



## **Project Report:Image processing**

Presented by:

Ferhan Taha

Prof. Saadi

### **Summary:**

- 1)Thanks**
- 2)About this report**
- 3)Introduction**
- 4)Documentation**
- 5)Functions and Results**
- 6)Instructions for using the program**
- 7)Assessment and difficulties**

### **1) Acknowledgments:**

I would like to thank my teachers for the good training, which allowed me to develop my knowledge, of all kinds, Classes, practical work, books... This challenge helped me a lot to seek more and broaden my knowledge do in this programming language Python but also in others because even if the syntax changes but the algorithm with which we write remains the same, that's why I thank **Pr.Saadi** for this project.

### **2) About this report:**

This report covers the usefulness and instructions for use of the program, by dividing the program into well-explained functions with the results found, then the documentation of the modules and

the installation of these, at the end it covers the difficulties encountered during this day.

### **3) Introduction:**

Nowadays, no one can live without encountering an image or video such as advertising, SMS, scanner photo, security camera.... For this we ask ourselves how this image that we see works, where it comes from, how can we modify it and do all the possible treatments to make it functional for our needs. This means defining a digital image first.

A digital image can be interpreted in the form of pixels, each pixel represents an element of the image matrix, it is a plane projection of a 3D scene and it is divided into 3 types: color image, gray scale image, binary image that contains only white and black.

The goal of this project is to become familiar with the different processing of an image with the use of manipulation modules, image display and matrices. All of this will help us understand how images work based on matrices, and by simply doing basic algebra theory operations on them will finally make what we are studying make sense and visualize all of this to ultimately a practical result that helps us progress in our journey.

### **4) Documentation**

Before starting any project, there must be tools to work with and to help us make our tasks easy. As this project must be programmed by the Python programming language, then the documentation of all the source code, its modules and its installation saves a lot of time for the user to search each time for the operation of the program.

#### **i. Installing Python:**

- First let's install Python from the official site "<https://www.python.org/downloads/>". In the following we use the Windows operating system.

- The following installation of Python in the system is easy, you have to follow the installation steps like any program.

#### **ii. Installing the modules:**

- Here the installation of the modules (libraries) which contain the predefined functions we need, is better to do it using the command prompt.

- Go to the Windows search bar, and search for 'cmd', or 'Command Prompt'. When the window opens, it shows you the path of your user.

```
Invite de commandes
Microsoft Windows [version 10.0.22000.1219]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\carin>
```

- If you have installed python in another path, it is better to access this, by the commands 'cd / cd ..', for example if I want to access my users' documents we write 'cd Documents' and I find the path.

```
Invite de commandes
Microsoft Windows [version 10.0.22000.1219]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\carin>cd Documents

C:\Users\carin\Documents>
```

And if I want to return to the parent file here 'carin' we write 'cd..'.

```
C:\Users\carin\Documents>cd ..

C:\Users\carin>
```

- In the Windows command prompt browse the installation path of the python file. To install the modules you simply have to follow this syntax 'py -m pip install 'the name of the module"' and press enter (new line).

- For example the two modules we need in this project are 'numpy' for matrix manipulation and 'matplotlib' for image display and transformations. So we proceed by writing in the command prompt 'py -m pip install numpy' as we see below

```
C:\Users\carin>py -m pip install numpy
Requirement already satisfied: numpy in c:\users\carin\appdata\local\programs\python\python310\lib\site-packages (1.23.5)
```

They tell us that the installation request is satisfied, as we already have this module installed, however if you do not have it, everything will be done as normal and installed. Do the same with matplotlib or any necessary module.

### iii. The Compiler

- Everyone wants to work in a professional and comfortable workspace,

so to facilitate our tasks we recommend installing a compiler and an efficient interface. Recommendation: Visual Studio Code, Clion. However, we can work with the classic IDLE of python.

#### **iv. Modules:**

##### **- Random:**

This module generates and manipulates numbers by returning random numbers. We need in this module:

- `randrange(value)`: this function returns a number between 0 by default and the value entered as a parameter. You can, however, enter the limits of the desired values.

