BENHIMA Mehdi
SALH Hamza
Date du TP (30/11/2021)

Rapport de TP2 - Lecture automatique de chiffres par analyse d'image :

I. Introduction :

Le TP n°2 se focalise sur la représentation des images numériques et la reconnaissance automatique de caractères. Pour cela, on utilisera plusieurs méthodes dont la binarisation, la localisation et l'adaptation de la taille au modèle ainsi que la mesure de ressemblance par corrélation et la décision pour pouvoir finalement garantir une bonne lecture automatique de chiffre par analyse d'image.

III - Travail préparatoire :

1. Question (1) :

```
class Image:
    def __init__(self):
        """Initialisation d'une image composee d'un tableau numpy 2D vide
        (pixels) et de 2 dimensions (H = height et W = width) mises a 0
        """
        self.pixels = None
        self.H = 0
        self.W = 0
```

v

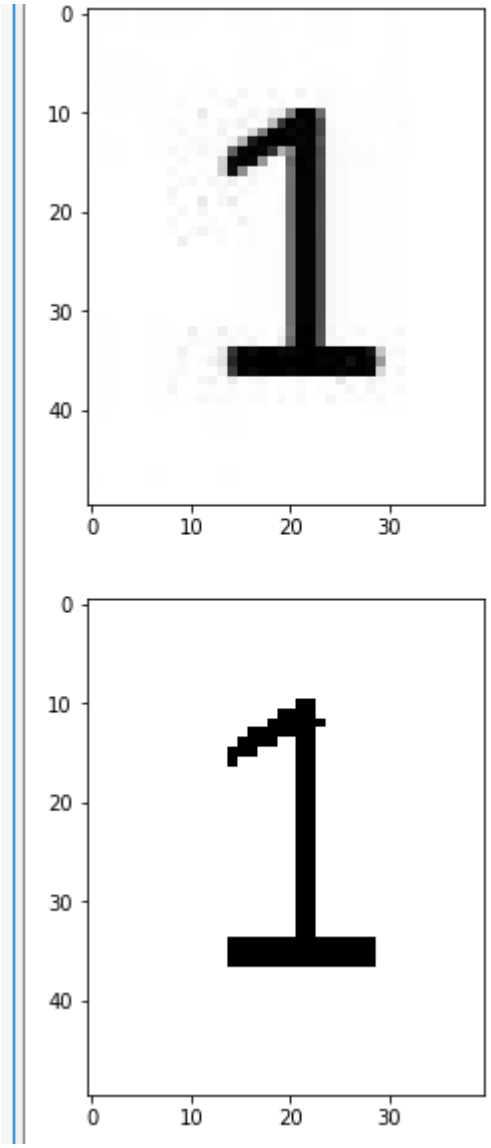
La classe Image se compose des attributs H, W qui correspondent respectivement à la hauteur et la largeur du tableau 2D représentant l'image, ainsi que l'attribut **pixels** qui contient un tableau 2D numpy contenant les valeurs de l'image en pratique.

2. Question (2) :

Le but de cette question est de réduire le nombre de valeurs possibles pour un pixel.
Pour cela on doit comparer chaque pixel à une valeur seuil que l'on fixera. Si le pixel dépasse cette valeur, alors on lui donne la valeur 255 qui représente le blanc, sinon il prendra la valeur 0 qui représente le noir.

```
#=============================================================================
def binarisation(self, S):
    # creation d'une image vide
    im_bin = Image()

    # affectation a l'image im_bin d'un tableau de pixels de meme taille
    # que self dont les intensites, de type uint8 (8bits non signes),
    # sont mises a 0
    im_bin.set_pixels(np.zeros((self.H, self.W), dtype=np.uint8))
    for i in range (self.H):
        for j in range (self.W):
            if self.pixels[i][j]<=S:
                im_bin.pixels[i][j]=0
            else:
                im_bin.pixels[i][j]=255
    return im_bin
```

Résultat obtenu après binarisation :



