DELFT UNIVERSITY OF TECHNOLOGY
FACULTY OF AEROSPACE ENGINEERING

Course	:	Programming and Scientific Computing in Python (AE1205)
Date	:	August 18, 2017 from 13:30 until 16:30 hr
Lecturer	:	prof. dr. ir. Jacco Hoekstra, dr. ir. René van Paassen
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, and only with the provided account. 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, 8 and 9: 5 points, 4 points, 4 points; total 13 points
Question 10: 14 points
Question 11, 12 and 13: 15 points each; total 45 points

Part I: What will this print? 6x3 = 18 points

1. What will the following program print?

```
a   = [-1, 4, 2, 7, -2, 9, 3, 5 ]
eat = 0
for x in a:
    if eat > 0:
        eat = eat - 1
    else:
        if x < 0:
            eat = -x
        else:
            print x,
```

2. What will the following program print?

```
poem = """
much to and dad his mum's dismay
himself ate horace one day
stop didn't he to say he grace
his just sat ate and down his face"""

parts       = poem.split()
tmp         = parts[2::5]
parts[2::5] = parts[4::5]
parts[4::5] = tmp

for w in parts:
    if (w[-2:] == 'ay' or w[-3:] == "ace") and w[0] not in 'hs':
        print w
    else:
        print w,
```

3. What will the following program print?

```
monty = """Hello, this is your Captain speaking.""".lower().split()
monty.sort()
print monty[-1:-3:-1]
```

4. What will the following program print?

```
tri = [ [1] ]
for i in range(3):    # do 3 times
    lastrow = tri[-1]
    newrow  = []
    for j in range(len(lastrow)+1):
        if j == 0 or j == len(lastrow):
            newrow.append( 1 )
        else:
            newrow.append( lastrow[j - 1] + lastrow[j] )
    tri.append(newrow)
print tri
```

5. What will the following program print?

```
from numpy import *
a = arange(0., 6.5, 0.5)
print a[::2]
```

6. What will the following program print? (hint: what would a plot of the function look like?)

```
from numpy import *

def f(x):
    return sin(x)

t      = linspace(0, 15, 1501)
y      = f(t)
nexty  = y[1:]
y      = y[:-1]
t      = t[:-1]
switch = (y<0.0)*(nexty>0.0)
print t[switch]
```

Part II: Debug the programs 5+4+4 = 13 points

7. A student runs an experiment in which subjects had to press a touchscreen at highlighted locations. For each touch a line with x, y and time is written to a data file.

A student develops a program to read and plot this datafile. The file contains one or more comment lines starting with 'C', and then the lines with data.

Data lines contain the x and y coordinates as integers, scaled from 0 to 4095, with 0,0 being top left of the screen, and the time as a float, these three values are separated by comma's.

The actual display size is 40 by 30 cm. After reading the data, the program should convert the x and y coordinates to meters before plotting them, with 0,0 at the bottom left. To show the target areas, the program should draws three circles with a 1 cm radius in the plot. The plot also shows the different locations pressed marked by stars. When he runs the program, the student uses datafile `myfile.dat`. This file is present in the directory where the program is run.

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  # Open file for reading
5  fname    = input("Enter the filename: ")
6  datafile = open(fname, 'r')
7
8  # Read all x, y data
9  xt, yt = [], []
10 for line in datafile.readlines():
11     if line[0] != 'C':
12         x, y, time = line.split(',')
13         xt.append(int(x.strip()))
14         yt.append(int(y.strip()))
15
16 # Convert to numpy arrays and scale to meters
17 xt = np.array(xt, dtype=np.int) / 4095 * 0.4
18 yt = 0.3 - np.array(yt, dtype=np.int) / 4095 * 0.3
19
20 # Make circle coordinates and define centers
21 r    = 0.01
22 phi  = np.linspace(0, 2*np.pi, 100)
23 cx   = r*np.cos(phi)
24 cy   = r*np.sin(phi)
25 cloc = ((0.1, 0.1), (0.1, 0.2), (0.3, 0.15))
26
27 # Plot the circle and count the targets
28 for loc in cloc:
29     plt.plot(loc[0] + cx, loc[1] + cy, 'g')
30     tcount = 0
31     for i in range(len(xt)):
32         # Test if touch inside circle
33         if (xt[i] - loc[0])**2 + (yt[i] - loc[1])**2  >  r**2:
34             tcount = tcount + 1
35     print tcount, "hits at", loc
36
37 # Show all touch points
38 plt.plot(xt, yt, 'b*')
39 plt.show()
```

Indicate for the following statements whether they are true or false:

(A) The student needs to enter the filename with quotes (e.g., `'myfile.dat'`), or else the program fails

(B) The program correctly draws the three target circles

(C) The data conversion has an error, and therefore the data is plotted at the wrong location

(D) The program will ignore empty lines in the datafile

(E) The program correctly counts the number of hits

8. A student makes a plot with the following program. The plot should show a line at $y = 0.3$ and the function given below, for the range of $t = 0$ to $t = 10$:

$$f(t) = e^{-t/3}\cos(2t)$$

Check the listing for errors or problems, and indicate which statement applies to this program

