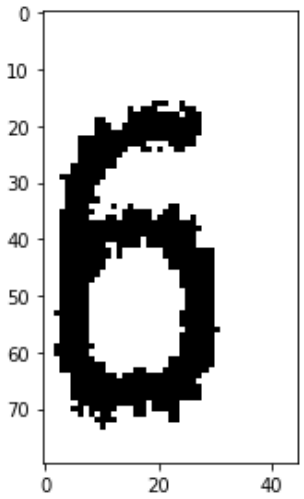
HAMADOU Nadia
DECOOPMAN Clement

Compte rendu

Fonction binaris :

```
def binaris(self, S):

    # preparaton du resultat : creation d'une image vide
    im_modif = image()
    # affectation de l'image resultat par un tableau de 0, de meme taille
    # que le tableau de pixels de l'image self
    # les valeurs sont de type uint8 (8bits non signes)
    im_modif.set_pixels(np.zeros((self.H,self.W), dtype=np.uint8))

    # boucle imbriquees pour parcourir tous les pixels de l'image
    for l in range(self.H):
       for c in range(self.W):
          # modif des pixels d'intensite >= a S
          if self.pixels[l][c] >= S:
             im_modif.pixels[l][c] = 255
          else :
             im_modif.pixels[l][c] = 0
    return im_modif
```

Image resultant : seuil de 150



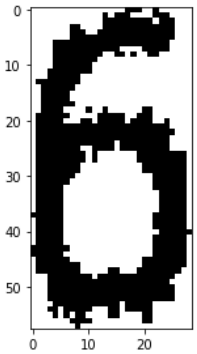

```
seuil = 150
im_bin = im.binaris(seuil)
im_bin.display("Image binarisee")
```

HAMADOU Nadia
DECOOPMAN Clement

Fonction localisation :

```
def localisation(self):

    #creation de la nouvelle image recadree
    im_loc = image()

    # declaration des coordonnees de localisation
    l_min = l_max = c_min = c_max = -10

    # determination des coordonnees min/max des lignes
    for l in range(self.H) :
       for c in range(self.W) :
          if self.pixels[l][c] == 0 :     # om tombe sur un pixel noir
             if(l_min == -10) :         # si la valeur n'a pas deja ete trouve, on note les coords
                l_min = l
             l_max = l                # on note les coords max jusqu'a ce qu'on ait plus que des pixels blancs

    # determination des coordonnee min/max des colonnes
    for c in range(self.W) :
       for l in range(self.H) :
          if self.pixels[l][c] == 0:
             if(c_min == -10) :
                c_min = c
             c_max = c

    #on affetce les pixels
    im_loc.set_pixels(self.pixels[l_min:l_max+1, c_min:c_max+1])

    print(l_min, l_max, c_min, c_max)
    return im_loc
```

Image resultant :



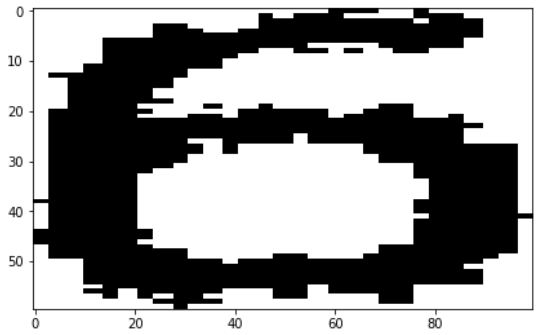

```
im_loc = im_bin.localisation()
im_loc.display("Localisation du chiffre")
```

HAMADOU Nadia
DECOOPMAN Clement

Fonction redimensionnement de l'image :

```
def resize_im(self,new_H,new_W) :

    # creation de la nouvelle image redimensionnee
    im_resized = image()
    # affectation des nouvelles dimensions
    im_resized.H = new_H
    im_resized.W = new_W

    # redimensionnement de l'image
    im_resized.pixels = resize(self.pixels, (new_H, new_W), 0)
    # normalisation sur l'intervalle de pixels [0, 255] et conversion en uint8
    im_resized.pixels = np.uint8(im_resized.pixels*255)

    return im_resized
```

Image resultant :




```
new_H = 60

new_W = 100

im_resized = im_loc.resize_im(new_H, new_W)

im_resized.display("Image redimensionnee")
```

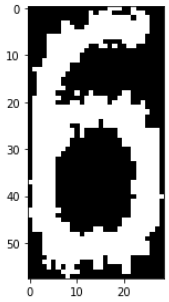
HAMADOU Nadia
DECOOPMAN Clement

Fonction de similitude :

```
def simil_im(self, im) :

    # Nombre de pixels correspondant entre les deux images
    pixels_simil = 0
    #Nombre total de pixels dans l'image de base
    px_total = self.H * self.W

    if self.H==im.H and self.W==im.W :
       for l in range(self.H) :
          for c in range(self.W) :
             if self.pixels[l][c] == im.pixels[l][c] :
                pixels_simil += 1

    else :
       print("Dimensions differentes")é


    return pixels_simil / px_total
```

Fonction de transformation d'une image en son image negative :

```
def negative(self) :

    #creation de l'image negative
    im_neg = image()
    im_neg.pixels = 255 - self.pixels

    return im_neg
```



Similitudes résultants : Rapport de similitudes entre l'image localisee et son image negative : 0.0

```
#creation de l'image negative de im_loc
im_loc_neg = im_loc.negative()
im_loc_neg.display("Image negative")


# test de similitudes

simi = im_loc.simil_im(im_loc_neg)

print("Rapport de similitudes entre l'image localisee et son image negative :", simi)
```

HAMADOU Nadia
DECOOPMAN Clement

Similitude maximum entre l'image de base traitée (binarisation et localisation) et la liste chargée d'images :

```
list_modif = [0]*len(list_model)    #liste des models traités
simil_max = 0                  #similitude maximum
id_im_max_simil = 0              #id de l'image avec la similitude maximum
new_im = image()                #nouvelle image


#Pour chaque model dans la liste, on traite et
#on compare les similitudes avec l'image de base (localisee)
for i in range( len(list_model) ) :
    new_im = list_model[i].resize_im(im_loc.H, im_loc.W)
    list_modif[i] = new_im
    s = im_loc.simil_im(new_im)

    if s > simil_max :
        simil_max = s
        id_im_max_simil = i


print("L'image la plus similaire est l'image n°", id_im_max_simil, " avec un rapport de " + str(simil_max))
im_loc.display("Image de base")
list_modif[id_im_max_simil].display("Image avec le plus de similitude ")
```

Resultats : L'image la plus similaire est l'image n° 6 avec un rapport de 0.7538644470868014