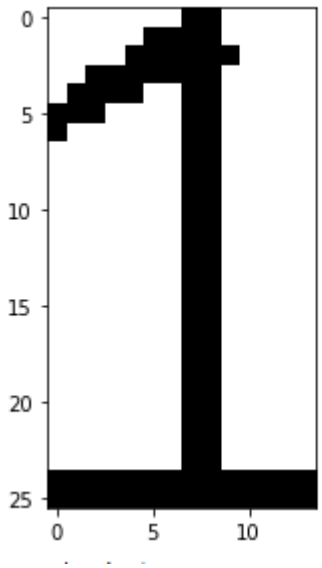
Question 3 :

Le but de cette question est de recadrer l'image pour se limiter au rectangle qui englobe toutes « les cases noires » du tableau. Pour cela, On initialise l_min à H et c_min à W pour avoir les valeurs maximales possibles au début et en déduire les coordonnées minimales et de méme on on initialise l_max et c_max à 0. On fait ensuite une double boucle for pour pouvoir parcourir toutes les cases de l'image de base et on s'intéresse uniquement aux endroits où la couleur est noire (if(self.pixels[i][j]==0.

Et on effectue à chaque fois dans la boucle des instructions if pour vérifier la condition et on prendra les valeurs finales de c_max, c_min, l_max, l_min, et on les affectera à la sortie Image1.

```
def localisation(self):
    l_max=0
    l_min=self.H
    c_max=0
    c_min=self.W
    for i in range(self.H):
        for j in range (self.W):
            if (self.pixels[i][j]==0):
                if i<=l_min:
                    l_min=i
                elif i>=l_max:
                    l_max=i
                if j<=c_min:
                    c_min=j
                elif j>=c_max:
                    c_max=j
    Image1=Image()
    Image1.set_pixels(self.pixels[l_min:l_max,c_min:c_max])
    return Image1
```

Résultat obtenue après localisation :

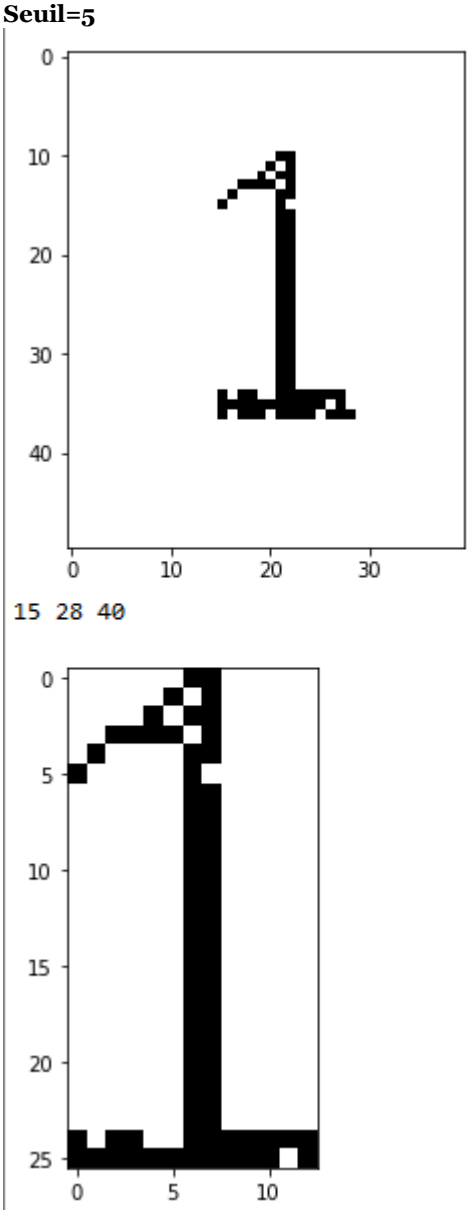


IV - Reconnaissance automatique de chiffre :¶

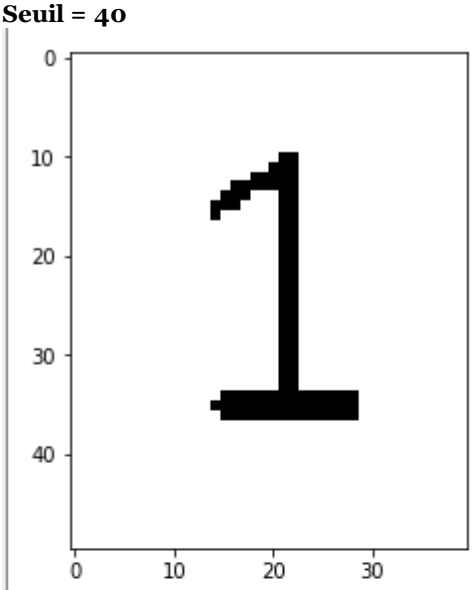
Question1 et 2 :

