UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Examination in:	INF1100 — Introduction to programming with scientific applications
Day of examination:	Monday, October 10, 2016
Examination hours:	15.00 – 19.00

This examination set consists of 11 pages.

Appendices:	None
Permitted aids:	None

Make sure that your copy of the examination set is complete before you start solving the problems.

- Read through the complete exercise set before you start solving the individual exercises. If you miss information in an exercise, you can provide your own reasonable assumptions as long as you explain that in detail.
- The maximum possible score on the exam is 25 points. The maximum number of points is listed for each question. For questions with sub-questions ((a),(b), etc), each sub-question has the same score.

(Continued on page 2.)

Question 1 (5 points)

What is printed in the terminal window when the programs below are run?

(a)

```
y = 4
n = 2
print y**n
```

(b)

```
a = "All work and no play makes Jack a dull boy".split()
print a[-2]
```

(c)

```
A = [[-1,0,1],[0,0,3],[5,6,7]]
print A[1][2]
```

(d)

```
s = 0
for n in range(2):
    s = s+3
print s
```

(e)

```
A = ['5', '6', '7', 'end']

try:
    b = float(A[4])
except IndexError:
    print 'A has length %d' %len(A)
except ValueError:
    print 'Cannot convert "%s" to float'% A[4]
```

(Continued on page 3.)

(f)

```
F = [C*0.5 + 30 for C in range(5)]
print F[-1]
```

(g)

```
import numpy as np
a = [7,6]
b = [3,2]
a_array = np.array(a)
b_array = np.array(b)

print a+b
print a_array+b_array
```

(h)

```
from numpy import *
v0 = 1.0
a = 1.0
t = linspace(0,1,3)
y = v0*t + a*t**2
for t_,y_ in zip(t,y):
    print '%4.2f %4.2f' %(t_, y_)
```

(i)

```
def f(x):
    return x**3

def test_f():
    x = 2.0
    expected = 8.0
    computed = f(x)
    tol = 1E-14
    success = abs(expected-computed) < tol
    msg = 'expected %g, computed %g' %(expected,computed)
```

(Continued on page 4.)

```
        assert success, msg

    test_f()
```

(j)

```
    def f(x):
        return 2*x

    def g(y):
        return y+7

    x=7
    print 'The magic number is %g' %f(g(f(x)))
```

Solution:

```
a:
16
b:
dull
c:
3
d:
6
e:
A has length 4
f:
32.0
g:
[7, 6, 3, 2]
[10  8]
h:
0.00 0.00
0.50 0.75
1.00 2.00
```

(Continued on page 5.)

```
i:
j:
The magic number is 42
```

On question `(i)`, nothing is printed since the test passes.

Question 2 (2 points)

The following code uses a `while`-loop to make a list of temperature values:

```
Clist = []
F = 0
while F < 40:
    Clist.append((F-32)*5.0/9)
    F += 5
```

Rewrite the code to use a `for`-loop instead of a `while`-loop.

Solution:

```
Clist = []
for F in range(0,40,5):
    Clist.append((F-32)*5.0/9)
```

Or with a list comprehension:

```
Clist = [(F-32)*5.0/9 for F in range(0,40,5)]
```

(Continued on page 6.)

Question 3 (3 points)

In this question we want to convert velocities given in miles per hour (mph) to meters per second (m/s) and kilometers per hour km/h. One mile is 1609.344 meters. Write a Python-funksjon `convert_velocity(v)` that takes a velocity `v` given in mph as input, and returns the velocity in m/s and km/h. Write code that uses the function to convert the velocity 10 mph to m/s and km/h, and prints the result to the screen.

Solution:

```
def convert_velocity(v):
    """get velocity in mph, convert to km/h and m/s"""
    mile = 1609.344
    v_kph = v*mile/1000
    v_mps = v*mile/3600

    return v_mps, v_kph

print convert_velocity(10)
```

Question 4 (6 points)

A text file named `velocities.dat` contains approximate maximum velocities for various animal species, given in miles per hour (mph), on the following form:

```
velocities (mph)
---------------------------------
Peregrin_falcon    200.00
Cheetah            70.00
Lion               50.00
Gazelle            50.00
Wildebeest         50.00
```

(Continued on page 7.)

```
Cat                30.00
Kangaroo           30.00
Human              27.89
Elephant           25.00
Black_mamba_snake  20.00
Pig                11.00
Sloth              0.15
Garden_snail       0.03
```

The number of lines in the file is not known, but the two first lines are always the same, and you can assume that none of the species names contain blanks.

