:

* Making a function
* Checking your code with pylint
In [52]: math.pow?

Type:           builtin
Base Class:     <type 'function'>
String Form:    <built-in function pow>
Namespace:      Interactive
Docstring:
   pow(x, y)
   
   Return x**y (x to the y power).

In [53]: complex(1,4j)
Out[53]: (-3+0j)

In [54]: 4j * (2+9j)
Out[54]: (-36+8j)

In [55]: 4j + (2+9j)
Out[55]: (2+13j)

In [56]:

shipname.find('S') # returns 8
shipname.find('x') # returns -1 ... not found
shipname[8:] # from position 8 to the end
shipname[-4:] # last 4 characters

```python
# BEGIN_SRC python
1
type(1)
1.1
type(1.1)
str(1.1)
float('3.1415')
import math
math.pi
math.sin(math.pi/2)
math.radians(180)
math.degrees(2*math.pi)
math. # then press the <TAB> key to get a list

complex(1,4j)
4j * (2 + 9j)

# END_SRC
```

** list of items ...
** Basic operations on strings...
  * Working with files
  * A for loop
  * Making a function
  * Checking your code with pylint

---

11-ipython.org  Bot L546  (Org)
Lists start at 34:40 with range
At 36:00 is the ship list
37:10 for tuples
To modify a fixed tuple, the only thing you can do is overwrite it with a new and better tuple:

```python
mytuple = (1, 3, 5)
mytuple = mytuple + (4, 1, 2)
```
In [67]: ship.append('passenger')

In [68]: ship
Out[68]: ['tug', 'row boat', 'r/v', 'passenger']

In [69]: ship.sort()

In [70]: ship
Out[70]: ['passenger', 'r/v', 'row boat', 'tug']

In [71]: ship.sort?
Type:               builtin function or method
Base Class:         <type 'builtin_function_or_method'>
String Form:        <built-in method sort of list object at 0x95d89ec>
Namespace:          Interactive
Docstring:
L.sort(cmp=None, key=None, reverse=False) -- stable sort *IN PLACE*;
cmp(x, y) -> -1, 0, 1

In [72]:

** =list= of items

#+BEGIN_SRC python
range(4)
range(3,7)
range(3,28,5)

ships = [ 'tug','row boat', 303902000, 123456789 ]
type(ships)
ships.append(369970120)
ships.sort()
ships[0]

---

11-ipython.org 76% L558 (Org)
Asking ship.sort() if it can sort any other way. Yes, but writing a cmp (compare) function is a slightly advanced topic.
In [75]: ship.remove('tug')

In [76]: ship
Out[76]: ['passenger', 'r/v', 'row boat']

In [77]: ship.pop()
Out[77]: 'row boat'

In [78]:

range(3,7)
range(3,28,5)

ships = [ 'tug', 'row boat', 303902000, 123456789 ]
type(ships)
ships.append(369970120)
ships.sort()
ships[0]
ships[-1]
ships.remove('row boat')
ships. # press <TAB>

#+END_SRC
ship._gt__ship._reversed__ship.reverse
ship._hash__ship._rmul__ship.sort

In [75]: ship.remove('tug')

In [76]: ship
Out[76]: ['passenger', 'r/v', 'row boat']

In [77]: ship.pop()
Out[77]: 'row boat'

In [78]: ship.pop?
Type:           builtin function or method
Base Class:     <type 'builtin_function_or_method'>
String Form:    <built-in method pop of list object at 0x95d89ec>
Namespace:      Interactive
Docstring:
L.pop([index]) -> item -- remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

In [79]:

range(3,7)
range(3,28,5)

ships = [ 'tug', 'row boat', 303902000, 123456789 ]
type(ships)
ships.append(369970120)
ships.sort()
ships[0]
ships[-1]
ships.remove('row boat')
ships. # press <TAB>
#+END_SRC
String Form: <built-in method pop of list object at 0x95d89ec>
Namespace: Interactive
Docstring:
L.pop([index]) -> item -- remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

In [79]: ship?
Type: list
Base Class: <type 'list'>
String Form: ['passenger', 'r/v']
Namespace: Interactive
Length: 2
Docstring:
    list() -> new empty list
    list(iterable) -> new list initialized from iterable's items

In [80]: ships?
Object 'ships' not found.

In [81]:
Object 'ships' not found.

