UNIVERSITY OF OSLO

Faculty of Mathematics and Natural Sciences

Exam in **INF3331**
Day of exam: **2013-12-05**
Exam hours: **4**

This examination paper consists of 12 pages including 7 pages of appendices.
Appendices: **2 (Regex syntax and HTMLparser documentation)**
Permitted materials: **None**

Make sure that your copy of this examination paper is complete before answering.

It is possible to answer the exam in either Norwegian or English. For all the questions it is important to specify any assumptions you make, in particular in cases where you feel the question text is unclear.

1: Python-syntax (7 points)

Find errors in the following Python code. Write a suggestion with correct Python syntax for each case. For instance, `a = 4b` should be corrected to `a = 4*b`.

A)	`if __name__ = '__main__':`
B)	`mylist = range(0:100:10)`
C)	`def f(x)` `return 2+x**2`
D)	`for i in range[10]:` `  sum += 5i`
E)	`if a == 1,` `  b = a:`
F)	`import np as numpy` `a = np:arange(10)`
G)	`plt.title(nice figure)`

Answer:

```
A) if __name__ == '__main__':
B) mylist = range(0,100,10)
C) def f(x):
       return 2+x**2
D) for i in range(10):
       sum += 5*i
E) if a == 1:
       b = a
F) import numpy as np
       a = np.arange(10)
G) plt.title('nice figure')
```

Scoring this one should be fairly easy: One point for each correct answer. Could give 0.5 points if there are cases of doubt. To get full score, the students should write the correct code, but do not have to explain what is wrong with the code in the examples. If someone points out what is wrong but does not write the correct code, this can be given 0.5 point.

2: Vectorization (5 points)

The following Python script uses Monte-Carlo simulation to estimate the probability of getting at least three 6-es when you throw 10 dice.

```
import random
N = 1000
ndice = 10
nsix = 3
M=0
for i in range(N):
```

```
        sixes = 0
        for j in range(ndice):
            r = random.randint(1, 6)
            if r == 6:
                sixes += 1
        if sixes >= nsix:
            M += 1
p = float(M)/N
print 'probability:', p
```

Write a vectorized version of the script using NumPy. The answer should be a complete Python-script, including all import-statements.

Hint: `numpy.random.random_integers(n,size=(k,j))` generates an array with dimensions *k, j*, where each element is a random number between 1 and *n* (including the limits).

This one is relatively difficult, although it can be done with a single line. Here's a more readable example:

```
import numpy as np
n = 1000
ndice = 10
nsix = 3

dice = np.random.random_integers(6,size=(ndice,n))
p = np.sum(np.sum(dice==6,0)>=nsix)/float(n)
print 'probability:', p
```

The question text explicitly says "vectorized", which has been explained several times in the lectures, so a good score requires avoiding all Python loops. I suggest the following points:

5 points: nearly perfect, only minor typos allowed

3-4 points: correct thinking, i.e. a vectorized version with no explicit loops, but with incorrect syntax or wrong use of numpy tools (sum, Boolean arrays etc)

1-2 points: avoiding some, but not all for loops

Simply reproducing the original script, but looping over the array generated by numpy, should get no points.

3: Vectorization (3 points)

The following function computes the weighted sum of two vectors and returns the answer as a list. The arguments `x,y` may be lists, NumPy arrays or similar Python objects.

```
def vector_sum(x,y,a=1,b=1):
    sum = []
```

```
    for xi,yi in zip(x,y):
        sum.append(a*xi+b*yi)
    return sum
```

Write a function that performs the same operation, as efficient as possible. The arguments `x,y` should now be NumPy arrays, and the function should return a NumPy array. Include necessary import statements.

This one is very easy, although the question text may be slightly ambiguous. Since the arguments are supposed to be NumPy arrays, it is reasonable to assume NumPy was already imported. Therefore, full score should be given regardless of whether any import statements are used. The function reads:

```
def vector_sum(x,y,a=1,b=1):
     return a*x+b*y
```

Full score for getting this, or something very close. Two points if the idea is right but there are major errors. 1 point if correct thinking is applied, i.e. avoid all loops but not remembering that arrays can be added just like scalars. Zero points for anything involving a for loop.

4. Regular expressions (5 points)

A Django based web site has the following content in the file urls.py:

```
from django.conf.urls.defaults import *
urlpatterns = patterns('',
   (r'^articles/2003/$', 'news.views.special_case_2003'),
   (r'^articles/(\d{4})/$', 'news.views.year_archive'),
   (r'^articles/(\d{4})/(\d{2})/$', 'news.views.month_archive'),
   (r'^articles/(\d{4})/(\d{2})/(\d+)/$', 'news.views.article_detail'),
)
```

Recall that each URL-pattern in Django is a tuple with two elements. The first element is a regular expression, and the second element is the name of a view function that is called if the regular expression matches. The view-function is called with an HTTPrequest as first argument, and any groups in the regular expression as additional arguments.

We want two changes to the functionality:

1. Only proper dates should match; this means years between 1900 and 2099, months between 1 and 12, and days between 1 and 31.
2. The extpressions should accept dates on the form yyyy/mm/dd and yyyy-mm-dd. It should be optional to write the day and month using one or two digits. For instance, 2013/12/4 and 2013/12/04 should be equivalent.