(a) Write a Python-program that reads this file, calls the function in Question 3 to convert the velocities to m/s, and stores the result in two lists, named `animals` and `velocities`. The list `animals` shall contain the species names, while `velocities` contain the corresponding velocities in m/s.

(b) Assume that you have the lists `animals` and `velocities` from sub-question (a) available. Write the code to convert these lists to a single list, named `animal_velocities`, where each entry in the list is a pair (list or tuple). Each pair contains the species name and the corresponding velocity in m/s.

(c) Write the code for writing the converted velocities to a file named `velo_converted.dat`, with format similar to `velocities.dat`. You may assume that the lists from sub-questions (a) and (b) are available. We want a file with two columns, where the first column is the species name, and the second column is the velocity in m/s. The numbers in the second column shall be vertically aligned, as in the file `velocities.dat`.
Hint: The Python line
`print '%-20s %s' %('string1', 'string2')`
prints out
`string1              string2`,
that is, the first string is left-aligned and occupies 20 places (characters) in total.

Solution:

(Continued on page 8.)

```
#a:
infile = open('velocities.dat','r')
infile.readline()
infile.readline()

animals = []
velocities = []

for line in infile:
    words = line.split()
    animals.append(words[0])
    v = float(words[1])
    v_mps = convert_velocity(v)[0]
    velocities.append(v_mps)

infile.close()

#b:
#The simplest solution:
animal_velocities = zip(animals, velocities)

#Or a for loop:
animal_velocities = []

for ani_vel in zip(animals, velocities):
    animal_velocities.append(ani_vel)

#Or a list comprehension:
animal_velocities = [ani_vel for ani_vel in zip(animals, velocities)]

#c:
outfile = open('velo_converted.dat','w')

#the two header lines are optional
outfile.write('velocities (m/s)\n')
outfile.write('--------------------------------\n')

for ani, vel in animal_velocities:
```

(Continued on page 9.)

```
    line = '%-20s %g' %(ani, vel)
    outfile.write(line)
    outfile.write('\n')
outfile.close()
```

Question 5 (3 points)

The function `taylor_term(x,n)` computes one term in the Taylor-series of the exponential function, and is implemented as follows:

```
from math import factorial

def taylor_term(x,n):
    return x**n/factorial(n)
```

Write a test function `test_taylor_term()` for the function `taylor_term(x,n)`. Choose two combinations for x and n: $x = 0.5, n = 2$ and $x = 3.0, n = 5$, and compare the result of the function to the correct values, respectively 0.125 and 2.025.

Solution:

```
def test_taylor_term():
    """More elegant solutions exist, but
    this is one of the simplest"""
    x1 = 0.5; x2 = 3.0
    n1 = 2; n2 = 5
    exp1 = 0.125; exp2 = 2.025
    tol = 1e-10

    comp1 = taylor_term(x1,n1)
    comp2 = taylor_term(x2,n2)

    success = abs(comp1-exp1) < tol and abs(comp2-exp2) < tol
    assert success
```

(Continued on page 10.)

Question 6 (3 points)

The Taylor-series of order N for the exponential function is given by

$$e^x = \sum_{n=0}^{N} \frac{x^n}{n!}.$$

Write a program where you first set $x = 2.0, N = 5$, and then compute the sum and print the result to the screen. You may use the function from Question 5 to compute each term if you want to.

```
x=2.0
N=5
s = 0
for n in range(N+1):
    s += taylor_term(x,n)
print s

#Or without using the function from question 4:
x=2.0
N=5
s = 0
for n in range(N+1):
    s += x**n/factorial(n)
print s
```

Question 7 (3 points)

Extend the program from Question 6 so that x and N are read from the command line. Include a `try-except`-block which gives an error message and stops the program if it is used incorrectly. There should be one error message if the user provides too few command line arguments, and a different message if one or both of the arguments have

(Continued on page 11.)

the wrong format.

```
import sys
try:
    x = float(sys.argv[1])
    N = int(sys.argv[2])
except IndexError:
    print "You need to provide x and N as command line arguments"
    sys.exit(1)
except ValueError:
    print "The command line arguments must be numbers"
    sys.exit(1)
```

END