```
1 import matplotlib.pyplot as plt
2 from numpy import *
3
4 t = arange(0.0, 0.001, 10.0)
5 y = cos(2*t)*exp(-t/3.0)
6 plt.plot(t, y)
7 plt.plot(t, 0.3+0.0*t)
8 plt.show()
```

Indicate which of the following statements is correct (one answer possible):

(A) The program will give an error message of the type IndexError

(B) The program will give a ValueError in matplotlib, "x and y must have same first dimension"

(C) The program runs without python errors, and correctly plots the function and the line at $y = 0.3$

(D) The program runs without error messages, and a plot window comes up, but the plot does not show the function, only the line at $y = 0.3$

(E) The program runs without error messages, and a plot window comes up, but the plot does not show the function, and also does not show the line at $y = 0.3$

9. Which statement is true about the following program?

```
1  import numpy as np
2
3  # Calculate the derivative of a polynomial
4  ad = None
5  def derivative_poly(a):
6      n = len(a) - 1                # derivative is one order less
7      ad = np.zeros((n)) # room for the result
8      for i in range(n):
9          ad[i] = (n-i)*a[i]
10     return
11
12 # Polynomial coefficients, defines polynomial, highest power first
13 # a[0]*x**n + a[1] * x**(n-1) + .. a[n]
14 a = np.array([0.3, -2.0, 1.5, -5.0, -1.2, 3.6])
15 derivative_poly(a)
16 print ad
```

(A) The program will give an error message of the type IndexError

(B) The program has a SyntaxError: invalid syntax

(C) The program runs without python errors, and correctly prints coefficients of the derivative of the polynomial

(D) The program runs without error messages, but does not print the coefficients

Part III: Complete the program 7x2 = 14 points

10. The following program calculates the trajectories of coconuts shot from point (0, 0), with a velocity 30 [m/s], at angles between 5 and 85 degrees. It uses numpy arrays as two-dimensional vectors. Complete the deleted parts:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # test in five degree steps, from 5 to 85, numpy array
5  alpha = ______(a)______          # different angles, in degrees!
6
7  V0 = 30                          # initial velocity [m/s]
8  Cd = 1.2                         # drag coefficient [-]
9  A = 0.02                         # frontal area [m2]
10 M = 1.5                          # nut mass [kg]
11 gvec = ______(b)______           # gravity vector (x,y)!, 9.813 [m/s2]
12 rho = 1.225                      # density, [kg/m3]
13 dt = 0.01                        # time step [s]
14
15 # test loop
16 for a in alpha:
17     # initial position and velocity vectors
18     xvec = np.array([0.0,0.0])
19     vvec = V0 * np.array([ ______(c)______ ])
20
21     # for each alpha to collect all x,y position values
22     xall = []
23
24     # updates while ball above ground
25     while ______(d)______:
26         # absolute value of velocity
27         V = np.sqrt(np.sum(vvec*vvec))
28         # Drag vector, magnitude 1/2 rho Cd A V^2
29         # Use -vx/V, -vy/V for the direction of x, y drag
30         Dvec = -0.5*rho*Cd*A* ____(e)_____
31         # Acceleration, vector addition
32         accvec = gvec+Dvec/M
33         # Integration
34         vvec = vvec + accvec*dt
35         xvec = xvec + vvec*dt
36         # Assemble for plot, simply collect all xvec in list
37         ______(f)______
38
39     # convert to numpy array for the plot
40     xall = np.array(xall)
41
42     # plot this trajectory in x, y plot
43     plt.plot(xall[:,0], ____(g)____)
44 plt.show()
```

Part IV: Write programs 3x15 = 45 points

Submit your solutions on the answer sheet + by saving your files! Read here how.

You need to enter the answer on the answer sheet and give a one or two line description of your method on the answer form.