##### **- Matplotlib:**

In this module we need `pyplot`, which is used to display, open images, save them... All these manipulations help us in graph theory to display functions or any form of numerical values. So the methods or functions predefined in this object that we need are:

- `axis(option)`: this method activates or deactivates the axes of the graph or image wanted to display by entering as an option in parameter "on" to activate and "off" to deactivate the axes (x,y for example).
- `imshow(matrix, cmap)`: this function keeps an image in the form of a matrix with an optional color map and entered as a parameter (eg `gray = gray`).
- `show()`: this function opens a window containing an image which is maintained by the 'imshow()' function.
- `imread(image)`: this method reads an image entered as a parameter in the form of a path and returns an array.
- `imsave(path, matrix, cmap)`: this method saves a matrix as an image in the path entered as a parameter with a color entered in the last parameter.

##### **- Numpy:**

This module manages matrices with predefined functions which makes complex operations easy to process. The functions used by this module in the program are:

- `array(list)`: this function takes a list as a parameter and renders it into an array type array, you can add a second parameter of the type of variables you want stored (integers, reals, etc.)
- `array(list).astype(type)`: 'astype()' a method of the numpy module which

conditions and transforms the values of an array to the type entered as a parameter.

- `zeros(dimensions)`: this method initializes and returns a dimension matrix entered as a parameter in the form of a tuple (rows, columns).

-`shape`: this method returns the dimension of a matrix.

- `full(dimension, value)`: this function initializes and returns a dimension matrix entered as parameters with its elements which are equal to the value entered as the second parameter.
- `tolist()`: this method takes nothing as an argument, and transforms an array into a list.

5) To test our functions during programming we must have an image to manipulate it for this we use the gray scale image below:



## I) Input and output operations on images:

(1) (2) .... designate the numbering of the lines of the source code in image.

### 1) Show image:

```
4 def AfficherImg(img):  
5     plt.axis("off") #les axes du graphe ici désactivés  
6     #plt.imshow(img, interpolation = 'nearest') # affichage de l'img  
7     plt.imshow(img, cmap = "gray") # "gray" pour afficher les images en echelle grise  
8     plt.show() #pour voir l'image dans une fenetre ou à l'interface d'execution
```

- (4) This function displays a window containing an image. It takes a matrix as parameter.
- (5) We need to display an image not a graph for this we deactivate the axes using the `axis("off")` function.
- (7) We maintain the matrix in the form of an image with a gray color. (8) To finally display it.

**Result:**



## 2) Open the image:

```
10 def ouvrirImage(chemin):
11     return plt.imread(chemin)
```

- (11) Not much for this function which returns an array of numpy module by the imread function taking the path of the image as a parameter.

## 3) Save the image:

```
13 def sauImage(img):
14     chemin = str(input("Entrer le nom de votre nouvelle photo sans oublier son format : "))
15     plt.imsave(chemin, img, cmap = 'gray') # enregistrer l'image dans le chemin voulu avec
```

- (13) This function takes a matrix as a parameter and saves an image in the path (14) entered and requested by the user. (15) Finally the path obtained is assigned to the path of the imsave function with a gray color.

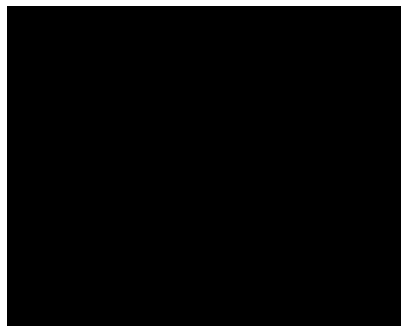
## II) White and black images:

### 1) Creating a black image

```
17 def image_noire(h, l):
18     M = [[0]*l for i in range(h)] #itérons et créant une matrice à deux dimensions de h lignes et l colonnes juste de 0
19     c = np.array(M).astype(np.uint8) # transformant la matrice initialisé en array comme type de 8 bits non int32
20     return c
```

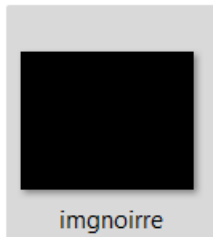
- (17) This function returns an array of which all its values are 0. It takes as parameters the dimensions of the desired matrix returned. (18) We initialize a list of "l" columns and "h" rows using the traditional list initialization method.
- (19) Then we assign this list to a new variable but transform it into an array and an 8-byte type.
- We used astype(np.uint8) in all the following functions because we encountered a problem when saving the image
- `ValueError: Image RGB array must be uint8 or floating point; found int32`

## Result:



Propriétés de : imgnoirre

Général		Sécurité	Détails	Versions précédentes
Propriété	Valeur			
<b>Images</b>				
ID de l'image				
Dimensions	250 x 200			
Largeur	250 pixels			
Hauteur	200 pixels			
Résolution horizontale	100 ppp			
Résolution verticale	100 ppp			
Profondeur de couleur	24			



- As we see we have created an image of 200 rows and 250 columns, the properties of this created image affirm this.

