Introduction to High Performance Scientific Computing

Autumn, 2017

Lecture 3

Imperial College
London

Prasun Ray

12 October, 2017

Basic data input/output

To open a file for reading or writing:

```
outfile = open("output.txt","w") #open file with flag "w" for writing

infile = open("input.txt","r") #open file with flag "r" for reading
```

Basic data input/output: write to file

Write two numbers to file, *output.txt*:

```
outfile = open("output.txt","w") #open file with flag "w" for writing

outfile.write("This is a header for output.txt \n \n") #header
```

Basic data input/output: write to file

Write two numbers to file, *output.txt*:

```
outfile = open("output.txt","w") #open file with flag "w" for writing

outfile.write("This is a header for output.txt \n \n") #header

x = 2.0 #some data to be written
y = 1.0

outfile.write("x = %6.3f, y = %6.3f \n" %(x,y)) #write data

outfile.close() #close file
```

Basic data input/output: write to file

Write two numbers to file, *output.txt*:

```
outfile = open("output.txt","w") #open file with flag "w" for writing

outfile.write("This is a header for output.txt \n \n") #header

x = 2.0 #some data to be written
y = 1.0

outfile.write("x = %6.3f, y = %6.3f \n" %(x,y)) #write data

outfile.close() #close file
```

If we run this code and look at *output.txt*:

```
$ cat output.txt

This is a header for output.txt

x =  2.000, y =  1.000
```

Basic data input/output: read from file

Read from file, *input.txt*, separate out numbers

```
infile = open("input.txt","r") #open file with flag "r" for reading

temp = infile.readline() #first read header
temp = infile.readline()

#loop through lines in file
for line in infile:
    print line
```

Output:

```
 x =  2.000, y =  1.000

 x =  3.000, y =  10.000

 x =  4.000, y =  100.000

 x =  5.000, y =  1000.000
```

Basic data input/output: read from file

Read from file, *input.txt*, separate out numbers

```
infile = open("input.txt","r") #open file with flag "r" for reading

temp = infile.readline() #first read header
temp = infile.readline()

#loop through lines in file
for line in infile:
    print line

words = line.split() #separate last line into words
print "words=",words
```

Output: words= ['x', '=', '5.000,', 'y', '=', '1000.000']

Basic data input/output: read from file

Use *line.split* to break line into words

```
infile = open("input.txt","r") #open file with flag "r" for reading

temp = infile.readline() #first read header
temp = infile.readline()

#loop through lines in file
for line in infile:
    print line

words = line.split() #separate last line into words
print "words=",words
```

Output: words= ['x', '=', '5.000,', 'y', '=', '1000.000']

Basic data input/output: read from file

And then the last word is y

```
infile = open("input.txt","r") #open file with flag "r" for reading

temp = infile.readline() #first read header
temp = infile.readline()

#loop through lines in file
for line in infile:
    print line

words = line.split() #separate last line into words
print "words=",words

y = float(words[-1]) #pick out y, convert from string to float
print "y=",y
```

Output: y= 1000.0

Functions

Basic idea: input → function → output

```
def function_name(input1,input2,inputN):
            #Code with operations involving input variables
            #that assigns values to output variables

            return output1,output2,outputM
```

- **Again: extent of function "block" set by colon and indentation**

Functions: an example

Add three numbers:

```
In [5]: def sum3(x,y,z):
   ...:     #return sum of three numbers, x,y,z
   ...:     return x+y+z
   ...:

In [6]: sum3(1,2,3)
Out[6]: 6
```

- **Function name is sum3 and can be called from command line**
- **Typically include functions in scripts and *import* them into command line (or other scripts)**

Functions: a few details

```
def example(x,y,z):
    '''Example of a python function,
    returns twice the first input variable
    and the product of the 2nd and 3rd input
    variables'''

    x2 = 2*x

    return x2,y*z

In [45]: from function_example import example

In [46]: example(1,2,3)
Out[46]: (2, 6)
```

- **Here, we have *imported* the function into the terminal and called it with input 1,2,3 generating output 2,6**
- ***x2*** **is a *local* variable and cannot be accessed from the terminal…**

Functions: a few details

```
In [45]: from function_example import example

In [46]: example(1,2,3)
Out[46]: (2, 6)

In [47]: x2
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-47-e2ee9ad17fdf> in <module>()
----> 1 x2

NameError: name 'x2' is not defined
```

- **Here, we have *imported* the function into the terminal and called it with input 1,2,3 generating output 2,6**
- ***x2*** **is a *local* variable and cannot be accessed from the terminal…**

Functions: a few details

- **Be careful when sending a mutable object (*e.g.* a list) into a function: it can change!**

```
def example2(x,y,z):
    '''Another example of a python function which
    returns twice the first input variable
    and the product of the 2nd and 3rd input
    variables, but now we assume that x is a list and
    only double its 1st element.'''

    x[0] = x[0]+1

    return x,y*z

In [98]: a=[1,2,3]

In [99]: example2(a,2,3)
Out[99]: ([2, 2, 3], 6)

In [100]: a
Out[100]: [2, 2, 3]
```

