Project N°01 - Python
Semester 1 - 2019/2020
SCHUR'S NUMBER S(n)

1. Preamble

The first project of the first semester is proposed to allow you to pursue your learning process in algorithmic and Python programming language. It is an opportunity for you to practice the skills acquired during the lectures, tutorials and lab sessions to achieve a concrete application. Working on this project will test your ability to:

- Implement an algorithmic solution coming from a problem description
- Use / Choose appropriate loop structures
- Use conditional structures and define the corresponding Boolean expressions
- Use arrays

2. Presentation of the project

A century ago, in 1916, the great mathematician Issai Schur (1875-1941) gave birth to a series of integers ***S(1), S(2), S(3), ...*** very interesting and so mysterious that, even 100 years later, we still know almost nothing about them.

The problem concerns colouring integer numbers using a certain number of colours and respecting some constraints. Schur thinks in his theorem that (**n** being the number of colours to consider):

Schur's Colouring Theorem (Schur, 1916):
Given a postive natural number **n**, there exists a number **S(n)** defined as follows :
S(n) is the highest integer value for which it is possible to colour, using **n** colours, the integers **1 … N** while excluding any monochromatic triplet (**a**,**b**,**a+b**) where **a**, **b** and **a+b** are between **1** and N.

For Schur, a triplet ***(a, b, c)*** with ***c* = *a* + *b*** is said to be monochromatic if the three integers ***a***, ***b*** and ***c*** have been coloured with the same colour after the coloring phase. Moreover, it is not forbidden to have triplets of type ***(a, a, c)*** (i.e., ***a* = *b*** and ***c* = 2*a***).

The objective of this project is to create a program that calculates ***N*** for any given number of colors ***n*** (***n*> = 2**) .

In order to reach this goal, we describe in the next section the process to follow in order to progressively create your final program.

3. Approach

3.1. Understanding the problem

Given a set of n colours (**n> = 2**), the question is: what is the highest integer number **N** for which it is possible to colour the numbers **1 … N** with these **n** colours (each colour must be used at least once) so that the emergence of monochromatic triplets (**a**, **b**, **a + b**) is avoided?

Once this value of **N** found, we can say that **S(n)=N**.

To find this value of ***N*** for ***n*** colors, no formula is required. Simply:

- For any **N>n**,
- List all triplets of the form ***(a, b, a+b)*** having ***a***, ***b***, ***a+b*** between **1** and ***N***.
- Find at least one coloring in which no triplet is monochromatic.
- Make sure this condition is not true for all possible colorings with ***n*** colors and (***N + 1***) integers.

Example:

In the following, we explain the process to compute ***S(2)***. We assume that for two colours (***n=2***), we use the colours Red and Blue to colour the numbers. We are going to create the series of integers 1 à ***N*** with N between **n+1** and **S(2)+1**. For each value of ***N,*** we check if there exists at least on colouring satisfying the following predicate:

All triplets ***(a, b, c)****, with* ***c=a + b*** *(****a*** *could be equal to* ***b****), created between 1 and* ***N*** *are not monochromatic. This predicate is proved unsatisfied as soon as a monochromatic triplet is found.*

Lets consider the following table that illustrates the how ***S(2)*** could be computed. It is composed of 2 main columns and 2 main rows. In the first row, we detail all the possible colouring starting the integer "1" coloured in red. In the second, we keep the series of integers starting with the integer "1" coloured in blue and whose truth value of the predicate could be true. When we are sure that a given colouring can not satisfy the predicate, we do not treat it.

By setting the colour of the first integer to red "1", the second integer (2) could be red or blue. Thus, for ***N* = 2**, we have tow possible colouring: "12" and "12". The triplets associated with these colouring area (1, 1, 2) and (1, 1, 2) respectively.

