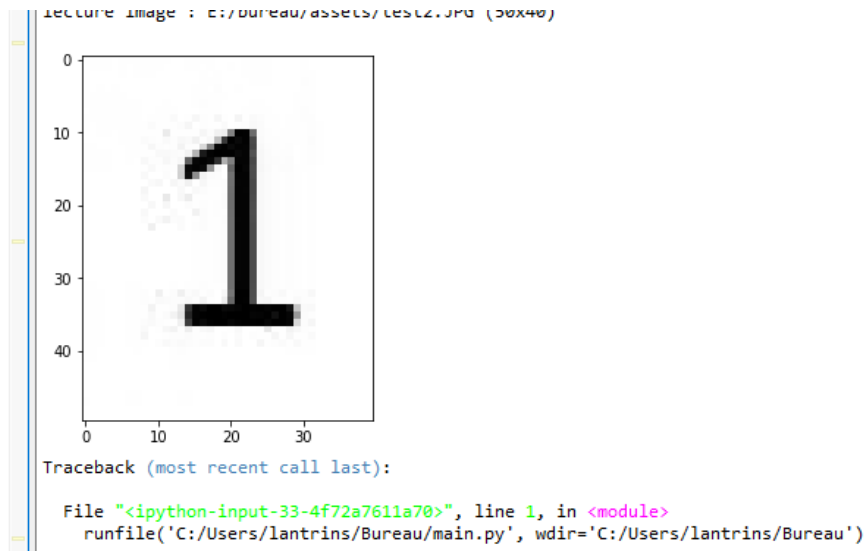


TP2 - Lecture automatique de chiffres par analyse d'image



III - Travail préparatoire¶

- 1) H et W correspondent à la taille de l'image, à chaque pixels correspond une valeur dans le tableau numpy soit 0 soit 255
- 2)

```
def binarisation(self, S):
    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))

    # TODO: boucle imbriquees pour parcourir tous les pixels de l'image im_bin
    # et calculer l'image binaire
    for i in range(self.H):
        for j in range(self.W):
            if self.pixels[i][j] >= S:
                im_bin.pixels[i][j] = 255
            else:
                im_bin.pixels[i][j] = 0

    return im_bin
```

On créer une image, que l'on initialise à 0 pour chaque pixels. Ensuite pour chaque pixel on test la couleur du pixel pour savoir si il est suffisamment noir garce a la valeur S qui est le seuil.

- 3)

```

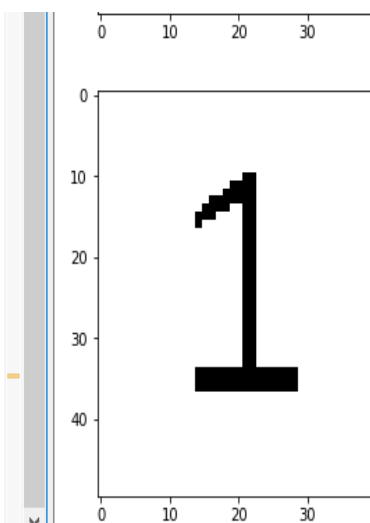
def localisation(self):
    lmax=0
    lmin=self.H
    cmax=0
    cmin=self.W
    for i in range(self.H):
        for j in range(self.W):
            if(self.pixels[i][j]==0):
                if i<lmin:
                    lmin=i
                if j<cmin:
                    cmin=j
                if i>lmin:
                    lmax=i
                if j>cmin:
                    cmax=j
    nbl=lmax-lmin
    nbc=cmax-cmin
    im_loc = Image()
    im_loc.set_pixels(np.zeros((nbl,nbc), dtype=np.uint8))
    for i in range(lmax-lmin):
        for j in range(cmax-cmin):
            im_loc.pixels[i][j]=self.pixels[lmin+i][cmin+j]
    return(im_loc)

```

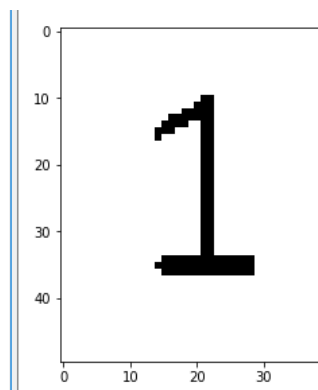
Dans cette fonction, on initialise les valeurs minimales aux dimensions de l'image et les maximales à 0. Puis pour chaque pixel on teste s'il est noir, si c'est le cas on change les valeurs max et min. Si le pixel noir est au-dessus de la valeur min on change la valeur, de même s'il est en dessous de la valeur max on change la valeur. Enfin on crée une image aux bonnes dimensions c'est-à-dire lmax-lmin et cmax-cmin et on copie l'ancienne image dans la nouvelle image.