Change or extend the list of URL-patterns to incorporate these changes.

Hint: Regex-syntax for “or” is | (”pipe”). For instance, the regular expression (a[bc]|x[yz]) matches either *a* followed by *b* or *c*, or *x* followed by *y* or *z*. See

also the enclosed regex documentation.

This one probably looks harder than it is. With all the hints given it should be possible to solve. We need to change three of the four regular expressions, but only list the changes to the last one, since these are the most advanced. This regex should work:

```
r'articles/(19\d{2}|20\d{2})[/-](1[12]|[1-9])[/-](1\d|2\d|3[01])/$'
```

Full score for anything close to this, 3-4 points for getting something correct but not all parts, 1-2 points for fulfilling one of the required extensions but completely failing the other.

5: Classes (5 points)

Explain what happens in the following Python code:

```
>>> class g:
...     cnt = 0
...     def __init__(self):
...         g.cnt +=1
...

>>> g0 = g()
>>> g1 = g()
>>> g0.cnt
2
>>> g1.cnt
2
>>> g0.__dict__
{}
>>> g1.cnt +=1
>>> g0.cnt
2
>>> g1.__dict__
{'cnt': 3}
```

The important part here is of course to get the distinction between class variables and instance variables, and providing a sensible explanation based on this. Full score for an explanation that also explains what the __dict__ attribute is, and that class attributes can be accessed but not assigned to through an instance of the class. 3-4 points for an explanation that covers most, but misses important parts, 1-2 points for getting something right.

6: Classes (10 points)

Implement a class for vectors in 3D. The class should support the following operations:

```
>>> from Vec3D import Vec3D
>>> u = Vec3D(1, 0, 0)  # (1,0,0) vector
>>> v = Vec3D(0, 1, 0)
>>> str(u)           # pretty print
'(1, 0, 0)'
>>> repr(u)         # u = eval(repr(u))
'Vec3D(1, 0, 0)'
>>> len(u)          # Eucledian norm
1.0
>>> u[1]            # subscripting
0.0
>>> v[2]=2.5        #subscripting with assignment
```

You do not need to consider the numerical efficiency of the operations. Remark that the len-operator returns the norm (length) of the vector. For a vector $u=(x,y,z)$ the norm $e(u)$ is defined as $e(u) = (x^2+y^2+z^2)^{1/2}$.

This is a reduced version of one of the weekly exercises given this fall, so many of the students will be familiar with it. The point of the question is of course to master the fundamentals of class programming in Python, i.e. the __init__ method and all the other special methods (5 in total). The somewhat non-standard use of len is intended as a test of the understanding of special methods, that they are tied to particular operators, but what they return is entirely up to the user. The solution is trivial, but has quite a few lines, and is probably not needed here.

A detailed suggestion for scoring is not so easy to provide in advance, but here's a suggestion:
10 points: all correct, maybe a minor typo or two
8-9 points: Mostly correct, with only minor errors such as wrong implementation of the Euclidian norm, minor errors in some of the special methods etc
6-7 points: Demonstrates knowledge of the basics of class programming in Python, including the right special methods etc, but with major errors in the syntax.
4-5 points: Major flaws such as missing some of the required special methods, consistently missing self as argument and for prefixing instance attributes, or other errors that demonstrate low knowledge of Python class programming.
1-3 points: Can be given to an answer that get something right, but with most of the features missing.

7: HTML parser (10 points)

As a new employee in Foomatic Inc. you took over an assignment from a previous employee. You are to complete a script that collects image files from a website, and stores them locally. The script you got looks as follows;

```
import HTMLParser
import urllib
import sys

urlString = "http://www.python.org"

def getImage(addr):
    u = urllib.urlopen(addr)
    data = u.read()

    splitPath = addr.split('/')
    fName = splitPath.pop()
    print fName

    f = open(fName, 'wb')
    f.write(data)
    f.close()

class parseImages(HTMLParser.HTMLParser):
    #find image tags and call getImage

lParser = parseImages()

u = urllib.urlopen(urlString)
print u.info()

lParser.feed(u.read())
lParser.close()
```

Write the contents of the class parseImages so that the script works as intended. Documentation for HTMLParser is attached.

With the supplied documentation, this one is not too hard. Especially since the class header is given, which reveals that it should be a sub-class of the HTMLParser class. The only thing needed is to implement one function correctly, and build up the right string as argument to the getImage function. Here's a working solution:

```
class parseImages(HTMLParser.HTMLParser):
    def handle_starttag(self, tag, attrs):
        if tag == 'img':
            for name,value in attrs:
                if name == 'src':
```

```
getImage(urlString + "/" + value)
```

With such a small code, and 10 points to hand out, the points scale needs to be fine-grained. Here's a suggestion:
10 points for a working solution with no errors
9 points for a working solution with minor errors (i.e. typos)
6-8 points for a solution that is mostly correct, but with syntax errors
4-5 points for getting some parts right, but with major gaps
1-3 points can be given for answers that are mostly wrong, but with some correct elements