N=1	N=2	N=3	N=4	N=5	Triplets associated with each colouring; Eval(Predicate) : N
				~~1 2 3 4 5~~	~~(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5)~~ **F:5**
			~~1 2 3 4~~		~~(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4)~~ **F:4**
				~~1 2 3 4 5~~	~~(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ;~~ (1, 4, 5); ~~(2, 2 , 4);~~ (2, 3, 5) **F:4**
		~~1 2 3~~			~~(1, 1 , 2) ; (1, 2 , 3)~~ **F:3**
				~~1 2 3 4 5~~	~~(1, 1 , 2) ; (1, 2 , 3)~~ ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); ~~(2, 3, 5)~~ **F:5**
			~~1 2 3 4~~		~~(1, 1 , 2) ; (1, 2 , 3)~~ ; (1, 3 , 4) ; (2, 2 , 4) **F:4**
				~~1 2 3 4 5~~	~~(1, 1 , 2) ; (1, 2 , 3)~~ ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
	~~1 2~~				~~(1, 1 , 2)~~ ; Premier triplet monochromatique **F:2**
				~~1 2 3 4 5~~	~~(1, 1 , 2)~~ ; (1, 2 , 3) ; (1, 3 , 4) ; ~~(1, 4, 5); (2, 2 , 4);~~ (2, 3, 5) **F:5**
			~~1 2 3 4~~		~~(1, 1 , 2)~~ ; (1, 2 , 3) ; (1, 3 , 4) ; ~~(2, 2 , 4)~~ **F:4**
				~~1 2 3 4 5~~	~~(1, 1 , 2)~~; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); ~~(2, 2 , 4)~~; (2, 3, 5) **F:5**
		~~1 2 3~~			~~(1, 1 , 2)~~; (1, 2 , 3) **F:3**
				~~1 2 3 4 5~~	~~(1, 1 , 2)~~ ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
			~~1 2 3 4~~		~~(1, 1 , 2)~~ ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **F:4**
				~~1 2 3 4 5~~	~~(1, 1 , 2)~~ ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
1					
				~~1 2 3 4 5~~	(1, 1 , 2) ; (1, 2 , 3) ; ~~(1, 3 , 4) ; (1, 4, 5);~~ (2, 2 , 4); (2, 3, 5) **F:5**
			~~1 2 3 4~~		(1, 1 , 2) ; (1, 2 , 3) ; ~~(1, 3 , 4)~~; (2, 2 , 4) **F:4**
				~~1 2 3 4 5~~	(1, 1 , 2) ; (1, 2 , 3) ; ~~(1, 3 , 4)~~; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
		1 2 3			(1, 1 , 2) ; (1, 2 , 3) **T:3**
				~~1 2 3 4 5~~	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); ~~(2, 2 , 4)~~; (2, 3, 5) **F:5**
			~~1 2 3 4~~		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; ~~(2, 2 , 4)~~ **F:4**
				~~1 2 3 4 5~~	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); ~~(2, 2 , 4)~~; (2, 3, 5) **F:5**
	1 2				(1, 1 , 2) ; **T:2**
				~~1 2 3 4 5~~	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; ~~(1, 4, 5)~~; (2, 2 , 4); (2, 3, 5) **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **T:4**
				~~1 2 3 4 5~~	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); ~~(2, 3, 5)~~ **F:5**
		1 2 3			(1, 1 , 2) ; (1, 2 , 3) **T:3**
				~~1 2 3 4 5~~	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); ~~(2, 2 , 4)~~; (2, 3, 5) **F:5**
			~~1 2 3 4~~		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; ~~(2, 2 , 4)~~ **F:4**
				~~1 2 3 4 5~~	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5);~~(2, 2 , 4); (2, 3, 5)~~ **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2, 3) ; (1, 3, 4) ; ~~(2, 2, 4)~~ **F:4**
		1 2 3			(1, 1 , 2) ; (1, 2, 3) **T:3**
				~~1 2 3 4 5~~	(1, 1 , 2) ; (1, 2, 3) ; (1, 3, 4) ; (1, 4, 5) ; (2, 2, 4) ; ~~(2, 3, 5)~~ **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2, 3) ; (1, 3, 4) ; (2, 2, 4) **T:4**
				~~1 2 3 4 5~~	(1, 1 , 2) ; (1, 2, 3) ; (1, 3, 4) ; ~~(1, 4, 5)~~; (2, 2, 4) ; (2, 3, 5) **F:5**
	1 2				(1, 1 , 2) **T:2**
			~~1 2 3 4~~		(1, 1 , 2) ; (1, 2, 3) ; (1, 3, 4) ; ~~(2, 2, 4)~~ **F:4**
		1 2 3			(1, 1 , 2) ; (1, 2, 3); **T:3** (1, 1
			~~1 2 3 4~~		, 2) ; (1, 2, 3) ; ~~(1, 3, 4)~~; (2, 2, 4) **F:4**
1					
	~~1 2~~				~~(1, 1 , 2)~~ **F:2**

We can see that the triplet of the first colouring is monochromatic which makes the predicate unsatisfied (**F:2**). However, the predicate is satisfied by the second colouring (**T:2**). Thus, for **N=2**, we already found one colouring function that excludes monochromatic without having to consider the second line of the table where the first integer "1" is coloured in blue. We must now check for N > 2 until we find a value N for which no colouring function satisfy the predicate.