-In all the functions that follow, we just display the desired result.

## 2) Creating a white image:

```
22 def image_blanche(h, l):
23     M = [[1]*l for i in range(h)] #itèrons et créant une matrice à deux dimensions de h lignes et l colonnes juste de 1
24     c = np.array(M).astype(np.uint8)
25     return c
```

- Same logic as the function which creates a black image except here the array values returned are all 1s.

**Result:**



## 3) Creating a white and black image:

```
27 def creerImgBlancNoir(h, l):
28     M = [[0]*l for i in range(h)] #initialisons une matrice de h lignes et l colonnes
29     for i in range(h):
30         for j in range(l):
31             M[i][j] = (i + j + 2)%2 # chaque element reçoit l'indice de colonne et de ligne '+2' car i et j debutent de 0, cette somme modulo 2
32     c = np.array(M).astype(np.uint8)
33     return c
```

- (27) This function returns an array whose values are 1s and 0s. It receives the matrix dimension as parameters.
- (31) For filling the matrix the algorithm introduced and that each element is equal to the sum of the index of its row and its column plus 2 because we start with the indices 0, finally the element receives this sum module 2.

### Result:



Images	
ID de l'image	
Dimensions	500 x 500
Largeur	500 pixels
Hauteur	500 pixels

### 4) Negative of an image:

```

35 def negatif(img):
36     img = img.tolist()
37     for i in range(len(img)):
38         for j in range(len(img[0])):
39             if img[i][j] == 0: #verifions si un element egal à 0
40                 img[i][j] = 1 #pour qu'il reçoit 1
41             elif img[i][j] == 1: #ici le contraire si 1 l'element devient 0
42                 img[i][j] = 0
43     c = np.array(img).astype(np.uint8)
44     return c

```

-This function returns an array in which the 1s of this array become 0 and vice versa.

- (36) We transform the array as a parameter into a list by introducing the .tolist() method so as not to receive any errors by trying to iterate as an array.
- (39) Simple logic of condition if an element equal to 1 becomes 0 and if is a 0 becomes 1
- To use simple iteration algorithms for arrays, you must transform the numpy array into a list because otherwise you encounter a problem

```

if img[i][j] == 0: #verifions si un element egal à 0
ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()

```

### Result:

- Almost the same as the previous image because there are a lot of pixels not a big difference.



### III) Gray scale images:

#### 1) Luminance:

```
46 def luminance(img):
47     s = 0
48     ind = len(img) * len(img[0]) #Le nombre d'elements dans la matrice img
49     for i in range(len(img)):
50         for j in range(len(img[0])):
51             s += img[i][j] #sommant toute les valeurs dans la matrice img
52     return s / ind #retournant la moyenne des valeurs de cette matrice
```

- This function returns the average of the values of a matrix.
- (48) Here we find the number of elements of the matrix.
- (51) Summing up all the elements (52) Returning them at the end.

#### Result:

-The luminance of the white and black image from the previous function returns 0.000204

```
- [0.000204 0.000204 0.000204]
```

#### 2) Contrast:

```
54 def contrast(img):
55     N = len(img) * len(img[0]) #Le nombre d'elements dans la matrice img
56     moy = luminance(img) #utilisons la fonction luminance() pour retrouver la moyenne de la matrice img
57     s = 0
58     for i in range(len(img)):
59         for j in range(len(img[0])):
60             s += (img[i][j] - moy)**2 # chaque pixel ou element de la matrice se reduit par la moyenne le tout à la puissance 2 , on somme tous
61     return s / N #On divise ensuite cette somme par le nombre d'elements dans la matrice img
```

- This function returns the grayscale variance
- (56) Finding the average according to the luminance function.
- (60) This algorithm consists of summing the subtraction of each element of the matrix by squaring it.
- (61) We return this sum by dividing it by the number of elements of the matrix.