On fait des exemples avec le fichier main.py pour la binarisation et la localisation :
On essaie différente valeurs de seuil et on obtient le valeurs suivantes :

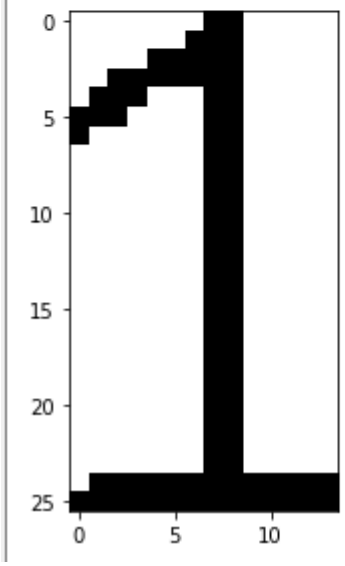
Seuil=5



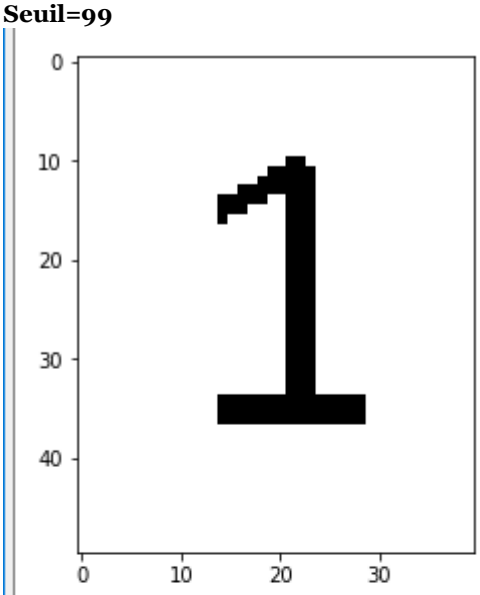
Seuil = 40



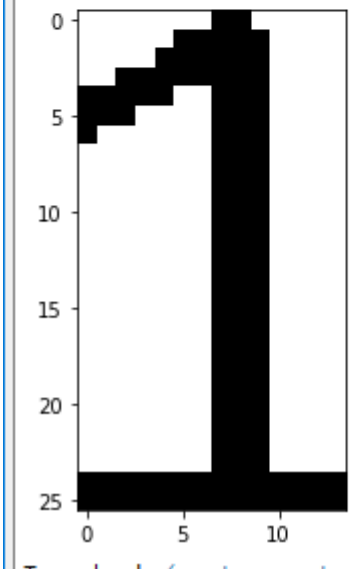

```
14 28 40
```



Seuil=99




```
14 28 40
```



Conclusioon :

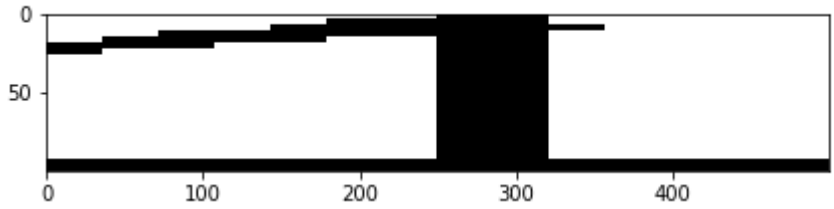
Bien evidemment, on remarque Que plus la valeur du seuil est importante, plus la forme noir est plus importante, et donc l'image plus claire.

Question 3 :

Le but de cette question est modifier les dimensions de l'image. On utilise donc la fonction resize de la librairie skimage, et on affectera à l'image de sortie les nouvelles dimensions.

```
def resize(self, new_H, new_W):
    image3=Image()
    n=resize(self.pixels,(new_H,new_W),0)
    image3.set_pixels(np.uint8(n*255))
    return image3
```

Résultat obtenu avec (100,500) :



Question 4 :

Le but de cette question est de comparer 2 images avec la méthode de similitude. Pour cela on crée une variable a qu'on initialise à 0, et on effectue 2 boucles for pour parcourir toutes les valeurs de l'image et les comparer avec les valeurs de l'autre image.