In the first line of the table, all the possible colouring starting with "1" in red, for N=2 to N=5 have been analysed. In the second column, and for each colouring, we enumerate all the possible triplet of the form (**a**, **b**, **a+b**). Then the monochromatic triplets are barred. The list of tripled is followed by the letter "**F**" (when the predicate is false for the corresponding colouring), or the letter "T" otherwise, followed with the value of **N**. All the colouring that do not satisfy the predicate are barred in the first column. As précised in the previous section, the colouring "12", for N=2 is not a valid one. Thus, in the table, all the colouring, for N>2, starting with the prefix "12" are not valid as well. This is trivial since the list of triplets associated with the colouring "12" is included in the list of triplets of any colouring of the N(>2) first numbers that starts with "12". Indeed, the monochromatic triplet (1, 1, 2) will be present in these colourings.

Using this observation, we applied the following rule for the second line of the table (i.e., "1" in blue for **N**=1). Any colouring of the first N numbers containing a prefix corresponding to an invalid colouring of the first N'<N numbers will be ignored.

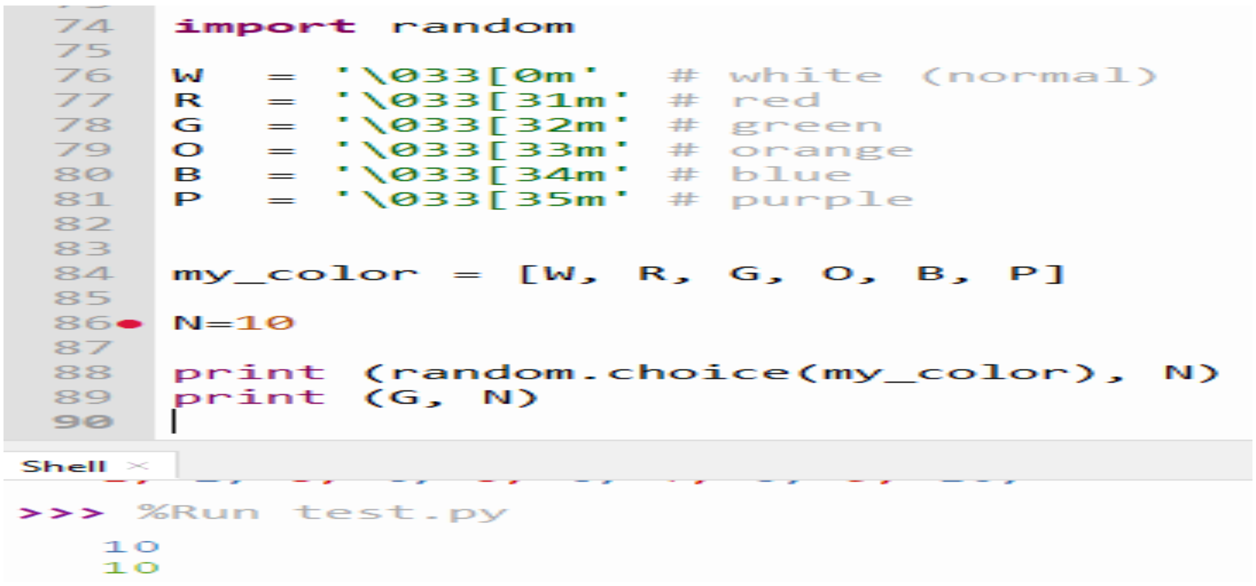
Finally, by analysing the couple Eval(Predicate):N associated with each colouring, we can see that we have **V:2,** two times, **V:3** appears four times, **V:4** appears two times and **V:5** **does not appear**. Thus **S(2) = 4.**

3.2. Warm-up exercises

1) Write a program that colours, and displays, the ***N*** first integer numbers with two colours s.t., the even numbers are coloured in red, and the odd numbers are coloured in blue.

