IE Informatique et Société Numérique 2
SCAN - April 2024

Duration: 1h30
The use of additional documents or calculators is strictly prohibited.

Programs with **bad indentation** or **poorly chosen variable names** will be penalized.

Exercise 1 Code reading (3.5pts)

(Q1.1) What will be displayed by the following code?

```
1 t1 = []
2 n = 2
3 l = []
4 for i in range(n):
5     for j in range(n):
6         l.append(i*j)
7     t1.append(l)
8 t1[0][1]=5
9 print(t1)
```

(Q1.2) What will be displayed by the following code?

```
1 t2 = [[1,2,3],[8,9,3]]
2 n = 3
3 for i in range(n):
4     for j in range(n):
5         print(t2[i][j])
6 print(t2)
```

(Q1.3) What will be displayed by the following code?

```
1  def decrement(a):
2      a = a -1
3
4  def sub(l):
5      for i in range(len(l)):
6         decrement(l[i])
7
8  def my_function(l,i):
9      l[i] = decrement(l[i])
10
11 x = 2
12 print(x)
13 l = [6, 7, 8]
14 sub(l)
15 print(l)
16 my_function(l,1)
17 print(l)
```

Exercise 2 Code correction (2,5pts)

(Q2.1) Identify any potential bugs and the two errors in the documentation of the program below.

```
1  def find_positions(grid, value) :
2  """Searches for the position of each occurrence of 'value' in grid.
3     Parameters :
4         grid : a list of integers
5         value : the integer being searched for
6     Returns :
7         A 2D list of integers. Each inner list is of size 2 and
8         corresponds to the coordinates of a value, formatted as
9         [line, column]"""
10     res = []
11     for num_line in range(len(grid[0])) :
12         for num_col in range(len(grid)) :
13             if grid[num_line][num_col] == value:
14                 res.append(num_line)
15                 res.append(num_col)
16     return res
17
18 def replace(grid, positions, val) :
19     """Stores val in grid at each location indicated in positions.
20        Parameters :
21            grid : a 2D list of integers
22            positions : a 2D list of integers (a list of positions
23                        [line, column] valid in grid)
24            val : integer to store
25        Returns : the modified grid """
26     for pos in positions :
27         grid[pos[1]][pos[0]] = val
28
29 my_list = [[1,2,5,9],[9,5,1,3],[8,7,4,1],[1,4,9,1]]
30 pos = find_positions(my_list, 1)
31 replace(my_list, pos, 42)
32 print(my_list)
```

Exercise 3 Problem - Sorting of Complex Numbers (14 pts)

Important : The questions at the end of the exercise (in particular 3.4 to 3.6) can be attempted even if the initial questions were skipped. You can use a function even if you did not provide its algorithm or code (but its header must have been defined).

Our aim is to study a sorting algorithm that works on complex numbers rather than on integers.

Comparison and ordering To compare and order complex numbers, we will use their norm (or modulus). For example, if we have the numbers $c_1 = -2 + 2i$ and $c_2 = 1 + i$, their norm are

$$|c_1| = \sqrt{(-2)^2 + (2)^2} = 2\sqrt{2}$$

and

$$|c_2| = \sqrt{1^2 + 1^2} = \sqrt{2}$$

Also, as :

$$|c_2| < |c_1|$$

we will consider that :

$$c_2 < c_1$$

Note that your sorting algorithm will thus consider two complex numbers as equal if they have the same norm.

Choice of representation for complex numbers To represent complex numbers, we will use lists with two elements, the first element representing the real part, and the second element the imaginary part. For example, the numbers c_1 and c_2 will be represented by :

```
1 c1 = [-2,2]
2 c2 = [1,1]
```

Sorting by selection For the sorting algorithm, you will use selection sort on integers, whose algorithm is reminded below :

```
1 # Function selection_sort(list of values) :
2 #     For each index i_pivot in the list
3 #         Search for the index i_min of the smallest element starting from the index i_pivot
4 # If necessary, swap the elements at indices i_pivot and i_min
```

(Q3.1) Propose a function `display_complex_list` that takes a list of complex numbers as a parameter and displays it exactly in the format specified below (1pt)

The call `display_complex_list([[-2,2],[1,-1]])` should display exactly :

```
[ (-2 + 2i) (1 + -1i) ]
```

(Q3.2) With the help of the reminder below, propose a functional decomposition of the sorting algorithm that takes a list of complex numbers as input and sorts it. This decomposition must include at least 3 functions (not counting `sort`). (3pts)

> **Reminder :** To perform a functional decomposition, one must :
> — identify the functions present in the provided algorithm ;
> — for each, provide a **definition**, a **description** (docstring), and an **algorithm** ;
> — apply the functional decomposition to these new algorithms until the algorithm of each function is trivial.

(Q3.3) Implement the `sort` function from the provided algorithm according to your functional decomposition. Also implement all its sub-functions. You do not need to manage imports, nor write the docstrings that would have already been written previously. (6pts)

(Q3.4) Write a function that tests if a list of complex numbers is sorted. (1 pt)

(Q3.5) Propose a Python code that performs the following operations using the previously defined functions. (1 pt)

— Define the list `l` containing the following complex numbers : $1+0i$, $-2+1i$, $6-2i$, and $2-1i$;
— Display the list `l`
— Test if `l` is sorted
— If `l` is not sorted, sort the list `l`
— Display the list `l`

(Q3.6) For one of your functions, propose a set of tests covering 4 cases. Explain why these cases are relevant. (2pts)