Si la valeur du pixel de même cordonnées des 2 images (if self.pixels[i][j]==im.pixels[i][j]:), on ajoute 1 à la valeur de a, sinon on n'ajoute rien, et on retourne la valeur de a finale divisé par le nombre totale d'essaies effectués.

```
    def similitude(self, im):

        a=0
        for i in range(self.H):
            for j in range(self.W):
                if self.pixels[i][j]==im.pixels[i][j]:
                    a+=1
                else:
                    a+=0
        return a/(self.H*self.W)
```

Question 5 :

Dans cette question, on écrira la fonction **reconnaissance_chiffre(image, liste_modeles, S)** qui va effectuer la reconnaisance de chiffre sur l'image **image** donnée en entrée de la fonction.

Pour cela, on utilisera les différentes méthodes déjà programmés, commençant par la binarisation, la localisation de l'mage et aussi le redimensionnement et la similitude.

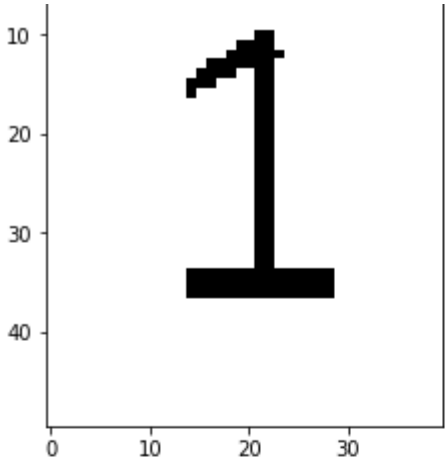
Le but de cette question est de comparer l'image fournie en entrée avec les différentes images fournies en « liste_modeles » et retourner l'indice qui correspond à l'image qui correspond le plus à l'image fournie en entrée, c-à-d l'image qui aura le plus grand score de similitude.

```python
from image import Image

def lecture_modeles(chemin_dossier):
    fichiers= ['_0.png','_1.png','_2.png','_3.png','_4.png','_5.png','_6.png',
            '_7.png','_8.png','_9.png']
    liste_modeles = []
    for fichier in fichiers:
        model = Image()
        model.load(chemin_dossier + fichier)
        liste_modeles.append(model)
    return liste_modeles


def reconnaissance_chiffre(image, liste_modeles, S):
    bin_im=image.binarisation(S)
    loc_im=bin_im.localisation()
    b=0
    indice=0
    for i in range(len(liste_modeles)):
        res=loc_im.resize(liste_modeles[i].H,liste_modeles[i].W)
        if res.similitude(liste_modeles[i])>b:
            b=res.similitude(liste_modeles[i])
            indice=i
    return indice


```

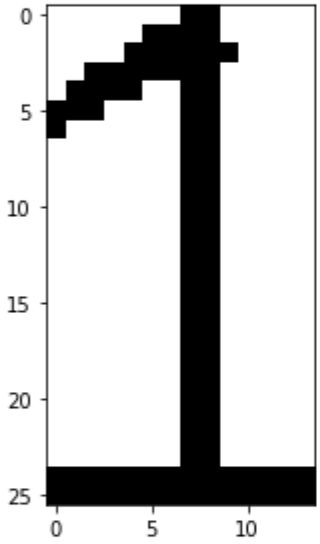
Question6 :

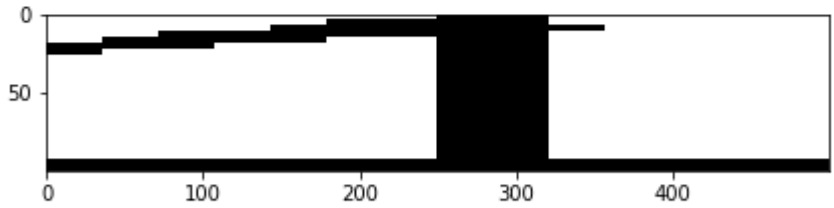
Dans cette question, On essayera la fonction de reconnaissance en modifiant l'image de test dans **main.py** avec différentes images disponibles dans **assets/** et en modifiant également le seuil avec quelques valeurs.

