Exam for the course DAT171 Object oriented programming in Python

Time: 7th April 2016 8:30-12:30

Teacher: Mikael Öhman (mobile: 0736 837 674, office: 031 772 1301)

Permitted aids: Cay Horstmann: Python for everyone. Manuals and lecture notes are available on the computers.

Teacher will visit the rooms: Around 9:30 and 11:00-11:30

Formalities: In each file you hand in, you should write your examination code. Also enter the computer number (from the exam cover).

The documentation and lectures notes are available on `C:\__EXAM__`. Verify that these files are there immediately. Handing in the code should be done under the same folder: `C:\__EXAM__\Assignments\`, when you are finished. If you do not store the files here, they are not included in the exam. Save the files for each question in the appropriate folder, e.g. `C:\__EXAM__\Assignments\Question1\`. *Make sure you only hand in one solution for each question, otherwise you will receive **zero** points.* Complex sections of your code should have a descriptive comment of what is achieved.

When you finish the exam you should log out and fill in the (empty) exam cover page like normal and hand this in at the end of the exam.

Corrections: The results will be announced at the latest on 21th of April on the course homepage. The review will be the same day 12:20-13:00 and on the 26th of April 12:20-13:00.

Grading: There is a total of 25 points which yields grades at the standard 40%, 60%, and 80% limits:

```python
def grade(points):
    if points >= 20:
        return 'Grade 5'
    elif points >= 15:
        return 'Grade 4'
    elif points >= 10:
        return 'Grade 3'
    else:
        return 'Fail'
```

Question 1 (8p total)

Search engines use different methods for ranking the best pages. One such technique is to count how many other pages link to it. To do this, they scan through HTML pages, looking for all the links.

In HTML, links are specified as

```
Page with a link to <a href="www.google.com">google</a>.
Another link to <a href="http://www.amazon.co.uk/gp/deals">amazon deals</a>
HTTPS is also allowed (secure connection) like <a href="https://somebank.com/login">this</a>
```

Part A (3p)

Write a support function that extracts all links from a given text. It should work precisely like this example:

```
>>> x = extract_links('Links to <a href="http://www.amazon.co.uk/gp/deals">amazon deals</a>
and <a href="www.google.com">google</a>')
>>> print(x)
['http://www.amazon.co.uk/gp/deals', 'www.google.com']
```

You may assume that the links are correctly formatted. You may also assume that the links themselves do not contain the " character.

Hint: `help(str)`

Part B (2p)

Write another support function that picks out the domain name from a link by stripping off everything but the domains name itself, like in these examples:

```
>>> a = extract_domain('https://www.amazon.co.uk')
>>> b = extract_domain('http://www.foo.org/stuff/index.html')
>>> c = extract_domain('google.se/search?q=kitten')
>>> print(a)
www.amazon.co.uk
>>> print(b)
www.foo.org
>>> print(c)
google.se
```

Hint: Domain names can not contain special characters, like /

Part C (3p)

Read through the file `links.html` and extract all the domain names by making use of the functions you wrote for parts A and B. Count the frequencies of links to each domain (e.g. 34 links to `'www.google.com'`, 10 links to `'www.chalmers.se'`, 5 links to `'dx.doi.org'`) Print the domain names, and the number of links to this domain to the screen in a nice tabular format, for example, like this:

```
| Domain          | Links |
+-----------------+-------+
| www.google.com  |    34 |
| www.chalmers.se |    10 |
| dx.doi.org      |     5 |
```

and so on.

You may assume that there are no line-breaks inside the `<a href...</a>` links.

Hint: `Counter` from the `collections` module

Hint: str.format allows for alignment. You start by specifying colon + direction + width, like `':>5'` for right align width 5. Try out `'{:>5}'.format(123)` in the python command line. You may also find multiplication of strings a useful feature; `'+'*10`

Do not submit multiple copies of Part A or Part B. You may write all of these in the same file. Handing in multiple solutions results in ZERO POINTS.

Question 2 (5p total)

In this task NumPy and SciPy **must** be used for any array-type of data as well as for doing the integration.

A common problem in chemistry is to calculate the concentration of a constituent in a stirred tank reactor.

The concentration of constituent A can be described by

$$\frac{\mathrm{d}C(t)}{\mathrm{d}t} = \frac{F(t)}{V}(C_{in} - C) - kC^2$$

Here we assume: $C_{in} = 5.5$ mol/L, $V = 100$ L, $F(t) = 20.1$ L/min (constant) and $k = 0.15$ L/(mol*min)

Part A (1p)

Create a function calculating $\frac{\mathrm{d}C(t)}{\mathrm{d}t}$ given the parameters above.