#### Result:

- The Contrast of the white and black image from the previous function returns 32113.87 as a value

```
- [32113.87944961 32113.87944961 32113.87944961]
```

#### 3) Depth:

```

63 def profondeur(img):
64     max = img[0][0] #affectons le premier element au maximum
65     for i in range(len(img)):
66         for j in range(len(img[0])):
67             if max < img[i][j]: #verifions si le max est plus petit que le nombre suivant
68                 max = img[i][j] # si oui on affecte au max cet element
69     return max

```

- (63) This function returns the maximum pixel of the image, that is to say the maximum value of the image matrix.
- (64) By storing the first element in the max variable
- (67) Let's check if the following element is greater than max, if yes we assign this element to max, if not we continue the iteration.

#### IV) Images in gray mode:

##### 1) Reverse image:

```

74 def inverser(img):
75     return 255 - img

```

- (74) This function returns the inverse of an image, as each pixel in gray mode varies between 0 and 255 so if we need the inverse of a pixel (75) we replace this pixel with 255. As here the image parameter is in the form of a numpy array so we can subtract with a simple subtraction operation all the elements of the matrix by 255.

#### Result:



##### 2) Flip Horizontally:

```

77 def flipH(img):
78     nouv = img.tolist() #.tolist() transforme un array en liste, la fonction list() ne marche pas elle retourne une liste d'arrays
79     for i in range(len(img)):
80         for j in range(len(img[0])):
81             nouv[i][j] = img[-(i+1)][-(j+1)] #renversant la matrice img dans une nouvelle matrice
82     c = np.array(nouv).astype(np.uint8)
83     return c

```

- This function flips the image horizontally
- (78) We transform the numpy array into a list and store it in "new".

- (81) Then let's reverse the entire matrix, reversing all the columns (horizontally).
- (82) Transforming this new list back into numpy array type with 8 bytes because otherwise the returned image will not be processed as desired.

**Result:**



### 3) Install Vertically and Horizontally:

```

86 def poserV(img1, img2):
87     M = list(img1) + list(img2) #on ajoute les lignes de la matrice img2 en dessous de la matrice img1 pour former une matrice M
88     c = np.array(M) #ensuite on affecte à c la matrice M en mode array
89     return c
90 def poserH(img1, img2):
91     nouv = [[0]*(2*len(img1[0])) for i in range(len(img1))] #creant une matrice de deux dimensions pour colonne le double de nombre de colonne
92     img1, img2 = img1.tolist(), img2.tolist() #transformons les matrice img1 et img2 en liste
93     for i in range(len(img1)):
94         nouv[i] = img1[i] + img2[i] # les sous listes ici se concatène dans une sous liste
95     c = np.array(nouv).astype(np.uint8)
96     return c

```

- (86),(90) These two functions allow you to return two images, one placed on top of the other and the second, one next to the other.
- (87) The operation of addition in lists allows you to add one list to another to find another, for example if we have [1,2,3] + [4, 5, 6] it will be [ 1,2,3,4,5,6]. So here we transform the array into a list to perform this task, we add the first image as a matrix to the second. At the end we obtain columns below the columns of the first matrix.
- (88) Transforming this list into an array.
- (91) Let's initialize a matrix with a dimension of the same rows as the two images, but with double the columns because we want to place the second image next to the other. Transforming the two arrays into lists.
- (94) Let's iterate in each line and adding to the "new" the combination of the first line and the second line as explained previously it becomes a single line.
- (95) Transforming this new "new" list into an 8-bit array by returning it.

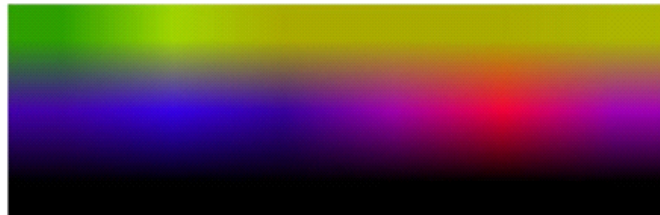
**Result:**



## V) RGB Images:

1 -

```
>>>M=[[[210, 100, 255],[100, 50, 255],[90, 90, 255],[90, 90, 255],[90, 90, 255],[90, 80, 255]],
[[190, 255,89],[ 201, 255,29],[200, 255,100],[100, 255,90],[20, 255,200], [100, 255,80]],
[[255,0, 0],[ 255,0, 0],[255,0, 0],[255,0, 0],[255,0, 0], [255,0, 0]] ]
>>>import matplotlib.pyplot as plt
>>>plt.imshow(M)
>>>plt.axis ("off")
>>>plt.show ()
```