IV - Reconnaissance automatique de chiffre

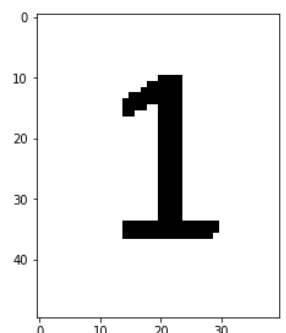
1)



Essai de binarisation pour S=70



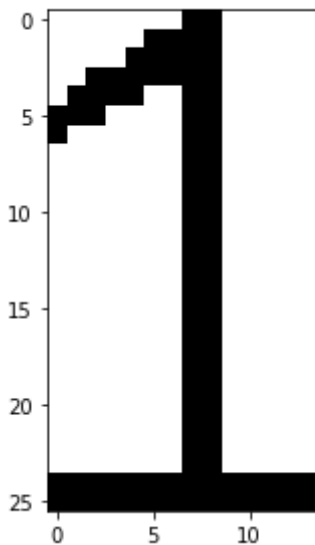
Essai de binarisation pour S=50



Essai de binarisation pour S=200

On n'a pas pu essayer avec test_image car nose2 n'était pas installé.

2)essaie localisation



3)

```
def resize(self, new_H, new_W):
    im_resized=resize(self.pixels, (new_H,new_W), 0)
    im_resized=np.uint8(im_resized*255)
    im_res = Image()
    im_res.set_pixels(np.zeros((new_H,new_W), dtype=np.uint8))
    for i in range(new_H):
        for j in range(new_W):
            im_res.pixels[i][j]=im_resized[i][j]

    return(im_res)
```

Dans cette fonction on crée un nouveau tableau aux bonnes dimensions, puis on utilise la fonction `uint8` pour passer des float en entier. Ensuite on crée une image qu'on initialise à 0 pour tous les pixels. Enfin on remplit l'image avec les valeurs du tableau.

4)

