Autumn Quarter 2005
Math. Methods Problem Set 1: Elementary Python

October 2, 2005

1 Working with lists

Create a list of the first 20 powers of 3, without using a multiline `for` loop. Append the next power of 3 to the list. Insert the value $\frac{1}{3}$ at the beginning of the list. Reverse the order of the list. Find the index of the value 81. Print out the 11th item in the list. Using a `for` loop, pop 5 values off the end of the list.

2 Functions and conditionals

Write a function `alpha(T)` which returns the following:

$$\alpha(T) = \begin{cases} \alpha_i & \text{for } T \leq T_i, \\ \alpha_o + (\alpha_i - \alpha_o)\frac{(T-T_o)^2}{(T_i-T_o)^2} & \text{for } T_i < T < T_o \\ \alpha_o & \text{for } T \geq T_o \end{cases} \tag{1}$$

Besides the argument T, this function requires the specification of various parameters, namely T_o, T_i, α_o and α_i. These could be treated as additional arguments to the function, but that makes the use of the function a little complicated if the parameters aren't being changed too often, since you need to remember a long string of arguments and what they are, even if you're not changing them much. Instead, you can just leave the parameters undefined when you write the function, and then just define them any time before you use the function. For example, after you have defined the function you can do:

```
To = 200.
#Put rest of your definitions here
...

alpha(300.)
alpha(270.)
```

and it will work. There are better ways to handle this situation, but the use of globals is simple and effective for lots of everyday use in scripts that don't get too complicated.

After you have defined your function, set $\alpha_i = .1$, $\alpha_o = .9$, $T_i = 260.$ and $T_o = 290.$ and print out a table of T and $\alpha(T)$ for some useful range of T. Check the values to make sure the behavior is reasonable.

3 Loops and functions

Write a function `f(n)` which returns the value

$$f(n) = \sum_{j=1}^{n} \frac{1}{j} \tag{2}$$

Compare the value of $f(j)$ to the function $\ln(j)$ over a range of j extending to very large values. How does $f(j) - \ln(j)$ behave?

4 Approximate derivative of a function

If f is a function of x, then the derivative can be approximated by

$$f'(x) \approx \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon} \tag{3}$$

where ϵ is a suitably small number. Write a function `deriv(x,f,eps)` which takes a number `x`, an arbitrary function `f` and a number `eps` as its arguments and returns an approximation to the derivative df/dx evaluated at x. Try out your function for $f(x) = \sin(x)$, and see how well the actual derivative is approximated for various values of ϵ

5 Elementary use of objects

Define a class `SpaceShip`, which creates an object containing the following data for the spaceship: the positions `x` and `y` in a two dimensional space, the velocities `u` and `v` in the two directions, the current time `t` and a time increment `dt`. In addition, the class should have two methods:

- `move(...)` which advances the spaceship by updating the position according to `x = x + u*dt` and so forth, and also increments the current time.
- `distance(...)`, which returns the current distance of the ship from the origin.

Make two instances of the class with different initial positions, and velocities. Write a `for` loop which moves each ship 100 times, and prints out the distance of each ship from the origin after each move.

6 Callable objects

In this problem, we re-do Problem 4 in a more elegant way, by defining a class instead of a function. The class, which we'll call `deriv` takes as its arguments a function `f` and a number `eps`, and creates a callable object which computes the estimate of the derivative of the specified function, using the specified value of `eps`. This callable object acts just like a function, but it is really an object. The advantage of doing this this way is that the function whose derivative is being calculated is stored in the object, as is the value of `eps`, so that the callable object needs only to have a single argument, `x`. Thus, once you define the `deriv` class you could compute the derivative of sin and cos as follows:

```
class deriv(f,eps):
     #Define the methods of your class here
     ...

import math
sinDeriv = deriv(math.sin,.0001)
cosDeriv = deriv(math.cos,.0001)
```

```
print sinDeriv(.5),math.cos(.5)
print cosDeriv(.5),-math.sin(.5)
```

(Reminder: To make an object callable, you need to define the `__call__` method in the class definition.)