DELFT UNIVERSITY OF TECHNOLOGY
FACULTY OF AEROSPACE ENGINEERING

Course	:	Programming and Scientific Computing in Python (AE1205)
Date	:	June 30, 2015 from 9:00 until 12:00 hr
Lecturer	:	prof. dr. ir. Jacco Hoekstra, dr. ir. René van Paassen, Drs. Peter van Nieuwenhuizen
Remarks	:	Write your name, initials and student number on your work. Answer all questions in English and mark all pages with your name.

Instructions

NOTE: For Part I, II and III of the exam, you must use a (white) answer sheet. Computer use is *not allowed* for these parts. Do not switch on your computer/monitor, and work only with pen, paper and calculator.
After you completed parts I, II and III, raise your hand. You then hand in the answer sheet to an invigilator, and will receive a yellow(!) answer sheet for Part IV of the exam.
For Part IV you may use the computer. You can use the IDLE or Spyder development environments, and interactive python help through the interpreter. You may also use the Python documentation installed on the computers, to access this from Spyder, select "Help" and "Installed Python Modules" for the Matplotlib, Numpy or Scipy documentation.

Warning: Computer use is only allowed when working with the yellow answer sheet. Any other computer use must be reported as exam fraud.

Allowed Items

Parts I, II and III:
Reader AE1205
Scrap paper, ruler, protractor
Normal (non-graphical, non programmable) calculator.

Part IV:
Reader AE1205
Scrap paper, ruler, protractor
Computer, Python, Python(x,y) documentation (local only)

Grading information

A total of 90 points can be scored for the exam. The questions list the number of points that can be earned for each question. The final grade will be $1 + npoints/10$, and rounded off as per TU Delft policy.
Questions 1 to 6, 3 points each, total 18 points
Questions 7 and 8, 3 points each, total 6 points
Questions 9 and 10, 4 points each, total 8 points
Question 11: 13 points
Question 12, 13 and 14: 10 points each, total 30 points
Question 15: 15 points

Part I: what will this print?

1. What will the following program print?

```
a = range(6,-4,-1)

print a[-1:-4:-2]
```

2. What will the following program print?

```
haiku = [ 'i am', 'proper numbers',
          'am i', 'integer',
          'in order', 'haiku now']
punct = [ '?', ',' ]
idx = -1
count = 6
while count > 0:
    idx = idx + 1
    print haiku[idx], haiku[idx-3], punct[min(count-2, 1)]
    count = count - 2
```

3. What will the following program print?

```
digits = '0123456789'
result = 0
for digit in digits:
    if int(digit) % 2 == 0:
        result = -result + int(digit)
print(result)
```

4. What will the following program print?

```
tally = 0

for i in range(5):
    for j in range(3):
        tally = tally + j * i
print tally
```

5. What will the following program print?

```
n = 0
j = 0
for i in range(4):
    while j < (i % 2):
        n, j = n + 1, j +1
    n = n + 6
print n
```

6. What will the following program print?

```
import numpy as np
a = np.linspace(0,10,5)
b = np.arange(-1,1, 0.4)

print np.sum(b[a > 5.0])
```

Part II: Debug the program

7. which statement is true about the following program?

```
def myfun(x, a):
return x**2 + a*x + 5

a = 0
y = 3
print y, myfun(y, 2)
```

(A) The program will give an error message NameError: name 'x' is not defined

(B) The program has an indentation error

(C) The program runs correctly, and prints "3 14"

(D) The program runs correctly, and prints "3 20"

8. Which statement is true about the following program:

```
i = 7
for j in range(i/2):
    i = i + 2

print j
```

(A) This program is correct and will print 0 and 1

(B) This program will give a runtime error

(C) This program will only print 2

(D) This program will print 0, 1 and 2

(E) This program enters an infinite loop, and will have to be aborted

9. Consider the following program

```
 1  from math import *
 2  import numpy as np
 3  import matplotlib.pyplot as plt
 4
 5  ftab = np.arange(0, pi/2, float(1/100))
 6  xtab = cos(ftab)
 7  ytab = []
 8  for f in ftab:
 9      ytab.append(sin(f))
10
11  plt.plot(xtab, ytab)
```

Mark all statements about this program that are true (Note: multiple statements may apply):

(A) There will be a runtime error in line 2, since the program cannot import from both math and numpy

(B) There will by an runtime error in line five, in the np.arange call

(C) There will be a runtime error in line 6

(D) The program is correct, and plots a quarter circle

(E) The program will run correctly, but only plots a part of the quarter circle, since pi/2 is rounded to 1

(F) There will be a runtime error in line 11, in the `plt.plot` call, since a numpy array (`xtab`) cannot be combined with the `ytab` list

(G) After fixing all applicable errors above, the program will still not show a plot