Example: **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**, **...**

Example of Python code to display coloured text.



```
import random

W  = '\033[0m'   # white (normal)
R  = '\033[31m'  # red
G  = '\033[32m'  # green
O  = '\033[33m'  # orange
B  = '\033[34m'  # blue
P  = '\033[35m'  # purple


my_color = [W, R, G, O, B, P]

N=10

print (random.choice(my_color), N)
print (G, N)
```

Shell

```
>>> %Run test.py
    10
    10
```

2) Write a program that randomly colours the ***N*** first integer numbers using ***n*** (the ***n*** colours must be present).

> **Example**:
>
> For ***n=2***:
>
> $$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \ldots$$

3.3. Some basic functions

1) Write a program check if a given coloured triplet is monochromatic or not. The input of this program is a triplet and its colouring.

> **Example:**
>
> For ***n = 3***
>
> - The triplet (8, 5, 13) is monochromatic
> - The triplet (1, 9, 10) is not monochromatic

2) Write a program that generates all triplets ***(a,b,a+b)*** avec ***a***, ***b***, ***a+b*** for ***a***, ***b***, ***a+b*** ≤ p, for a given value p*.*

> **Example**:
>
> For p =4:
>
> The list of all the triplets ***(a, b, a+b)*** is:
>
> (1, 1, 2), (1, 2, 3), (1, 3, 4), (2, 2, 4).
>
> N.B. We ignore (2, 1, 3) et (3, 1, 4) since the order between ***a*** and ***b*** is not relevant.

3) Write a program that randomly colours ***N*** first integer numbers, for a given ***N***. Then, the program evaluates the predicate and displays the result i.e., it displays TRUE if all triplets are not monochromatic, and FALSE otherwise.

> **Example**:
>
> Let's consider the following colouring: 1, 2, 3, 4 for n=2 and N=4.
>
> The four corresponding triplets are (1, 1, 2), (1, 2, 3), (1, 3, 4), (2, 2, 4). None is monochromatic.

4) Write a program that colours the ***N*** first numbers with ***n*** colours, and checks whether the ***n*** colours are present in the colouring.

Example:

For ***n=2***, ***N=4***, the colouring 1, 2, 3, 4 is valid

For ***n=3***, ***N=4***, the colouring 1, 2, 3, 4 is not valid.

5) **TQ**: Who many different colourings are possible to colour ***N*** numbers with 2 colours?

6) **TQ**: Same question for ***n*** colours ?

7) Write a program that converts a decimal number to a binary one (e.g., 11→1011).

8) Generalise the previous program to convert a decimal number to a number in a base ***b*** (>1).

9) Write a program to store all the possible colourings of the ***N*** first integer numbers with ***n*** colours. The program then displays this list.

3.4. Application

1) Based on the previous programs and the problem description (see the table to illustrating ***S(2)***), write a program to compute ***S(2)***.

2) **TQ**: What is the value of ***S(2)*** ? Justify your answer.

3) Generalize the previous program to compute ***S(n)*** *for* ***n > 2***.

4) **TQ:** How many Schur's numbers were you able to calculate? What was the encountered problem ?.

5) Based on the following code, calculate the execution time to compute ***S(n)***.

```
import time

# Start counting of time
start_time = time.time()

# Put your code here …

# Displays the execution time
print("Execution time: %s seconds ---" % (time.time() -
start_time))
```

4. Advanced functions

5. There exists a theoreticla results stating that for any ***n >= 6***, it is possible to bound ***S(n)*** as follows:

$$\frac{3^n - 1}{2} \leq S(n) \leq 3 \times n! - 1$$

Write a program that displays the interval to which belongs ***S(n)*** for ***n*** (***n >= 6)***.

1) For the most motivated/advanced: Rewrite the program that computes ***S(n)*** using functions.

6. General instructions

1) The project is to be done in pairs. A single group of three students could be authorized if the number of students in the group is odd.

2) You will have to upload on the **moodle** the following:
   - A .zip file containing all the developed Python programs (one file per program)
   - A .pdf file (10 to 15 pages) describing your solution and data structure choice (use algorithm language to describe the main functionalities).

3) The uploaded file must be named as follows: **NAME1_NAME2** (where NAME1 and NAM2 are the names of the two students of the group).

4) The deadline to upload your work is **11/07/2019 at 23:59**

7. Planning

- Wednesday 10/09/2019: Subject is available on the **moodle**
- Week 10/14/2019: Lab session dedicated to the project
- 12/04/2019: Project defence for INT1
- 11/21/2019: Project defence for INT2 and INT3

8. Evaluation

The final mark will be composed of:

1) The source
2) The report
3) The defence