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

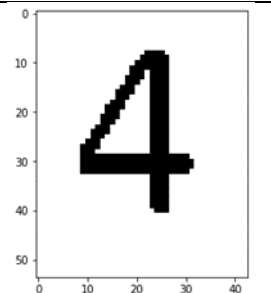
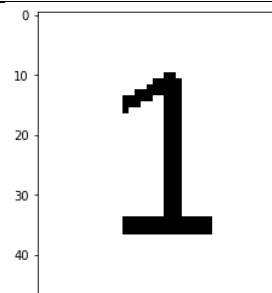
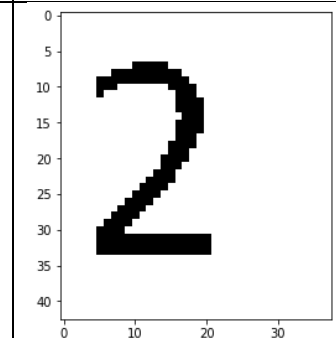
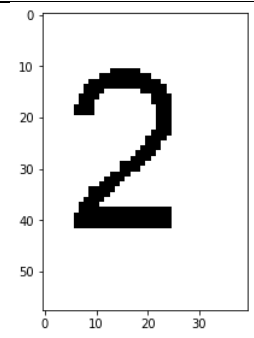
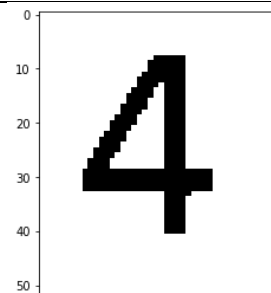
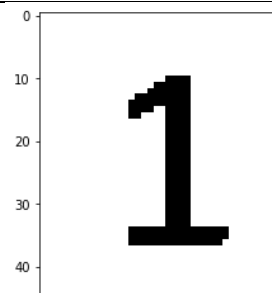
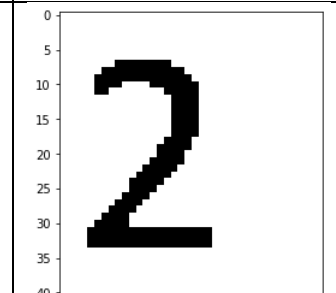
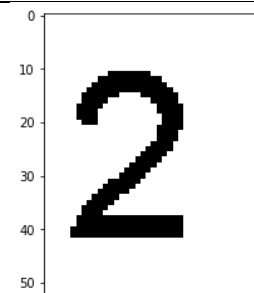
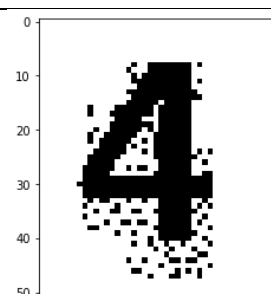
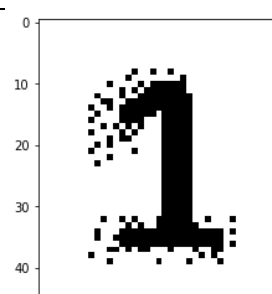
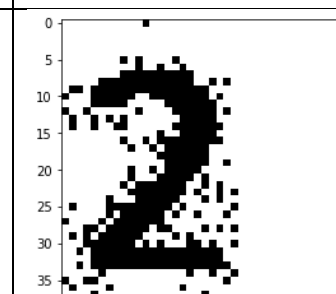
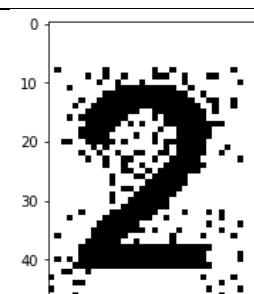
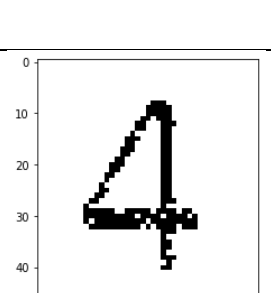
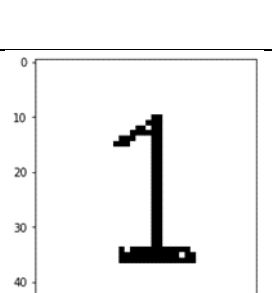
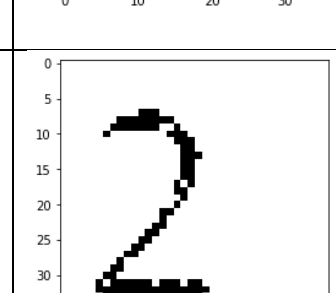
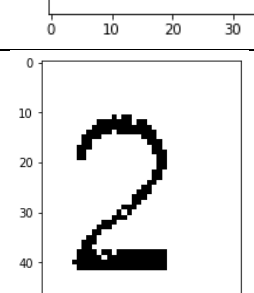
Dans cette fonction on initialise similitude à 0 puis on parcourt l'image en testant si l'image possède des pixels similaires. Si c'est le cas on ajoute 1 à similitude puis on renvoie similitude diviser par la surface de l'image pour avoir un nombre entre 0 et 1

5)

```
def reconnaissance_chiffre(image, liste_modeles, S):  
    image=image.binarisation(S)  
    image=image.localisation()  
  
    sim=0  
    chiffre=0  
    for i in range(len(liste_modeles)):  
        image=image.resize(liste_modeles[i].H,liste_modeles[i].W)  
        if image.similitude(liste_modeles[i])>sim:  
            sim=image.similitude(liste_modeles[i])  
            chiffre=i  
    return(chiffre)
```

Dans cette fonction utilise les différentes méthodes créées, d'abord binarisation puis localisation. Ensuite on parcourt les différents modèles, pour chaque modèle on créer une image à la bonne taille grâce a resize puis on test si l'image est similaire avec la méthode similitude. On ajoute mémorise le chiffre qui a le plus de similitude et on le retourne.

6)

Test1	Test2	Test3	Test4	s
				100
				200
				250
				10