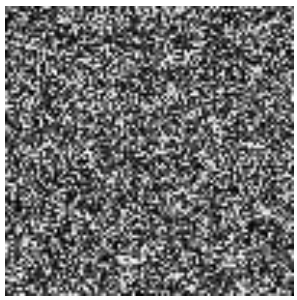
CNAM - DIU EIL - Bloc1 (USAL45)
Travaux pratiques : numériser, coder, chercher

P. Cubaud <cubaud @ cnam.fr>, juin 2019

Les exercices proposés ci-dessous ne suivent pas exactement l'ordre de l'exposé. Vous pouvez sauter ceux qui vous semblent inutiles au vu de vos compétences, mais il faudra me rendre deux exercices terminés avant le début du Bloc2. J'évalue très subjectivement la difficulté des exercices en deux niveaux : "ISN" (*) ou "hors ISN" (**) au vu des exercices proposés dans le manuel ISN Terminale S de G. Dowek et al. Les corrigés indicatifs des exercices seront diffusés au début du Bloc2.

1ère partie : formats de fichier d'image

EX1 - Produire une image calculée au format PGM (*)



Consulter Wikipedia sur les formats d'image PBM, PGM et PPM en mode texte. Ecrire ensuite un script qui produit sur la sortie standard un fichier PGM pour une image de dimension N x N pixels avec 256 niveaux de gris. Les valeurs des pixels sont générées aléatoirement entre 0 et 255. Avec la redirection Unix, le résultat du programme sera enregistré dans un fichier :

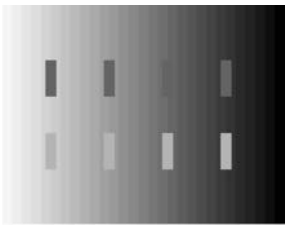
```
% python3 monprog.py > monfichier.pgm
```

Utiliser un visualiseur d'image (xv, par ex.) pour convertir ensuite l'image au format PNG. On peut le faire aussi avec la commande :

```
% ppmtopng  < monfichier.pgm >monfichierN.png
```

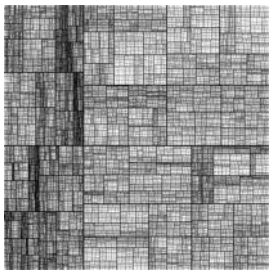
Comparer la taille des deux fichiers pour quelques valeurs de N, conclure sur la compression de ce type d'image.

EX2 - Dessin de base et export PGM (*)



Ecrire un script pour produire l'image de droite par export de PGM. On créera une structure de donnée qui stocke les valeurs de pixels avant l'export. On définira une fonction `rectangle()` pour remplir cette structure des diverses zones rectangulaires de l'image (dégradé de l'arrière plan et les 8 petits rectangles de l'avant plan).

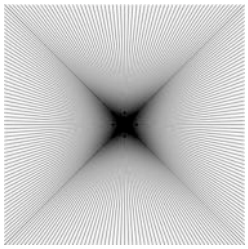
EX3 - Dessin récursif et export PBM (**)



Ecrire un script pour produire l'image de droite par export de PBM (le PGM est ici inutile). On créera une structure de donnée qui stocke les valeurs de pixels avant l'export. On définira des fonctions `vertical()` et `horizontal()` pour remplir cette structure avec des segments de droites verticales ou horizontal. L'algorithme de dessin est récursif avec un nombre de subdivisions N fixé à l'avance. Un point (x, y) est choisi au hasard (uniforme) dans le plan image, on trace les droites horizontales et verticales passant par ce point. Si le nombre de subdivisions n'est pas atteint, on relance le traitement sur les 4 portions de plan définies par le tracé précédent.

Bonus : faire un trait d'épaisseur décroissante selon N, et utiliser une loi de Gauss pour le placement du point de subdivision dans le plan.

EX4 - Image calculée au format SVG (*)
Consulter Wikipedia sur le format SVG, en particulier sur la balise à produire pour dessiner des droites. Ecrire via un éditeur de texte un petit fichier SVG avec quelques rectangles et droites. Le fichier résultant sera ensuite visualisé avec un navigateur Web.
Ecrire ensuite un script qui dessine par export de SVG la figure de droite. Etudier le phénomène de moiré selon l'espacement choisi pour les segments de droite.