Also save your scripts on the volume **H:** on your computer. Make a folder named after your student number (e.g. 4345678). Start all program files names with a 'p' followed by the problem number. So p11motion.py is a good name. You're free to use any name you want after the 'p11', etc. but it has to start with these three characters so we can find your program automatically as: H:/4345678/p11*.py, H:/4345678/p12*.py, etc.

As always, adding comments, a header with student name+number, using whitespace and clear variable naming are appreciated.

11. Motion path

A motion path is described by the following set of 3 equations.

$$\frac{dx}{dt} = 10(y - x)$$
$$\frac{dy}{dt} = 28x - y - xz$$
$$\frac{dz}{dt} = xy - \frac{8}{3}z$$

Simulate this numerically with steps of $dt = 0.001$, using Euler integration. As an example, for a variable u, if you use the variable `dudt` for du/dt, do an integration step with:

```
1 u = u + 0.001 * dudt
```

Start with initial values 0.0, 3.0, -10.0 for x, y, z. Simulate for 8 seconds, and plot y against x. How many times does the plot cross the line $x = 5$? Also write down the final values of x, y and z after the calculation.

You can use the following data to check your calculation; if you run the simulation for 1.0 s, the x, y, and z values will be -8.051, -11.111, 21.597

By the way, this set of equations is known as the Lorentz "strange attractor". Don't forget to save the program file on the H: drive!

12. Division sequence tail

The 3n+1 problem is a conjecture about the following algorithm that generates a sequence from an positive integer n:

- if n is even, divide n by 2
- if n is odd, multiply by 3 and add 1
- repeat this procedure while n is not equal to 1

It is suspected that this procedure always ends up at 1. As an example, the number 75 generates the sequence $75 \rightarrow 226 \rightarrow 113 \rightarrow 340 \rightarrow 170 \rightarrow 85 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

It may be clear that the sequence ends once the number n becomes a power of 2, since from then on the number can be divided by 2 until it becomes one.

If you test this for all numbers up to 100, you will find that there are two numbers that generate the largest sequence of divisions by 2, namely 8 divisions. These numbers are **75** and **85**, which both have 256 (2^8) in their sequence.

Create a program that implements this algorithm, test it with numbers n up to 30000. Determine the length of the largest sequence of divisions by 2 at the end of the sequence. On the sheet enter all the starting numbers below 30000 that which have this largest amount of divisions by 2 at the end of their sequence.

Save the program on the H: drive in the folder named after your student id.

13. Elevator time

Create a program to calculate the total time an elevator takes for a row of 100 people. The elevator has a car that can hold a maximum of 9 people. We assume the elevator can be filled in 7 seconds, and it fills to its maximum capacity each time, and finally to the number of people still waiting in the hall for the last ride.

The simulation starts with a row of 100 people waiting to get to different floors.The code below creates a list of 100 integers for the people in the order of how they enter the elevator, specifying the number of the floor they want to go to. So if the list waiting starts with [11, 11, 7, ...], this means the first two persons want to go to the 11th floor, while the third wants to go to the 7th floor.

First, copy this code at the start of your program:

```
1 from random import randint, seed
2
3 def fillhall(seedno, npeople):
4     seed(seedno)
5     return [randint(2,13) for i in xrange(npeople)]
6 waiting = fillhall(6, 100)
```

Using the elements of the list, which determine which floor the people have to go to, start filling the car from the front of the list, then take the next group etc. We can now

derive the formula for the travel time of a single trip for a car based on the floors the people currently in the car want to go to:

It always takes 7 [s] for a car to get filled. The car fills with the *maximum of 9 people*, and is only partially filled if the last batch enters, the car stops for 5 [s] at each floor where people have to leave, and it takes 2.3 [s] to get to a floor above, and 2 [s] to descend a floor. With n_f as the number of *different* floors that the people in the car have to go to, and f_{max} as the highest floor selected by the people in the car, the travel time for a single trip is calculated by:

$$T = 7 + (2.3 + 2.0) f_{max} + 5.0 n_f$$

Calculate the total travel time of all trips for processing the 100 people waiting in the hall, and consider the end of the travel time to be when the elevator is back down again after all trips.

To check your program, first generate an alternative test list with 300 people, with by calling the fillhall function with different values:

```
1 waiting = fillhall(0, 300)
```

Check your program: The total travel time for this list will be 3108.9 [s].

For the final version, use

```
1 waiting = fillhall(6, 100)
```

again, and re-run the program; then fill in the answer you get on the sheet.

Also give a short description of your approach on the answer sheet and save the program on the H: drive