10. A student wants to plot the formula $y = x^2$, for a range of $x = -2$ to $x = 2$ and he uses the following program to do this.

```
import matplotlib.pyplot as plt

xtab=[]
ytab=[]

for i in range(-20,20):

    x=i/100

    y=x*x

    xtab.append(x)
    ytab.append(y)

    if i == 0:
        print "halfway!"

plt.plot(xtab,ytab)
plt.show()
```

Which statements are true about the program:

(A) The program works correctly, and will show the parabola $y = x^2$ as intended

(B) The program works correctly, but only shows a parabola from -0.2 to 0.19

(C) The program produces a plot, but the plot does not show the parabola

(D) The program contains an indentation error and will not work

(E) The program contains a runtime error

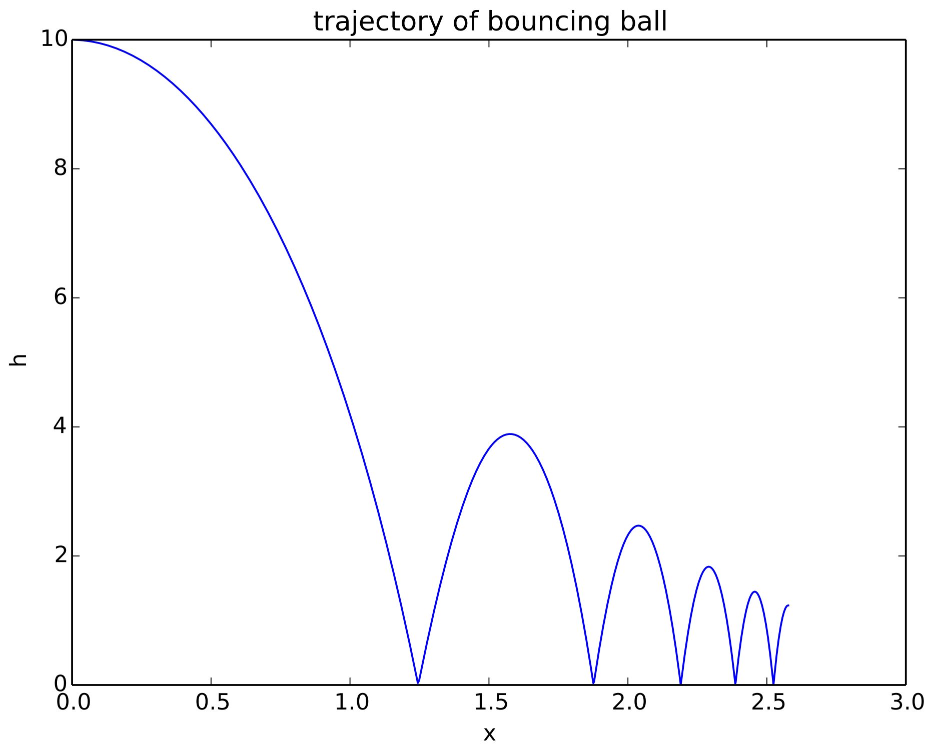
Part III: complete

11. The program below calculates how a ball moves after it has been launched with a horizontal velocity of 1 [m/s] from an initial height of 10 [m].

Bounces on the ground (at $y = 0$ [m]) are simulated as perfect, without energy loss, but the ball is subject to aerodynamic drag. The size of the ball is neglected, so it bounces at $y = 0$. The program stops when the ball is near the peak of its trajectory after bouncing five times

```
from matplotlib import pyplot as plt
import numpy as np

# inital state of the ball
vball = np.array([1., 0.])      # vx = 1, vy = 0
xball = ___(a)___               # position, x = 0, height = 10
# note: xball[0] is x-coordinate, xball[1] is y-coordinate
#       vball[0] is velocity in x direction, vball[1] in y direction
mball = 1.
g = 9.80665
S_times_cd = 0.12

xtab = []
dt = 0.01
nbounce = 0

# bounce 5 times, and then stop when the ball is at the top
while ___(b)___:

    # drag, total value, acting opposite to velocity (scalar!)
    fd = 0.5 * 1.225 * S_times_cd * np.linalg.norm(vball)**2

    # update velocity and position
    a = -(fd * vball / ___(c)___)/mball + np.array([0., -g])
    # Note: a[0] acceleration in x-direction, a[1] in y-direction
    vball = vball + a*dt
    xball = xball ___(d)___

    # bounce if hitting ground
    if xball[1] < 0:
        xball[1] = -xball[1]
        vball[1] = -vball[1]
        nbounce += 1

    # collect in a list
    xtab.append([xball[0], xball[1]])

xtab = np.array(xtab)
plt.plot(___(e)___)
plt.show()
```

For the intended output, see the included figure:



Complete the gaps in the program

Part IV: write

12. Consider the following two functions:

$$f(x) = \sin x * (0.003x^4 - 0.1x^3 + x^2 + 4x + 3)$$

and

$$g(x) = -10 \arctan x$$

Plot these functions on the interval from $x = -7$ up to $x = 7$. One of the points where the two functions cross is at $x = -5.91131$. Find the x-coordinates of the other four intersection by zooming into the plot by hand, with an accuracy of at least 3 digits.