Functions: keyword arguments

- **Can easily set default values for optional input arguments**

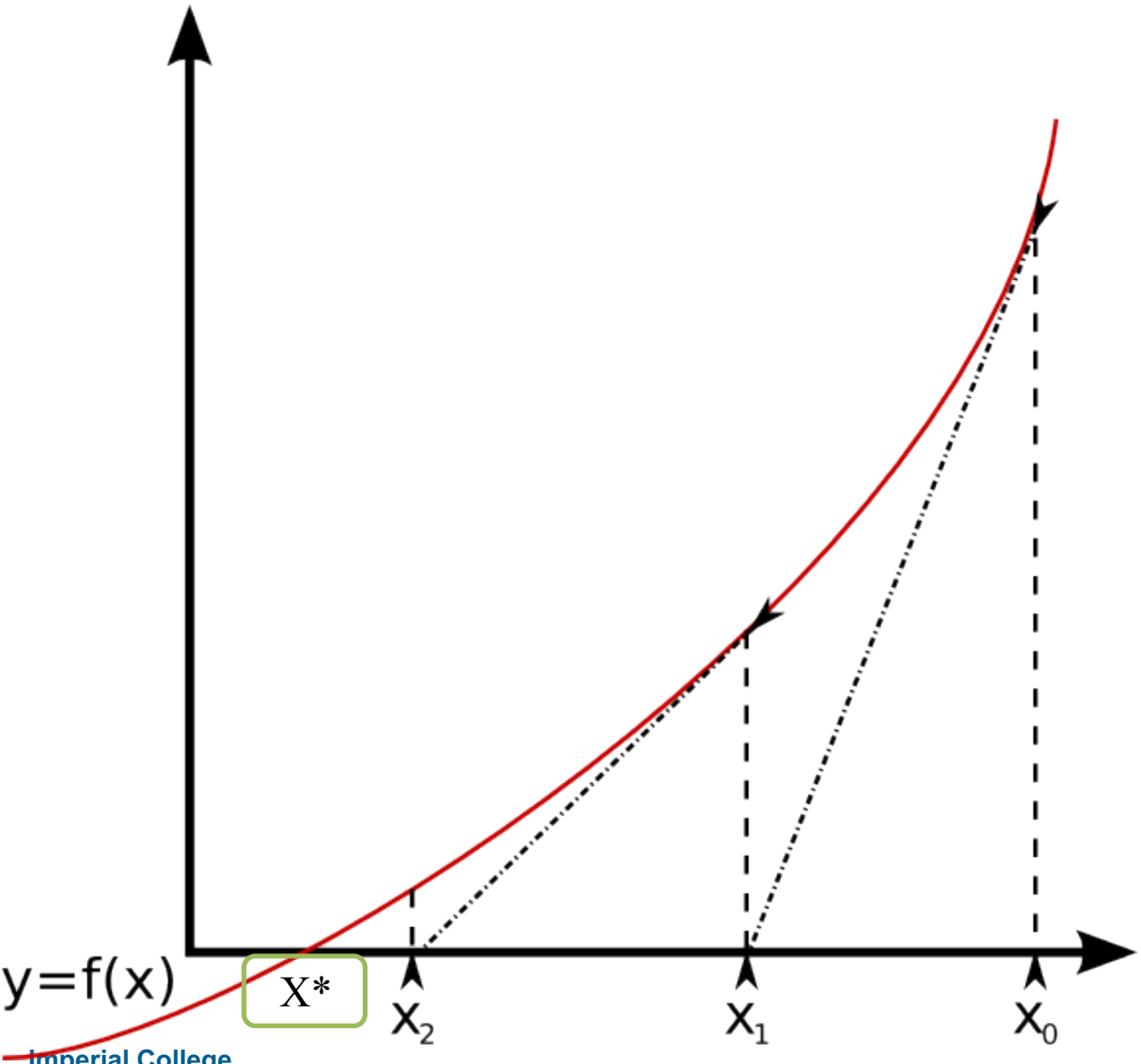
```
def example3(x,y,z=1):
    '''Example of a python function,
    returns twice the first input variable
    and the product of the 2nd and 3rd input
    variables, and z has a default value of 1'''

    return 2*x,y*z

In [105]: example3(1,2,3)
Out[105]: (2, 6)

In [106]: example3(1,2)
Out[106]: (2, 2)
```

Demo: computing sqrt with Newton's method

Newton's method: solve $f(x)=0$

1. **guess solution** x_1
2. **compute** $f(x_1)$
3. **Is** $f(x_1)$ **sufficiently close to zero?**
4. **If not, compute** df/dx **and use Newton's formula to generate new guess,** x_2
5. **Repeat steps 2-4**

Demo: computing sqrt with Newton's method



Want to find x*:

1. **Guess x0**
2. **Estimate x* based on tangent at x0**
3. **Repeat**

Demo: computing sqrt with Newton's method

We want to solve: $x = \sqrt{a}$

Or: $x^2 - a = 0$

with $\frac{df}{dx} = 2x$

Demo: computing sqrt with Newton's method

We want to solve: $x = \sqrt{a}$

Or: $x^2 - a = 0$

with $\frac{df}{dx} = 2x$

General Newton's method: $x_1 = -f_0 / \frac{df}{dx}|_{x_0} + x_0$

Here, x_0 **is the initial guess**

Demo: computing sqrt with Newton's method

We want to solve: $x = \sqrt{a}$

Or: $x^2 - a = 0$

with $\frac{df}{dx} = 2x$

General Newton's method: $x_1 = -f_0 / \frac{df}{dx}|_{x_0} + x_0$

Here, x_0 **is the initial guess**

For our function, Newton's method becomes:

$$x_1 = \frac{a}{2x_0} + \frac{x_0}{2}$$

Let's code this!

Demo: computing sqrt with Newton's method

We want to solve: $x = \sqrt{a}$

Or: $x^2 - a = 0$

with $\frac{df}{dx} = 2x$

General Newton's method: $x_1 = -f_0 / \frac{df}{dx}|_{x_0} + x_0$

Here, x_0 **is the initial guess**

For our function, Newton's method becomes:

$$x_1 = \frac{a}{2x_0} + \frac{x_0}{2}$$

Let's code this!: ***see mysqrt.py for details***