2ème partie : quantification, codage

EX5 - Quantification du son (**)
Il existe en Python un module pour lire et écrire des fichiers sonores au format WAV. Lire sa documentation sur le site officiel python.org.
Ecrire ensuite un script qui charge un fichier de son mono WAV quantifié sur 8 bits (256 valeurs) et qui requantifie chaque échantillon sur 2 bits (4 valeurs). Le résultat est sauvegardé dans un autre fichier WAV et ensuite visualisé pour vérification avec Audacity.
Le fichier de départ est disponible sur l'espace moodle USAL45 et sur http://cedric.cnam.fr/~cubaud/TPDIU/quant8.wav

Bonus : Produire d'autres fichiers avec des quantifications différentes. Etudier l'opération inverse (conversion de 2 à 8 bits). Etudier aussi le ré-échantillonnage ("sous" ou "sur").

EX6 - Synthèse sonore (**)
Ecrire un script qui produit un fichier WAV monophonique de durée 10s pour un son de forme d'onde carrée de fréquence 400Hz. On fixera le framerate à 44100 Hz et la quantification sur 8 bits. Attention, ici, il faudra s'assurer que les échantillons produits sont bien codés sur 8 bits. Le résultat sera visualisé avec Audicity.
On utilisera ensuite la formule `freq = pow(2, 4.03 + K + n/12)` pour générer un son de la gamme tempérée, de note n = 0..11 et d'octave K = 3..6.

Bonus : Tirer au sort de notes et d'octaves pour faire de la composition automatique, produire une (ou plusieurs) forme(s) d'onde sinusoïdale, etc. On devra ensuite trouver une solution au phénomène de "claquement" qui se produit au passage d'une note à l'autre - voir l'illustration à droite. On peut par exemple moduler les échantillons de chaque note par une enveloppe en rampe décroissante ("fade-out").

EX7 - Itérations chaotiques de Colonna (*)
Ecrire un script qui effectue une itération de la suite logistique avec les mêmes conditions initiales et les variantes de calculs que le code de J.F. Colonna (cf. exposé)

```
dx1 = ((r+1)*dx1) - (r*(dx1*dx1))
dx2 = ((r+1)*dx2) - ((r*dx2)*dx2)
dx3 = (((r+1) - (r*dx3))*dx3)
dx4 = (r*dx4) + ((1 - (r*dx4))*dx4)
dx5 = dx5 + (r*((dx5) - (dx5*dx5)))
```

Ecrire une première version qui utilise des flottants. Ecrire une autre version avec le module décimal (lire avant sa documentation sur python.org). Faire varier la précision de

représentation jusqu'à obtenir un résultat identique sur les 6 variantes de calcul, pour diverses valeurs d'itérations (100, 500, 1000 par exemple).

Bonus : Trouver une méthode pour visualiser l'évolution de la suite dans les 6 variantes de calcul au lieu d'en afficher juste les valeurs.

EX8 - Codage et occurrences des caractères d'un texte (*)
Les textes classiques diffusés par la bibliothèque ABU (http://abu.cnam.fr) sont encore encodés en ISO-Latin1. Le vérifier en téléchargeant un texte de ce site puis en analysant le fichier avec la commande :

```
%od -ta letextechoisi.txt | more
```

Convertir le fichier en UTF-8 avec la commande Unix `iconv` (lire son manuel avec `man`)
Ecrire ensuite un script qui charge le fichier et compte les occurrences de chaque caractère. Le programme affichera ensuite ces mesures pour les caractères présents dans le texte.

Bonus : afficher une liste triée par occurrences décroissantes, compter les digrammes (séquences de deux caractères), comparer les statistiques de digrammes dans des textes de diverses langues, étudier la loi de Zipf (voir par ex. Wikipedia sur le sujet)

3ème partie : Données en tables

EX9 - Performance des tris (*)
Ecrire un script qui remplit une liste de N nombres entiers aléatoires. On veut ensuite trier cette structure en valeurs croissantes. On réalisera pour cela deux versions du script : une utilisant les fonctions Python de tri et une autre avec un tri par insertion codé directement. Mesurer les temps d'exécution des deux versions pour quelques valeurs de N. Vérifier si les mesures sont cohérentes avec la complexité en temps des algorithmes sous jacents.