This matrix returns this image by the `imshow()` function of the `matplotlib.pyplot` module, to access the value of these elements we can proceed through the indexes of the matrix.

```
5 print(M[0][1][1])
6 print(M[1][0][1])
7 print(M[2][1][0])
```

```
PS C:\Users\carin\AppData\Local\Temp> python -u "C:\Users\carin\AppData\Local\Temp\tempCodeRunnerFile.python"
50
255
255
```

2- The amount of memory needed in bytes to store an array representing an RGB image is 3 bytes, because RGB is composed of three colors (RED, GREEN, BLUE); with 1 bit has two possibilities 1 and 0; 2 bits has 4 colors, 8 bits has 256 colors this is our case, 8 bits -> 1 bytes, 3 elements, 24 bits then 3 bytes.

## 3 - Initialization of a random image in RGB:

```
103 def initImageRGB():
104     n = rd.randrange(256) #lignes aléatoires
105     p = rd.randrange(256) #colonnes aléatoires
106     imageRGB = np.zeros((n, p, 3)) #on initialise une matrice de trois dimensions avec n lignes et p colonnes
107     for i in range(n):
108         for j in range(p):
109             for k in range(3):
110                 imageRGB[i][j][k] = rd.randrange(256) #chaque element prend une valeur aléatoire comprise entre 0 et 255
111     return imageRGB
```

- (103) This function initializes a random image of n rows and p columns with randomly generated elements.
- (104), (105) Finding a random number between 0 and 256 and assigning it to rows and columns with the "randrange()" function.
- (106) Let's initialize a matrix of dimension (n, p, 3) "3" Because we have RGB.
- (110) Let's iterate and assign each element a random value.

**Result:**



#### 4- Vertical Symmetry:

```
120 def symetrieV(img):
121     nouv = np.full(img.shape, 0) # on initialise une matrice de meme dimension que img, on peut meme utiliser np.zeros(img.shape)
122     for i in range(len(img)):
123         nouv[i] = img[-i-1] # on inverse Les lignes pour que La première ligne soit la dernière
124     c = np.array(nouv).astype(np.uint8)
125     return c
```

- (120) This function returns the vertical symmetry of an image.
- (121) Let's initialize a matrix with "0" using the "full()" method of the numpy module which takes as argument the dimension of the image and the desired values. We can use the zeros() function but it's good to open up to other functions.
- (123) Let's iterate and invert the image values into a new matrix.
- (125) Finally returning this matrix as an 8-bit type.

**Result:**



#### 5- Gray Scale:

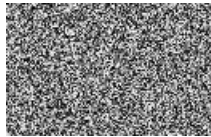
```

113 def grayscale(imageRGB):
114     M = np.zeros((len(imageRGB), len(imageRGB[0]))) #initialisons une matrice de meme dimension que imageRGB
115     for i in range(len(imageRGB)):
116         for j in range(len(imageRGB[0])):
117             M[i][j] = (max(imageRGB[i][j]) + min(imageRGB[i][j])) // 2 #chaque element on le remplace par le max et le min valeur entière
118     return M

```

- (113) This function transforms an RGB image to a gray scale image.
- (114) Let's initialize a matrix of the same dimensions as the image.
- (117) Let's iterate and assign to each element the sum of the max and min of each sublist, with the integer value of this sum.

**Result:**



## 6) How to use the program:

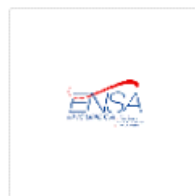
```

230 if __name__ == '__main__':
231     chemin = str(input("Veuillez saisir le nom de l'image que vous voulez traiter sans oubliant le format à la fin :"))
232     while not "." in chemin: # si l'utilisateur n'entre pas le format
233         chemin = str(input("Vous avez oublier le format d'image veuillez resaisir :"))# il doit resaisir le chemin
234     img = ouvrirImage(chemin)
235     main()

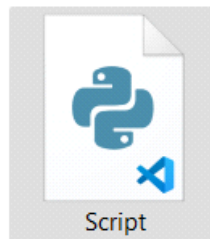
```

- (230) In the main program, (231) the user must enter the name of the image in the folder or the path of the image, if he forgets to enter the format of the image (232) the while loop keeps asking it to re-enter the name.
- (234) Finally we assign to the variable "img" the matrix opened by the function "openImage"
- (235) Let's declare the "main()" function which contains the possible choices.

