582332: Programming in Python, Fall 2009

Exercise sheet 4
28.9.-2.10.2009

1. Create a class `Fibonacci` whose instances are iterators that give the items of the *Fibonacci sequence* up to a given limit. The class should work like this:

   ```
   >>> list(Fibonacci(8))
   [0, 1, 1, 2, 3, 5]
   >>> for i in Fibonacci(8):
           print i      # would print the same numbers as above
   ```

2. Modify the exercise 2 from last week so that it allows *iterables* as its parameter. The returned value should still be a list.

   Try the modified function, for example, with iterables `xrange(40)` and `Fibonacci(40)`.

3. Create a class `Rational` whose instances are rational numbers. A new rational number can be created with the call to the class. For example, the call `r=Rational(1,4)` creates a rational number "one quarter". Make the instances support the following operations:

   ```
   +  -  *  /  <  >  ==
   ```

   with their natural behaviour. Make the rationals also printable so that from the printout we can clearly see that they are rational numbers.

4. Create a class `Mydate` that contains three attributes: day, month, and year. Have control over these attributes so that they cannot be set to illegal values.

   Make your class also printable in a nice format. So the following should work:

   ```
   print "%s" % Mydate(22,9,2009)
   ```

   Possible output formats could be a simple `"22.9.2009"` or a more complex `"22th of September 2009"`.

5. What happens in the following program?

   ```
   class B(object):
       def derived(cls):
           return not issubclass(B,cls)
       derived=classmethod(derived)
   class C(B):
       pass
   C.derived()
   B.derived()
   ```

6. Create a class `Glue` whose `__init__` method gets two lists as parameters. The resulting instance is an iterator over the sequence of pairs so that the first element of the pair comes from the first sequence and second element of the pair comes from the second sequence, until the shorter of the sequences runs out of elements.

   An example of usage:

   ```
   >>> list(Glue([1,2,3],[5,6,7,8]))
   [(1,5),(2,6),(3,7)]
   ```

7. Create a class `Histogram` that gets two lists as parameters. The first list consists of numbers. The second list consists of intervals of numbers. An interval is defined by a pair `(a,b)`, where `b` is the first element that is excluded. You can assume that the given intervals don't overlap.

   The class can then be queried how many elements of the first sequence belongs to a given interval. Also some nice printable output of the histogram should be possible.

8. Choose a topic for your project work. The topics are listed on the homepage of the course. Before the exercise session send the topic and your name by e-mail to your exercise group's teacher.

   In case you want to suggest your own topic, send a short description of what you are doing, so that your teacher can judge whether it is doable.