# Computing for mathematics handout 4 - Functions, Sorting and Searching algorithms and what to expect on the class test.

Lecturer: Vince Knight

Office: M1.30

email: knightva@cf.ac.uk

**Office hours: Thursday 1300-1500**

## What you have learnt this week:

* How to create an unsorted list;
* Some sorting algorithms;
* Some searching algorithms.

## Functions

Some of us are still having problems understanding important concepts of functions:

* Defining functions;
* Using functions.

In *mathematics* we'll often see things like this:

Let us define , and .

Calculate , and for taking on the following values: .

We would simply calculate this by hand as follows:

Our question could then go on to ask:

Now define . Which function ( or ) has the lowest mean value over the above list of values?

Here we would calculate for the above function:

From here we would write: let be the mean values of respectively.

The way we have *defined* functions and passed values to them is exactly the same as we would do it in Python:

def f(x):  
 """  
 return f(x) from example  
 """  
 return x \*\* 2  
  
def g(x):  
 """  
 return g(x) from example  
 """  
 return x \*\* 3 - 1  
  
def h(x):  
 """  
 return h(x) from example  
 """  
 return x \*\* (1/float(2))  
  
values = [0, .5, 1, 3] # This defines our list of values  
for x in values: # We now loop over our values  
 print "When x=%s" %x  
 print "f(x)=%s" % f(x) # Here we calculate f(x) for a given x  
 print "g(x)=%s" % g(x) # Here we calculate g(x) for a given x  
 print "h(x)=%s" % h(x) # Here we calculate h(x) for a given x

Now to do the second half of the problem:

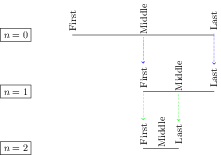
def k(x):  
 """  
 Using our previously defined functions to define k(x)  
 """  
 return f(g(h(x)))  
  
print [k(x) for x in values] # This just return k(x) for x in values using list comprehensions (this is another way of what we did above)  
  
def mean(lst):  
 """  
 A function to return the mean of a list  
  
 Arguments: lst: a list  
  
 Outputs: The mean of all elements in lst  
 """  
 return sum(lst) / len(lst)  
  
print "f: %s" % mean([f(x) for x in values])  
print "g: %s" % mean([g(x) for x in values])  
print "h: %s" % mean([h(x) for x in values])  
print "k: %s" % mean([k(x) for x in values])

## Sorting algorithms

* We saw two algorithms for sorting scrambled lists:
  + Selection sort
  + Bubble sort (this was not a tickable)

## Searching algorithms

* We saw binary search this week which relies on constantly 'dividing' a list in to two smaller lists:



## The class test (8th of November)

* 50 minute class test.
* The class test will have 3 questions:
  + Q1 will be a **non tickable** question from the lab sheets;
  + Q2 will be a question of comparable difficulty to what has been done in the lab sheets;
  + Q3 will be a harder question similar to a question from the lab sheets.
* This will be an open 'book' test. You will have access to the internet, **BUT NOT TO EMAIL** or any other communication with other students: make sure your scripts from the sheets are on your machine: 'I was just logging in to my email to get my sheets' will not be tolerated.
* You will write 3 scripts (1 for each question) and will be uploading and emailing them.

## What you should do next:

* Start revising for the class test: work through all your lab sheets. If you can do exercises in the lab sheets (not just 'understand them' but actually 'do them') you will be fine.
* **Start the next sheet**, there is quite a lot to learn and Object Orientated Programming is a very powerful tool so make sure you make progress outside of the labs.
* To make the best use of the lab sessions turn up having finished your sheets;
* If anything is still unclear **please** come and see me during office hours.