In [81]: who
math    ship    shipname    tmp

In [82]: whos
Variable   Type    Data/Info
---        ---    --------
math        module <module 'math' (built-in)>   
ship        list   ['passenger', 'r/v']
shipname    str     Coastal Surveyor  
tmp         tuple   ('tug', 'row boat', 'r/v')

In [83]: numbers = '1 2 3 4 99'
In [84]: numbers = '1 2 3 4 99 ten'
In [85]: type numbers
-----> type(numbers)
Out[85]: <type 'str'>

#BEGIN_SRC python
shipname='Gulf Challenger, R/V'
shipname.split()
fields = shipname.split(', ')
len(fields)
name = fields[0]
name * 4
'-- '.join(fields)
#END_SRC

* Working with files
* A for loop
* Making a function

--- 11-ipython.org  78% L565 (Org)
SUBTREE (NO CHILDREN)
numbers.formatter_field_name_split
numbers.formatter_parser
numbers.capitalize
numbers.center
numbers.count

In [86]: numbers.split

Type:         built-in function or method
Base Class: <type 'builtin_function_or_method'>
String Form: <built-in method split of str object at 0x95da2a0>
Namespace:   Interactive

Docstring:

S.split([sep [,maxsplit]]) -> list of strings

Return a list of the words in the string S, using sep as the
delimiter string. If maxsplit is given, at most maxsplit
splits are done. If sep is not specified or is None, any
whitespace string is a separator and empty strings are removed
from the result.

In [87]:

#+BEGIN_SRC python

shipname='Gulf Challenger, R/V'
shipname.split()
fields = shipname.split(',',)
len(fields)
name = fields[0]
nome * 4
' -- '.join(fields)

#+END_SRC

* Working with files
* A for loop
* Making a function

---**-- 11-ipython.org  78% L565 (Org) ---**--

SUBTREE (NO CHILDREN)
In [88]: numbers
Out[88]: '1 2 3 4 99 ten'

In [89]: shipname = 'Gulf Challenger, R/V'

In [90]: shipname.split()
Out[90]: ['Gulf', 'Challenger,', 'R/V']

In [91]: shipname.split(',',)
Out[91]: ['Gulf Challenger', ' R/V']

In [92]: shipname.split(',',)[0]
Out[92]: 'Gulf Challenger'

In [93]: fields = ship
    ship
    shipname

In [93]: fields = shipname.split
    shipname.splitlines
In [93]: fields = shipname.split
Number 93 at 45:40 talking about the “.spli” with tab for completions and what happens when there is more than one option to complete to.
```python
In [93]: fields = shipname.split
   ...:     shipname.splitlines

In [93]: fields = shipname.split(',,')

In [94]: fields
Out[94]: ['Gulf Challenger', ' R/V']

In [95]: type(fields)
Out[95]: <type 'list'>

In [96]: name = fields[0]

In [97]: name
Out[97]: 'Gulf Challenger'

In [98]: name * 4
Out[98]: 'Gulf ChallengerGulf ChallengerGulf ChallengerGulf Challenger'
```

---

`A for loop`
`Making a function`

---

11-ipython.org  78% L576 (Org)
In [169]:

In [170]:

In [171]:

In [172]:

In [173]:

In [174]:

In [175]:

In [176]: name * 4

In [177]: ' '*80
Out[177]:

In [178]:

```python
# BEGIN_SRC python
shipname='Gulf Challenger, R/V'
shipname.split()
fields = shipname.split(',',)
len(fields)
name = fields[0]
name * 4
' ' .join(fields)
# END_SRC

* Working with files
* A for loop
* Making a function
```
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Data/Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>fields</td>
<td>list</td>
<td>['Gulf Challenger', 'R/V']</td>
</tr>
<tr>
<td>math</td>
<td>module</td>
<td>&lt;module 'math' (built-in)&gt;</td>
</tr>
<tr>
<td>name</td>
<td>str</td>
<td>Gulf Challenger</td>
</tr>
<tr>
<td>numbers</td>
<td>str</td>
<td>1 2 3 4 99 ten</td>
</tr>
<tr>
<td>ship</td>
<td>list</td>
<td>['passenger', 'R/V']</td>
</tr>
<tr>
<td>shipname</td>
<td>str</td>
<td>Gulf Challenger, R/V</td>
</tr>
<tr>
<td>tmp</td>
<td>tuple</td>
<td>('tug', 'row boat', 'R/V')</td>
</tr>
</tbody>
</table>

In [180]: ' -- '.join?
Object ' -- '.join' not found.

In [181]: ' -- '.join(tmp)
Out[181]: 'tug -- row boat -- R/V'

In [182]: ', '.join(tmp)
Out[182]: 'tug, row boat, R/V'

In [183]: my_csv = ', '.join(tmp)

Open up the file in emacs: ~/class/11/data.csv

We can now read that data from python!

