Part I: what will this print?

1. What will the following program print?

```
1 rx = range(12, 3, -3)
2 mysum = 0
3
4 for i in rx[::-1]:
5     mysum = i - mysum
6
7 print i, mysum
```

2. What will the following program print?

```
1 v1 = "We are"
2 v2 = 'the Knights who say: "'
3 v3 = "Ni"
4 print
5 print ' '.join([v1, v2]) + '!", "'.join([v3]*3)
```

3. What will the following program print?

```
1 j = 0
2 r = 0
3 for o in range(5):
4     while j < 2*o:
5         if o % 3 == 0:
6             break
7         r = r + j
8         j = j + 1
9 print r
```

4. What will the following program print?

```
1 a = "Wink wink nudge nudge say no more".split()
2 t = 1
3 for x in a[::-1]:
4     t = -2*t
5 print t, x
```

5. What will the following program print?

```
1 import numpy as np
2 t = np.linspace(-1, 1, 5)
3 y = t*t+t
4 y = (t > 0) * y
5 print y
```

6. What will the following program print? Please give the answer numerically in 3 significant digits. A mathematically exact answer is also acceptable.

```
1 from numpy import *
2 t = arange(10,0,-1)
3 # Note: exp(x) is e to the power of x
4 print min(1.0/exp(t)), max(1.0/exp(t))
```

Part II: Debug the program

7. Which statement is true about the following program?

```
1  from matplotlib import pyplot as plt
2  from numpy import *
3
4  def myfun(x):
5      y = -3 * x**2 + 4*x + 6
6
7  t = arange(10,-10,-1)
8  y = myfun(t)
9  plt.plot(t, y)
10 plt.show()
```

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

B The program has an indentation error

C The program has a runtime error from matplotlib, "ValueError: x and y must have same first dimension"

D The program runs correctly, and plots a graph from x=-9 to x=10

E The program runs correctly, and plots a graph from x=-10 to x=9

8. Which statement is true about the following program?

```
1 import numpy as np
2 rvec = np.linspace(-2.0,2.0,5)
3 for i in rvec:
4     print i,
5 print
```

A The program will give an error message TypeError: integer expected

B The program has a SyntaxError: invalid syntax

C The program runs correctly, and prints '-2.0 -1.0 0.0 1.0 2.0'

D The program runs correctly, and prints '-2.0 -1.0 0.0 1.0'

E The program runs correctly, and prints '0 1 2 3 4'

9. The following program calculates the pressure and temperature of a diatomic gas after compression.

Which of the following statements are true about this program (multiple statements may apply)?

```
1  P0 = 100000          # [Pa], atmospheric pressure
2  gamma = 1.4          # for diatomic gasses, O2, N2 ...
3  more = True
4
5  while more:
6      T0 = input("Temperature (C)  :") + 273.15
7      r =  input("compression ratio:")
8
9      P = P0 * r ** gamma
10     T = T0 * P/P0 * (1/r)
11
12     print "Pressure after adiabatic compression", P
13     print "Temperature after adiabatic compression", T-273.15
14
15     ans = ''
16     while len(ans)==0 or (ans[0] != 'n' or ans[0] != 'y'):
17         ans = raw_input("Continue (Y/N)? ").lower()
18     more = ans[0] == 'y'
19
20 print "Thank you for using the adiabatic compression program"
```

A The program has a SyntaxError: invalid syntax

B The program can produce the following output

```
Temperature (C)  :15
compression ratio:3
Pressure after adiabatic compression 465553.672175
Temperature after adiabatic compression -273.15
```

(the 15 and 3 were entered by the user)

C The program will correctly calculate the pressure and temperature for the following input:

```
Temperature (C)  :15
compression ratio:3.5
```

D After entering a return in response to 'Continue (Y/N)? ', the program ends normally

E After entering 'No' (without quotes) in response to 'Continue (Y/N)? ', the program ends normally

F After entering 'Y' (without quotes) in response to 'Continue (Y/N)? ', the program asks for a new temperature and pressure

G The program will end with an IndexError in line 17

Part III: complete

10. The following program creates balls at random intervals, with a random initial speed and position, at the bottom of the screen, and then simulates and shows the movement of the balls. Balls are simulated with gravity, aerodynamic drag has not been implemented.