EX10 - Cartographie (*)
On dispose d'un fichier CSV qui donne diverses informations concernant les 35000 communes de France continentale. Le fichier est disponible sur l'espace moodle de USAL45 et sur http://cedric.cnam.fr/~cubaud/TPDIU/communes.csv

Les colonnes contiennent les informations suivantes :
0 : numéro de commune
1 : département
2 : nom simplifié
3 : population
4 : superficie en km2
5 : longitude en degrés
6 : latitude en degrés
7 : altitude min
8 : altitude max

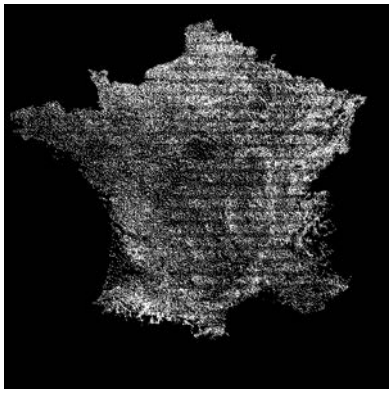
communes-simple-metro.csv

	A	B	C	D	E	F	G	H	I
1	1	1	ozan	618	6.6	4.91667	46.3833	170	205
2	2	1	cormoranche	1058	9.85	4.83333	46.2333	168	211
3	3	1	plagne	129	6.2	5.73333	46.1833	560	922
4	4	1	tossiat	1406	10.17	5.31667	46.1333	244	501
5	5	1	pouillat	88	6.23	5.43333	46.3333	333	770
6	6	1	torcieu	698	10.72	5.4	45.9167	257	782
7	7	1	replonges	3500	16.6	4.88333	46.3	169	207
8	8	1	corcelles	243	14.16	5.58333	46.0333	780	1081
9	9	1	peron	2143	26.01	5.93333	46.2	411	1501
10	10	1	relevant	466	12.38	4.95	46.0833	235	282
11	11	1	chaveyriat	927	16.87	5.06667	46.1833	188	260
12	12	1	vaux en bug	1169	8.22	5.35	45.9167	252	681
13	13	1	maillat	668	11.31	5.55	46.1333	497	825
14	14	1	faramans	681	11.22	5.11667	45.9	255	306
15	15	1	beon	373	10.3	5.75	45.8333	228	1412
16	16	1	saint bernard	1375	3.15	4.73723	45.945	167	198
17	17	1	rossillon	148	8.07	5.6	45.8333	324	1022
18	18	1	pont d ain	2627	11.22	5.33333	46.05	232	314
19	19	1	nantua	3713	12.79	5.61667	46.15	427	1125
20	20	1	chavannes su	680	16.55		5 46.4333	175	218
21	21	1	neuville les d	1504	26.59		5 46.1667	210	271
22	22	1	flaxieu	63	2.79	5.73333	45.8167	225	385
23	23	1	hotonnes	306	28.4	5.68333	46	636	1338
24	24	1	saint sorlin e	1061	9.07	5.36667	45.8833	196	700
25	25	1	songieu	130	20.58	5.7	45.9667	567	1275
26	26	1	virieu le petit	309	16.36	5.71667	45.9	419	1524
27	27	1	saint denis e	2178	2.61	5.33333	45.95	234	338

Ecrire un script qui charge le fichier dans une structure de donnée adaptée. Avec les fonctions Python de tri, ordonner cette structure selon les populations décroissantes des communes puis par nom. On affichera ensuite les 20 premiers résultats.

EX11 - Production de cartes en SVG (**)

On reprend l'EX10 avec cette fois la production d'un SVG qui produit une carte des communes. On utilisera les latitudes (axe y) et les longitudes (axe x). On ne fera pas de projection (Mercator, etc.). On dessine avec des rectangles de dimension 2x2.



Ensuite, on modifie le script pour afficher avec une couleur différente les communes dont les noms terminent par "-ac" et ceux terminent par "-y".

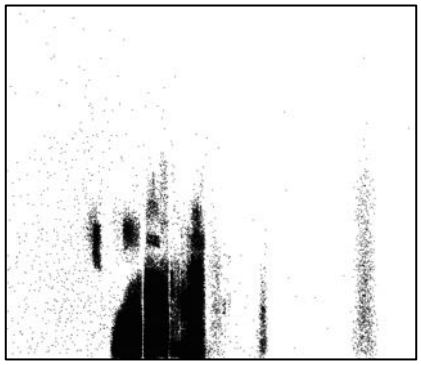
Pour les codes couleurs, on utilisera ceux préconisées par le site http://colorbrewer2.org

Bonus : effectuer d'autres traitements sur les données, reprendre le script avec production d'un PPM au lieu d'un SVG.

EX12 - Un peu d'astronomie et d'Unix (**)

Le Centre de données astronomique de Strasbourg diffuse sur le web de très nombreux catalogue de données astronomiques. Un de leurs catalogues recense les caractéristiques orbitales des astéroïdes du système solaire. Le fichier est disponible sur l'espace moodle de USAL45 et sur http://cedric.cnam.fr/~cubaud/TPDIU/asteroides.txt

- