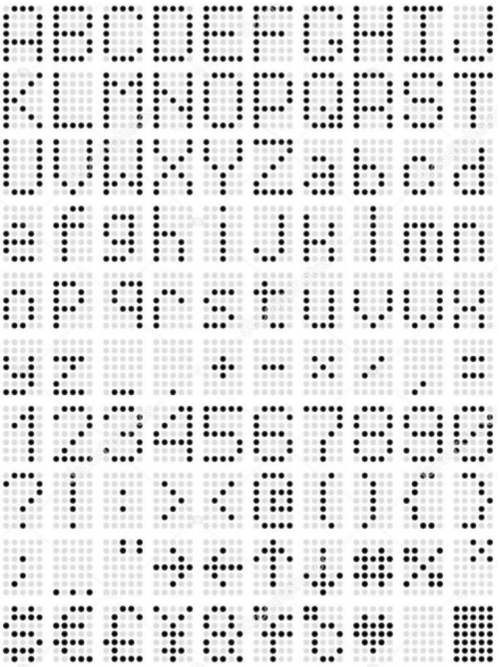
TD 7-8 Compression

Exercice 1: Codage

Choisir un de ces caractères et écrire sur un papier sa description afin qu'une autre personne puisse le refaire.



Afin de programmer un jeu pour le petit Cyril, proposer un encodage des données de ses cartes à partir de la capture d'écran ci-dessus:

Exercice 2: Entropie

1. Calculer les entropies des listes suivantes (à la main !)

Entropie de [1,2] :

Entropie de [1,1,2,2] :

Entropie de [1,2,1,2] :

Entropie de [1,2,3,4] :

Entropie de [1,2,3,4,5,6,7,8] :

Entropie de [0,1,2,2] :

Entropie de [0,1,2,2,3,3,3,3] :

Entropie de [0,1,2,2,3,3,3,3,4,4,4,4,4,4,4,4]

2. Classer ces types de fichiers suivants par entropie croissante : txt,odt,pdf,py,odp,zip,jpg,png,pcx,bmp
3. Programmer une fonction entropie pour un Binaire603 et vérifier vos réponses précédentes en TP.

Exercice 3: Compression par répétition

Créer une classe CompresseurSimpleParRepetition et de la tester sur un texte de votre choix.

On pensera notamment à comparer les entropies des textes compressés et non compressés.

Exercice 4: La classe Image603

Elle permet de travailler sur des images et de tester ainsi des méthodes de compression et de chiffrement.

Une Image603 contient une longueur lg, une hauteur ht et un tableau coul de lg×ht triplets d'octet représentant la couleur de chaque pixel

Ce tableau à deux dimensions est une liste et chaque couleur est un tuple de trois entiers de [0..255]

Les méthodes déjà écrites :

Image603(lg,ht) Le constructeur avec les dimensions en paramètres.

exImage603(num, lg, ht) Renvoie un Image603 de lg×ht pixels selon num (0: image de base, 1: image blanche, 2: image à lignes verticales et horizontales

iterXY(self) est un **itérateur** renvoyant les coordonnées (i,j) de chacun des pixels de l'image. Exemple :

```
for ix, iy in self.iterXY():
  print(f"Le pixel({ix},{iy} a pour couleur le triplet {self.coul[ix][iy]}")
```

A faire : toBinaire603(self) et fromBinaire603(monBin)

Proposer deux méthodes et en implanter au moins une.

Exercice 5: Compresseurs d'images

Programmer les compresseurs suivants et les appliquer à des images

- **CompresseurRLE**
- **CompresseurPCX**

Exercice 6: CompresseurHuffman.

On pourra éventuellement passer par les méthodes suivantes:

```
def dicoHuffmanDepuisArbre(arbre):
  """Renvoie les dictionnaires associant les étiquettes à leur codage d'Huffman"""

>>> a=CompresseurHuffman.arbreDepuisListePonderee([("B",0.3),("L",0.2),
("E",0.2),("I",0.1),("A",0.1),("T",0.1),("S",0.1),("N",0.1)])

>>> CompresseurHuffman.dicoHuffmanDepuisArbre(a)
  ({'000': 'E', '0010': 'S', '0011': 'N', '0100': 'A', '0101': 'T', '011': 'I', '10': 'B', '11': 'L'},
  {'E': '000', 'S': '0010', 'N': '0011', 'A': '0100', 'T': '0101', 'I': '011', 'B': '10', 'L': '11'})

def arbreDepuisListePonderee(lp):
  """Transforme la liste de couple (Etiquette,Entropie) en un tuple modélisant un arbre.
Un arbre pondéré peut-être un couple de la forme (Etiquette,pondération) ou
(Arbre,pondération)
>>>CompresseurHuffman.arbreDepuisListePonderee([("A",0.2),("B",0.3),"C",0.4)])
(((((('B', 0.3), ('A', 0.2)), 0.5), ('C', 0.4)), 0.9)

def codageHuffman(monBin,verbose=False):
  "Renvoie les dictionnaire associant les clés d'Huffman aux valeurs d'octets"
>>>CompresseurHuffman.codageHuffman(Binaire603([5,5,5,5,5,5,5,5,6,6,6,7,7,9]))
  ({'00': 6, '01': 5, '10': 7, '11': 9}, {6: '00', 5: '01', 7: '10', 9: '11'})

def binCode(self,monBin,verbose=False):
def binDecode(self,binC,verbose=False):

>>> monCodeur=CompresseurHuffman()
>>> monBin=Binaire603([6,6,6,6,6,5,5,5,5,6,6,6,7,8,9,8,8])
>>> monBinC=monCodeur.binCode(monBin)
>>> monBin==monCodeur.binDecode(monBinC)
True
```

Exercice 7: Travail personnel : Compression avec pertes)

Lire et tester les feuilles Jupyter et les programmes relatifs aux transformées de Fourrier et d'ondelettes (me les demander par mail.