Carefully read the partial program, and complete the missing parts.

```
1  import pygame as pg
2  from random import random as uniform  # uniform random numbers 0 .. 1
3
4  pg.init()                             # initialize pygame
5  scr = pg.display.set_mode((800,600))  # screen of 800x600 points
6
7  background = (14, 14, 70)             # background color
8  ballcolor = (255,70,70)               # ball color
9  ___(a)___                             # list of balls, empty for now
10 g0 = -9.80665                         # gravity
11
12 def ballUpdate(ball, dt):
13     x, y, vx, vy = ball
14     vy = vy + g0*dt
15     return _________(b)__________     # calculate and return ball tuple
16                                       # with update of x, y position
17 time = 0.0
18 balltime = uniform()
19 tick0 = pg.time.get_ticks()
20
21 while len(balls) < 10:                # stop the program at 10 balls
22     tick = pg.time.get_ticks()
23     dt = (____(c)_____) * 0.001       # loop time step in seconds
24     tick0 = tick                      # remember time in this cycle
25
26     scr.fill(background)
27     for i in range(len(balls)): # iterate over all existing balls
28         balls[i] = ballUpdate(balls[i], dt)
29         if ______(d)________:         # balls are only drawn when y > 0
30             pg.draw.circle(
31                 scr, ballcolor,
32                 (int(balls[i][0]*100), int(600-balls[i][1]*100)), 10)
33
34     # new ball?
35     balltime = balltime - dt          # reduce time until next ball
36
37     if balltime < 0.0:                # time for another ball
38         # new ball, x position, y position, x speed, y speed
39         # the x speed is a random value between -2.5 and 2.5
40         # the y speed is random between 5 and 15
41         balls.append((8.*uniform(), 0.01,
42                       ______(e)______, 10*uniform()+5.0))
43         balltime = 0.1+uniform()
```

```
44
45      pg.display.flip()
46      pg.event.pump()                 # remove stacked up window events
47
48  pg.quit()
```

Part IV: write

11. Consider the following equation:

$$f(x) = \frac{(1+0.2x)(1+0.1x)}{(1+x)(1+0.02x)^2(1+0.01x)^2}$$

Implement the equation as a function in Python. As a check on your implementation, the function's value for f(2j) = (0.3905-0.2964j)

For some function evaluations, a logarithmically spaced array of points produces a more readable plot. The array x=[0.1, 0.316, 1.0, 3.162, 10.0] for example is logarithmically spaced, since $\log(x[i]) - \log(x[i-1]) = 0.5$, for valid i.

Create a range of points for x, logarithmically spaced between 0.0001 and 100. Then multiply this range with 1j (the imaginary number in Python), so you get a range of imaginary values. Calculate the function f for these points; the outcome will be a list or array of complex numbers. Plot the real part against the imaginary part of this output (hint: np.real, np.imag). Ensure that the plot is smooth by selecting enough points in the array (but note that excessively many points may crash python). Then find the points where the imaginary part of f is -0.2j

List these points in your answer with an accuracy of 3 decimals.

12. For some numbers, the sum of the factorials of their digits is divisible by the number itself. For example 19 is such a number, because

$$1! + 9! = 1 + 362880 = 362881$$

and 362881 is divisible by 19.

Write a program to find the largest 2 divisible numbers under 10000, and enter those on the answer sheet.

P.S., Note that 0! = 1

13. To test whether a number X is divisible by 7, you can apply the L-2M rule. What you do is double the last digit of the number X and subtract it from X without its last digit. For instance, if the number X you are testing is 345678, you would subtract 16 (2x8) from 34567, and end up with 34551.

Repeat this procedure, now with 34551, etcetera, until you get a number that you know for sure is or is not divisible by seven. Then the number X's divisibility will be the same.

Write a program that applies the L-2M rule, and run it until you are left with a 2-digit number (i.e., a number smaller than 100).

Test this program against the above example 345678. After applying the L-2M rule 4 times, you are left with the number 15, which is not divisible by 7.

Then take the (very large) number (please carefully copy it, and note that the commas are only there to separate the thousands)

87,425,473,459,527,629,989,378,641,469

Or create this same number with the following code snippet:

```
1 import random as rnd
2 L = 29
3 rnd.seed(0)
4 bignumber = int('').join([ rnd.choice('0123456789')
5                            for i in range(L) ]))
```

How many steps are needed until you are left with a two-digit number, (i.e., stop at the first number below 100) and what is that two-digit number?

14. Create a program to play tic-tac-toe.

This program has two players, each with a different tactic.

- The first player (computer player) plays with random moves, based on the random generator. This player uses a symbol 'x' to mark the board.
- The second player simply finds the first *free* square starting in the topmost row, starting at the top left. Then if the topmost row is filled, the player finds the leftmost free square in the second row, etc. The second player marks with 'o'.

Initialise the random number generation with a seed (do that only once in your program!)

```
1 import random as rnd
2 rnd.seed(0)
```

The code to determine the 1st player's moves is:

```
1 row = rnd.randint(0,2)
2 col = rnd.randint(0,2)
```

Note that if the square determined in that way is occupied, another random square is chosen with the same method, until a free square is found and then marked.

The game ends when one of the players wins, or all squares are occupied. To check your implementation, once play the game with a seed of 1. The result is:

```
1 o o x
2 . x .
3 x . .
```

Here 'x' is a move from the first player (with the random moves), 'o' is from the second player (starts at top left, and simply finds the first free square, left to right, and top to bottom). The free fields are marked with a '.'. You can see that the game ended after five moves, with the first player winning.

Now play the game with a seed of 0, and enter the result on the answer form in the same manner as in the example above.