**Python**

**Running programs in ipython. Example:**Create a file my\_file.py in a text editor. In the file, add the following lines:  
string = ‘Hello world‘

print(s)

Start the Ipython shell (an enhanced interactive Python shell):

• by typing “ipython” from a Linux/Mac terminal, or from the Windows cmd shell,

• or by starting the program from a menu, e.g. in the Python(x,y) or EPD menu if you have installed one of these scientific-Python suites.

Get current directory:

**In [17]:** os.getcwd()

List a directory:

**In [31]:** os.listdir(os.curdir)

Make a directory:

**In [32]:** os.mkdir(’junkdir’)

Rename the directory:

**In [36]:** os.rename(’junkdir’, ’foodir’)

REMOVE A DIRECTORY!!!!!!!!:

**In [41]:** os.rmdir(’foodir’)

Now, you can run it in ipython and explore the resulting variables:  
**In [3]:** %run my\_file.py 🡨 cd ‘address’ to change drive

Hello word

**In [4]:** string

Out[4]: ’Hello word’

**In [5]:** %whos

Variable Type Data/Info

----------------------------

string str Hello word

**Once** you have started the interpreter, type

**>>> print** "Hello, world!"

Hello, world!

The message “Hello, world!” is then displayed. You just executed your first Python instruction, congratulations!

To get yourself started, type the following stack of instructions

**>>>** 1 + 1

2

**>>>** a = 3

**>>>** b = 2\*a

**>>>** type(b)

<type ’int’>

**>>> print** b

6

**>>>** a\*b

18

**>>>** b = ’hello’

**>>>** type(b)

<type ’str’>

**>>>** b + b

’hellohello’

**>>>** 2\*b

’hellohello’  
**>>>** ’An integer: %i; a float: %f; another string: %s’ % (1, 0.1, ’striiiing’)

’An integer: 1; a float: 0.100000; another string: striiiing’

**>>>** a = "hello, world!"

**>>>** a[3:6] # 3rd to 6th (excluded) elements: elements 3, 4, 5

’lo,’

**>>>** a[2:10:2] # Syntax: a[start:stop:step]

’lo o’

**>>>** a[::3] # every three characters, from beginning to end

’hl r!’

**>>>** l = [1, 2, 3, 4, 5]

**>>>**r = l[::-1]

**>>>** r

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

**>>>**a.replace(’l’, ’z’)

’hezzo, worzd!’

**>>>**a.replace(’l’, ’z’,2)

’hezzo, world!’



In addition, the type of a variable may change, in the sense that at one point in time it can be equal to a value of

a certain type, and a second point in time, it can be equal to a value of a different type. b was first equal to an

integer, but it became equal to a string when it was assigned the value ’hello’

Floats and imaginary:

**>>>** c = 2.1

complex (a native type in Python!)

**>>>** a = 1.5 + 0.5j  
**>>>** a.real  
1.5  
**>>>** a.imag  
0.5

**>>>** type(1. + 0j )

<type ’complex’>

and Booleans:

**>>>** 3 > 4

False

**>>>** test = (3 > 4)

**>>>** test

False

**Warning**: Integer division

**>>>** 3 / 2

1

Trick: use floats:

**>>>** 3 / 2.

1.5

**>>>** a = 3

**>>>** b = 2

**>>>** a / b

1

**>>>** a / float(b)

1.5

Special forms of parameters:

• \*args: any number of positional arguments packed into a tuple

• \*\*kwargs: any number of keyword arguments packed into a dictionary

**In [35]: def** variable\_args(\*args, \*\*kwargs):

**....: print** ’args is’, args

**....: print** ’kwargs is’, kwargs

**....:**

**In [36]:** variable\_args(’one’, ’two’, x=1, y=2, z=3)

args is (’one’, ’two’)

kwargs is {’y’: 2, ’x’: 1, ’z’: 3}

r opens the file for reading only

w truncates the file, overwriting whatever was already there

a appends to the file, adding onto whatever was already there

w+ opens for reading and writing, truncating the file but also allowing you to read back what's been written to the file

a+ opens for appending and reading, allowing you both to append to the file and also read its contents

**ADDRESS MANIPULATION**

**In [72]:** a = os.path.abspath(’junk.txt’)

**In [73]:** a

Out[73]: ’/Users/cburns/src/scipy2009/scipy\_2009\_tutorial/source/junk.txt’

**In [74]:** os.path.split(a)

Out[74]: (’/Users/cburns/src/scipy2009/scipy\_2009\_tutorial/source’,

’junk.txt’)

**In [78]:** os.path.dirname(a)

Out[78]: ’/Users/cburns/src/scipy2009/scipy\_2009\_tutorial/source’

**In [79]:** os.path.basename(a)

Out[79]: ’junk.txt’

**In [80]:** os.path.splitext(os.path.basename(a))

Out[80]: (’junk’, ’.txt’)

**In [88]:** os.path.expanduser(’~/local’)

Out[88]: ’/Users/cburns/local’

**In [92]:** os.path.join(os.path.expanduser(’~’), ’local’, ’bin’)

Out[92]: ’/Users/cburns/local/bin’  
**In [11]:** os.environ.keys()

**OOP:**

**class MasterStudent**(Student): 🡨masterstudent inherits from student

def function:

fkhbgdsjug;

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

From a script to functions

• A script is not reusable, functions are.

• Thinking in terms of functions helps breaking the problem in small blocks.

**print?**print value, ..., sep=’ ’, end=’\n’, file=sys.stdout

Long lines: you should not write very long lines that span over more than (e.g.) 80 characters. Long lines can be

broken with the \ character

**>>>** long\_line = "Here is a very very long line **\**

**...** that we break in two parts."

Numpy:

**>>>** image = np.random.rand(30, 30)

**>>>** plt.imshow(image)

**>>>** plt.gray()

**>>>** plt.show()

**>>>** plt.pcolor(image)

**>>>** plt.hot()

**>>>** plt.colorbar()

**>>>** plt.show()