CAB

Algorithme

**Table des matières**

[​ Fonction : read\_barcode 2](#__RefHeading___Toc381_1557683862)

[​ Description : 2](#__RefHeading___Toc383_1557683862)

[​ Paramètres d’entrée : 2](#__RefHeading___Toc385_1557683862)

[​ Paramètres locaux : 2](#__RefHeading___Toc387_1557683862)

[​ Paramètre de sortie : 2](#__RefHeading___Toc389_1557683862)

[​ Début 2](#__RefHeading___Toc391_1557683862)

[​ Fonction : Valid\_barcode 4](#__RefHeading___Toc393_1557683862)

[​ Description : 4](#__RefHeading___Toc395_1557683862)

[​ Paramètres d’entrée : 4](#__RefHeading___Toc397_1557683862)

[​ Paramètres locaux : 4](#__RefHeading___Toc405_1557683862)

[​ Paramètre de sortie : 4](#__RefHeading___Toc401_1557683862)

[​ Début 4](#__RefHeading___Toc403_1557683862)

[​ Fonction : SaveError 5](#__RefHeading___Toc705_1584694917)

[​ Description : 5](#__RefHeading___Toc395_155768386211)

[​ Paramètres d’entrée : 5](#__RefHeading___Toc397_155768386211)

[​ Paramètre de sortie : 5](#__RefHeading___Toc707_1584694917)

[​ Début 5](#__RefHeading___Toc403_155768386211)

[​ Fonction : Execute 6](#__RefHeading___Toc540_3319804551)

[​ Description : 6](#__RefHeading___Toc395_15576838621)

[​ Paramètres d’entrée : 6](#__RefHeading___Toc397_15576838621)

[​ Paramètres locaux : 6](#__RefHeading___Toc405_15576838621)

[​ Début 6](#__RefHeading___Toc403_15576838621)

# Fonction : read\_barcode

## Description :

Cette fonction recherche et lit le(s) code à barre d’une image. Une erreur est générée s’il y a plus de code à barre que prévu.

## Paramètres d’entrée :

* path\_image : str : chemin de l’image
* nb\_bc : int : Le nombre maximum de codes à barre dans l’images

## Paramètres locaux :

* img : image
* clahe : image
* temp : image
* seuil : int
* data : tableau de str

## Paramètre de sortie :

* Listes des codes à barre

## Début

img = Lecture de l’image en niveau de gris

clahe = Application à img d’un CLAHE

seuil = 1

temp = clahe

Tant que seuil<255 et taille de data < nb\_bc Faire

résultats = recherche de de codes à barre dans temp

Pour résultat dans résultats Faire

Si!(résultat dans data) Alors

Ajouter à data résultat

Fin Si

Fin Pour

temp = clahe

Pour i variant de 1 à nombre de ligne de temp Faire

Pour j variant de 1 à nombre de colonne de temp Faire

Si temp[i,j] ≤ seuil Alors

temp[i,j] = 0

Sinon

temp[i,j] = 255

Fin Si

Fin Pour

Fin Pour

seuil = seuil + 5

Fin Tant que

Si taille de data > nb\_bc Alors

Générer une erreur

Fin Si

Retourne data

Fin

# Fonction : Valid\_**barcode**

## Description :

Vérifier si le(s) code(s) à barre sont dans le fichier de validation. Il remet en ordre les codes à barre

## Paramètres d’entrée :

* valid\_file : str : chemin du fichier de validation
* found\_cabs : tableau de str : les codes à barre trouvés

## Paramètres locaux :

* Index : tableau de int :
* valid\_tab : tableau de tableau de str

## Paramètre de sortie :

* Ordered\_cabs : tableau de str : les codes à barre ordonnés

## Début

Index = []

valid\_tabs = lire le fichier valid\_file

Pour valid\_tab dans valid\_tabs Faire

Pour found\_cab dans found\_cabs Faire

Si found\_cab dans valid\_tab Alors

ajouter à Index l’index de found\_cab dans valid\_tab

Sinon

Index = []

Sortir de la boucle for

Fin Si

Fin Pour

Si taille Index == taille found\_cabs Alors

Sortir de la boucle for

Fin Si

Fin Pour

Si taille de Index == 0 Alors

retourne Index

Fin Si

Ordered\_cabs = found\_cabs

Pour i variant de 1 à taille de found\_cabs Faire

Ordered\_cabs[Index[i]] = found\_cabs[i]

Fin Pour

retourne Ordered\_cabs

Fin

# Fonction : **SaveError**

## Description :

Sauvegarde les fichiers qui ont posé problème

## Paramètres d’entrée :

* Dir\_output : str : Dossier de sortie
* file : str : Nom du fichier rejet
* msg : str : message indiquant le rejet du fichier
* i\_log : str : Nom du fichier de log (optionnel)