The function should take exactly two argument: C and t

Note: t is only not really needed at this stage, but should be included for later use

Part B (2p)

Given the initial condition $C(0) = 0.5$ mol/L, find the time t_1 at which the concentration $C = C_1 = 1.0$ mol/L.

Answer with t_1 given one digit after the decimal comma.

Hint: Look at odeint form the SciPy library integrate *Hint: Plot the result C for t in the interval $0 <= t <= t$ using "'python import matplotlib.pyplot as plt

plt.plot(t, C) plt.show() "' *

Hint: Look at odeint form the SciPy library integrate

*Hint: You do not need to find the exact value for t_1, only to the given precision!

Part C (2p)

The process is considered stationary when the derivative $\frac{\mathrm{d}C(t)}{\mathrm{d}t} < 10^{-2}$. Find t_2 when that happens.

Answer with t_2 given one digit after the decimal comma.

*Hint: Add the derivative to the plot

*Hint: You do not need to find the exact value for t_2, only to the given precision!

Question 3 (4p total)

Part A (3p)

Write an iterator `Diagonal` that loops over the diagonal entries of a matrix. Example usage that should work:

```
>>> my_matrix = np.array([[1,2,3,4,5],[1,4,3,2,4],[5,4,2,3,3],[1,1,1,1,1],[2,0,0,3,0]])
>>> s = 0
>>> d = Diagonal(my_matrix)
>>> for x in d:
        s += x
>>> print(s)
8
>>> print(list(d))
[1, 4, 2, 1, 0]
```

It only needs to work for a `NumPy` matrix, but should work for any (square) size.

Hint: Lecture notes

Part B (1p)

If the matrix isn't square, it should raise a suitable exception. Write a `try-except` statement to verify that it works as correctly.

Question 4 (8p total)

Do not use any NumPy or SciPy for this question. You will need the `math` library for `sqrt`.

Vectors are critical part of almost any software. In many applications, we can use OOP to make special versions of vectors for practical and performance reasons.

Part A (0.5p)

Write a base class `BaseVector` with an abstract method `norm()`.

The `norm` is computed by $\sqrt{\sum_i a_i^2}$

Part B (1p)

Write the (trivial) subclass `ZeroVector` (a vector of all zeros) that takes a length, and nothing else. Add suitable implementations to the `len` operator, and the `norm` and `x[i]` (getitem) methods. `x[i]` should raise an `IndexError` if `i` is outside of the size.

```
>>> x = ZeroVector(10)
>>> print(x.norm())
0
>>> print(x[5])
0
>>> x[15]
...
IndexError: bla bla bla
```

Hint: `len` *support through* `__len__(self)`

Hint: `x[item]` *support through* `__getitem__(self, item)`

Part C (2p)

Write a subclass `Vector` that is a ordinary vector using a `list` to store its values. Add the implementations to the `len` operator the `norm` and `x[i]` (getitem) methods.

```
>>> x = Vector([3, 2, 6, 4, 2, 8])
>>> y = Vector([2, 7, 1, 1, 4, 3])
>>> print(x.norm())
11.533
```

Part D (3p)

Write a subclass `VectorSlice` that takes a `Vector` and a sub-range (slice) **without copying** any data from original `Vector`. Instead it simple stores a reference to the original, and uses the range to determine what part of the vector to represent.

Add the implementations to the `len` operator, the `norm` and `x[i]` (getitem) methods. `x[i]` should raise an `IndexError` if `i` is outside the slice range.

```
>>> x = Vector([1, 3, 5, 7, 9, 11, 13, 15, 17, 19])
>>> y = VectorSlice(x, 3, 6)  # will represent [7, 9, 11]
>>> print(y.norm())
15.843
>>> print(y[1])
9
```

Part E (1.5p)

Add a new method `slice(a, b)` to all three classes `Vector`, `VectorSlice`, `ZeroVector` and a corrsponding abstract method in `BaseVector`. Make use of the `VectorSlice` class where it is suitable.

```
>>> x = Vector([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> y = x.slice(2, 7) # y will contain values 2, 3, 4, 5, 6
>>> z = y.slice(3, 5) # z will contain values 5, 6
>>> u = ZeroVector(10)
>>> v = u.slice(2, 5) # v will contain only zeroes length
```

Hint: The zero vector is a bit special! The `slice` method does not have to return a `VectorSlice` type.

Hint: A slice of a slice is still just a slice.