- First launch the "Image processing" application or just the python script:

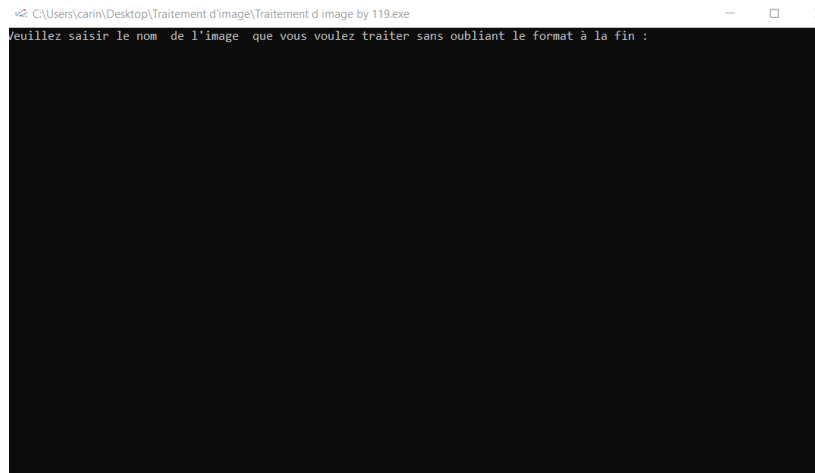


Traitement d  
image by 119

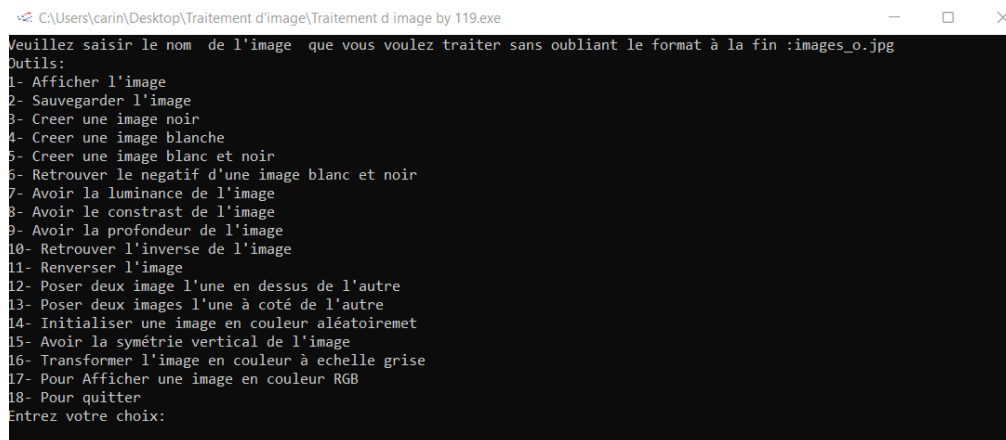


Script

- Secondly enter the image you want to modify even if you do not need to modify an image enter one for the program to launch. Without forgetting its format.



- Third, the program displays you a list of choices.



- Enter your choice. If you have entered a choice of creation or modification, the program finally asks you to save your image which you will find in your folder.



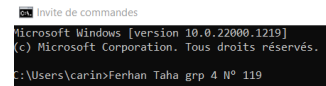
## 7) Assessment and Difficulties:

Image processing is not much difficult if we understand the science of images, there is more to discover in this field, but these manipulations give us a summary of how images work.

The difficulties encountered are basic in documenting functions and understanding digital images. As the "matplotlib" module is new and is not taught in the course, we had to look for the documentation of the functions of this module and understand how they work. Regarding images, we searched the web to better understand the theory of digital images.

Finally, this project helped us to deepen our knowledge and research skills to progress well in our next projects.

## Credits:



```
invite de commandes
Microsoft Windows [version 10.0.22000.1219]
(c) Microsoft Corporation. Tous droits réservés.
C:\Users\carin>ferhan Taha grp 4 N° 119
```