## Paramètre de sortie :

* o\_log : str : Nom du fichier de log

## Début

Si dossier de sorti n’existe pas Alors

Création du dossier de sortie

Fin Si

Si dossier error n’existe pas Alors

Création du dossier error

Fin Si

o\_log = i\_log

Si i\_log vide ou i\_log n’existe pas Alors

o\_log = « AAAAMMJJHHMMSS\_log.txt »

Fin Si

Écrire dans o\_log → file : msg

Retourne o\_log

Fin

# Fonction : **Execute**

## Description :

Gère l’utilisation de cab

## Paramètres d’entrée :

* in\_args : namespace : valeur des arguments passés au programme

## Paramètres locaux :

* files : tableau de str : liste de fichiers
* file : str
* temp : tableau de str
* cabs : tableau de str
* file\_cabs : tableau de tableau de str
* file\_cab : tableau de str
* f\_csv : fichier
* log : str : Nom du fichier de log

## Début

Si dossier de sorti n’existe pas Alors

Création du dossier de sortie

Fin Si

files = Récupération du nom des fichiers contenu par le dossier d’entrée

temp = []

log = None

Pour file dans files Faire

Si file n’a pas de caractères spéciaux excepté ‘\_’ ni d’espace Alors

ajouter a temp file

Sinon

log = SaveError(Dossier de sortie, file, « Le nom du fichier contient un\des caractère(s) refusé(s) », log)

Fin Si

Fin Pour

Pour file in temp Faire

cabs = Essayer de trouver le(s) code(s) à barre de file

Si Erreur(plus de codes à barre que prévu) Alors

log = SaveError(Dossier de sortie, file, «Trop de code à barre », log)

Passer à un autre élément

Fin Si

Si cabs vide Alors

log = SaveError(Dossier de sortie, file, « pas de code à barre », log)

passer à un autre élément

Fin Si

Si fichier de validation Alors

cabs = Valid\_barcode (fichier de validation, cabs)

Si cabs vide Alors

log = SaveError(Dossier de sortie, file ,  « code à barre non présent

dans le fichier de validation », log)

Passer à un autre élément

Fin Si

Fin Si

Si cabs existe déjà Alors

log = SaveError(Dossier de sortie, file, « Code à barre  déjà utilisé » + cabs , log)

passer à un autre élément

Fin Si

Si cabs n’a pas de caractères spéciaux excepté ‘\_’ ni d’espace Alors

Ajouter à file\_cabs file et cab

Sinon

log = SaveError(Dossier de sortie, file, « Le code à barre contient un\des caractère(s) refusé(s) », log)

Fin Si

Fin Pour

files = liste de fichiers contenu dans le dossier de sortie

Si Export en csv Alors

Création du fichier csv f\_csv dans le dossier de sortie

Fin Si

Pour file\_cab dans file\_cabs

Si Export en csv Alors

Ajouter le ligne file\_cab dans f\_csv

Fin Si

Si code à barre seulement Alors

Si la jonction par « \_ » des codes à bar de file\_cab[2:] dans files Alors

log = SaveError(Dossier de sortie, file, « Le fichier existe déjà dans le dossier de sortie », log)

Sinon

Copier le fichier file\_cab[1] dans le dossier de sortie

Renommer le fichier file\_cab[1] par la jonction par « \_ » des codes à bar de file\_cab[2:]

Fin Si

Fin Si

Si code à barre en préfixe Alors

Si la jonction par « \_ » des codes à bar de file\_cab[2:] et du nom du fichier file\_cab[1]dans files Alors

log = SaveError(Dossier de sortie, file, « Le fichier existe déjà dans le dossier de sortie », log)

Sinon

Copier le fichier file\_cab[1] dans le dossier de sortie

Renommer le fichier file\_cab[1] par la jonction par « \_ » des codes à bar de file\_cab[2:] et du nom du fichier file\_cab[1]

Fin Si

Fin Si

Si code à barre en suffixe Alors

Si la jonction par « \_ » du nom du fichier file\_cab[1] et des codes à bar de file\_cab[2:] dans files Alors

log = SaveError(Dossier de sortie, file, « Le fichier existe déjà dans le dossier de sortie », log)

Sinon

Copier le fichier file\_cab[1] dans le dossier de sortie

Renommer le fichier file\_cab[1] par la jonction par « \_ » du nom du fichier file\_cab[1] et des codes à bar de file\_cab[2:]

Fin Si

Fin Si

Fin Pour

Fin