#BEGIN_SRC python
datafile = open('data.csv')
type( datafile )
datafile.readline()
datafile.readline()
#END_SRC

out.write('9,-1\n')
out.close()

---** 11-ipython.org  80% L585  (Org)---
Here is a Python code snippet to create a file:

```python
# BEGIN_SRC python
out = open('data.csv', 'w')
out.write('1,2
')
out.close()
# END_SRC
```

Open up the file in Emacs: `~/class/11/data.csv`

We can now read that data from Python!
In [192]: out.write('1,2\n')

In [193]: out.write('4,5\n')

In [194]: out.write('9,-1\n')

In [195]: ls -l
total 104
-rw-r--r-- 1 researchtools researchtools 21262 2011-10-04 10:28 11-ipython-matplotlib.org
-rw-r--r-- 1 researchtools researchtools 21269 2011-10-04 12:05 #11-ipython.org#
-rw-r--r-- 1 researchtools researchtools 21269 2011-10-04 10:34 11-ipython.org
-rw-r--r-- 1 researchtools researchtools 21269 2011-10-04 10:34 11-ipython.org
-rw-r--r-- 1 researchtools researchtools 21269 2011-10-04 10:34 11-ipython.org
-rw-r--r-- 1 researchtools researchtools 0 2011-10-04 12:18 data.csv
-rw-r--r-- 1 researchtools researchtools 2264 2011-10-04 12:21 ipython_log.py
-rw-r--r-- 1 researchtools researchtools 842 2011-10-04 11:48 ipython_log.py.001#

In [196]: out.close()

In [197]: ls -l data.csv
-rw-r--r-- 1 researchtools researchtools 2011-10-04 12:22 data.csv

In [198]:

```
#BEGIN_SRC python
out = open('data.csv','w')
out.write('1,2\n')
out.write('4,5\n')
out.write('9,-1\n')
out.close()
#END_SRC

Open up the file in emacs: ~/class/11/data.csv
We can now read that data from python!
```

51:15 write
53:10 out.close()
In [192]: out.write('1,2

In [193]: out.write('4,5

In [194]: out.write('9,-1

In [195]: ls -l
total 104
-rw-r--r-- 1 researchtools researchtools  0 Oct 1 12:27 data.csv
-rw-r--r-- 1 researchtools researchtools  0 Oct 1 12:27 .
-rw-r--r-- 1 researchtools researchtools 104 Oct 1 12:27 ..
-rw-r--r-- 1 researchtools researchtools  0 Oct 1 12:27 __init__.py
-rw-r--r-- 1 researchtools researchtools  0 Oct 1 12:27 __init__.pyc
-rw-r--r-- 1 researchtools researchtools 104 Oct 1 12:27 __init__.pyw
-rw-r--r-- 1 researchtools researchtools  0 Oct 1 12:27 __init__.pyo

In [196]: out.close()

In [197]: ls -l data.csv
-rw-r--r-- 1 researchtools researchtools  0 Oct 1 12:27 data.csv

... data.csv
All L1 (CSV yas)--F1--

In [198]:

type( datafile )
datafile.readline()
datafile.readline()
datafile.readline()
datafile.readline()
del(datafile)
datafile = open('data.csv')
lines = datafile.readlines()
len(lines)
lines[0]
lines[0].strip()
lines[0].strip().split(', ') # yikes! you can chain things together
#* END_SRC

* A for loop

...
In [198]: datafile = open('data.csv')

In [199]: type(datafile)
   ---> type(datafile)
Out[199]: <type 'file'>

In [200]: datafile.read
   datafile.readinto datafile.readline datafile.readlines

In [200]: datafile.readline
   datafile.readline datafile.readlines

In [200]: datafile.readlines()
Out[200]: ['1,2
', '4,5
', '9,-1
']

In [201]: datafile = open('data.csv')

In [202]: datafile.readline()
Out[202]: '1,2'

In [203]:

```python
# BEGIN_SRC python
datafile = open('data.csv')

print type(datafile)
print datafile.readline()
print datafile.readline()
print datafile.readline()
print datafile.readline()
print del(datafile)

datafile = open('data.csv')
lines = datafile.readlines()
len(lines)
lines[0]
```

```bash
** 11-ipython.org 83% L605 (Org)
```
```python
# BEGIN_SRC python
datafile = open('data.csv')
type(datafile)
datafile.readline()
datafile.readline()
datafile.readline()
datafile.readline()
del(datafile)

lines = datafile.readlines()
len(lines)
lines[0]
```

```
# END_SRC
```