Extra programming exercise for chapter 6

Øyvind Ryan (oyvindry@math.uio.no)

September 19, 2014

In chapter 6 we run into many different difference equations. It is therefore smart to split our code as follows.

1. Define a "kernel function" which only applies the formula of the difference equation.
2. Define a function which prints or stores the next values in the difference equation. This function should take a kernel function as in 1. as parameter. Since this is a general function working for any kind of difference equation, we should place it in a dedicated module.

Documentation for the functions you are required to implement in these exsercies can be found at `http://folk.uio.no/oyvindry/matinf1100/python/chap6/difference_eqs.html`

Exercise 1. Implement a function `difference_eq_print2(f, x0, x1, N)` which prints the next `N` iterations of a second order difference equation. `x0` and `x1` are the initial values. `f` represents the second order difference equation, and is a function which takes the three values x_n, x_{n+1}, n as input, and computes x_{n+2} from x_n and x_{n+1}, as in algorithm 6.11 (in general `f` may not represent a linear difference equation as in algorithm 6.11). Place `difference_eq_print2` in a module called `difference_eqs`.

Exercise 2. To test your code, implement a kernel function `fibonacci2` for the Fibonacci equation (exercise 6.3.2), and implement a program where the function `difference_eq_print2` is run with `f=fibonacci2` as parameter, and where `N` is read from the command line. Run the program with 10 iterations. If you have time, you can also write your program more generally so that the kernel function is read from the command line, and transformed into a function in your code using the `exec` command.

Exercise 3. Many difference equations are not of second order. Implement also functions `difference_eq_print1(f, x0, N)` and `difference_eq_print3(f, x0, x1, x2,N)` which does the same for first and third order difference equations, as `difference_eq_print2` does for second order difference equations (place also these functions in the module `difference_eqs`). To test your code, implement a kernel function `fibonacci3` for the third order Fibonacci equation of exercise 6.3.3, and a program where `difference_eq_print3` is run with `f=fibonacci3` as parameter, with `N` again read from the command line (run the

program with 10 iterations also here).

Exercise 4. It seems unnecessary to have one function for each order of difference equations as above. Sketch how you can write a general function for printing the values of a difference equation which works for any order (you are not required to implement this).

Exercise 5. We may be interested in doing other things with the values from a difference equation than printing them. In order also to plot the values, we need to store the values i a list for further use. Implement a function `difference_eq2(f, x0, x1, N)` which returns the next `N` values `x2`, `x3`,... in the difference equation in a list (place the function in the same module).