13. Consider the following infinite series, evaluated for a coordinate x:

$$y = \sum_{n=1}^{\infty} \frac{1}{n^2} \cos(n\pi x)$$

When the partial sum of the first 1000 terms is evaluated for $x = 0.3$, the result is 0.38656

Implement the calculation given above, and evaluate the partial sum of the first 5000 terms for $x = 0.4$. Give your answer rounded on 6 digits behind the dot.

14. "Juffen" is a popular drinking game in Delft. Students count up (or down) in turn while trying to avoid the "forbidden" numbers, saying "Juf" instead of the number. The student who makes a mistake must empty his glass.

The forbidden numbers in Delft's version of Juffen are:

- Numbers that are divisible by seven (7, 14, 21, etc.)
- Numbers that are divisible by eleven (11, 22, 33, etc.)
- Numbers that contain (at any place), a 7 or an 11 (17, 112, etc.)

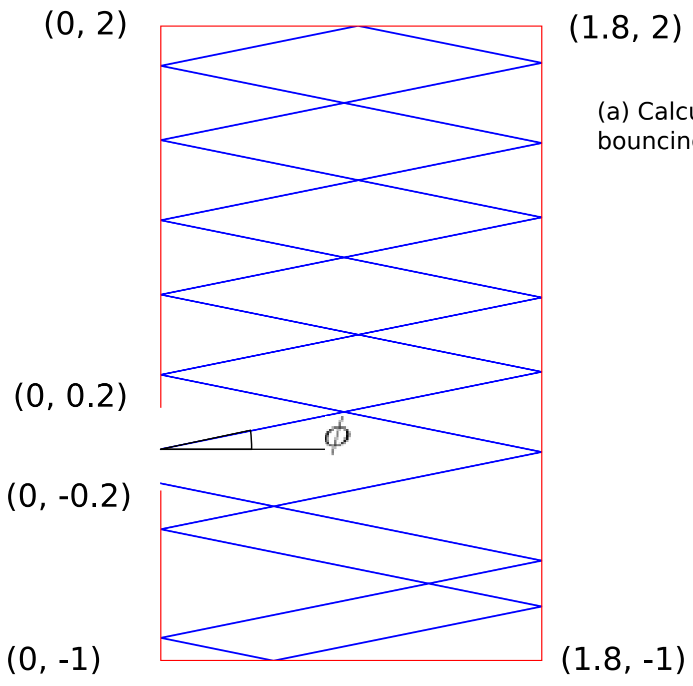
So this sequence starts as:

```
1, 2, 3, 4, 5, 6, juf , 8, 9, 10, juf , 12, 13, juf , 15, 16, juf , ...
```

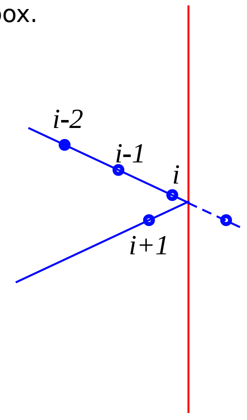
The sum of all numbers smaller than or equal to 200 that are *not* "Juf" numbers, is 11484.

Create a program to calculate this sequence, and sum all numbers smaller than or equal to 1000 that are not "Juf" numbers.

15. The picture below shows the result of a calculation of reflections of a (laser) beam entering a box. The walls of the box reflect the laser beam, and, as shown in the illustration, when a ray originating from point (0,0) enters the box with an angle of $\phi = 0.2$ [rad], the ray will reflect off the insides 17 times, before leaving the box again through the gap between $y = -0.2$ and $y = 0.2$



(a) Calculation of a light beam bouncing in a mirror-clad box.



(b) Detail of the "bouncing". When a point "i+1" is calculated to be outside one of the boundaries, take care to position its reflection inside at the correct position

Create a simulation of this process. You may use a numerical simulation, in which the beam travels in small steps, so given a direction of the beam ϕ, calculate:

$$\begin{bmatrix} x \\ y \end{bmatrix}_{i+1} = \begin{bmatrix} x \\ y \end{bmatrix}_i + 0.001 * \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix}$$

Repeat the calculation until the beam exits the box through the gap. If the beam "hits" one of the walls, correctly update the (x, y) position of the reflected beam, as shown in part (b) of the figure above. I suggest you plot the resulting trace, to verify the calculation, and check the correctness of your program with the data given for $\phi = 0.2$ [rad].

Then calculate the number of bounces for $\phi = 0.8$ [rad] and enter that as answer.