On effectuera plusieurs exemples et on obtiendra les résultats suivants :

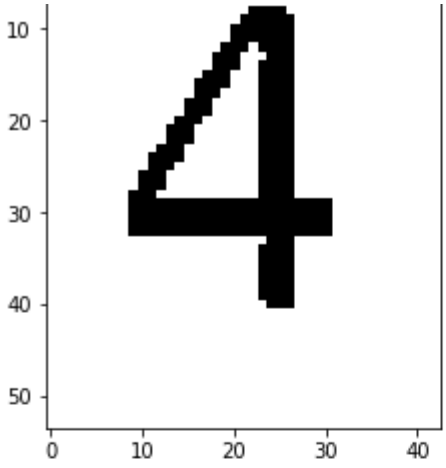



```
14 28 40
```

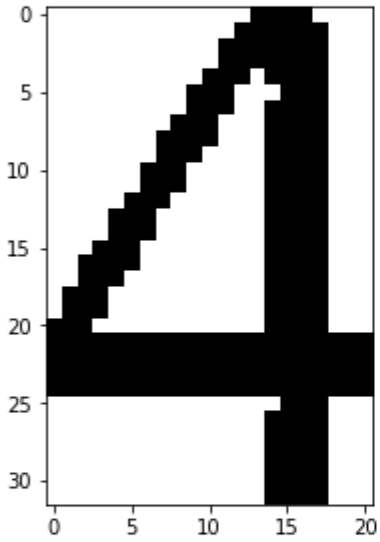


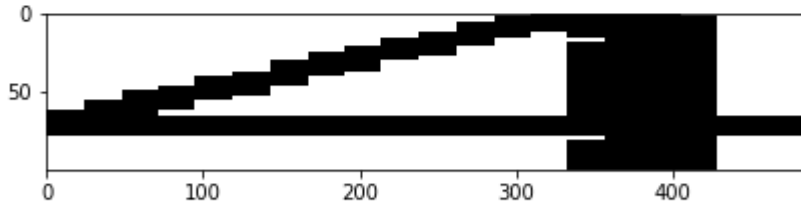



```
lecture image : ../assets/_0.png (32x22)
lecture image : ../assets/_1.png (32x18)
lecture image : ../assets/_2.png (32x20)
lecture image : ../assets/_3.png (32x20)
lecture image : ../assets/_4.png (32x24)
lecture image : ../assets/_5.png (32x20)
lecture image : ../assets/_6.png (32x21)
lecture image : ../assets/_7.png (32x21)
lecture image : ../assets/_8.png (32x22)
lecture image : ../assets/_9.png (32x22)
14 28 40
Le chiffre reconnu est :  1
```




```
) 30 43
```






```
lecture image : ../assets/_0.png (32x22)
lecture image : ../assets/_1.png (32x18)
lecture image : ../assets/_2.png (32x20)
lecture image : ../assets/_3.png (32x20)
lecture image : ../assets/_4.png (32x24)
lecture image : ../assets/_5.png (32x20)
lecture image : ../assets/_6.png (32x21)
lecture image : ../assets/_7.png (32x21)
lecture image : ../assets/_8.png (32x22)
lecture image : ../assets/_9.png (32x22)
) 30 43
.e chiffre reconnu est :  4
```

On remarque qu'avec une bonne valeur de seuil, le résultat de la fonction est toujours vrai. Par contre, avec une valeur de seuil très basse, la fonction peut donner des résultats fausses, car elle dans ce cas elle ne pourra pas faire la différence entre 2 nombres presque semblable à une valeur basse de seuil (1 et 4 par exemple).

Conclusion :

Dans ce TP, on a appris à programmer plusieurs méthodes qui permettent **Lecture automatique de chiffres par analyse d'image,** commençant par la binarisation, la localisation, le redimensionnement, la similitude, et finalement la reconnaissance qui utilise toutes les méthodes programmés le long du TP.

On a du mal au début à bien comprendre et utiliser la notion du représentation de l'image en un tableau 2D, et aussi dans la dernière fonction lors de l'appel des différentes méthodes précédentes.

Mais on pu gérer la situation et on a réussi à terminer toutes les fonctions, et par conséquent, on a pu lors de ce TP à bien comprendre et utiliser la représentation et l'analyse numérique des images et ses différentes applications.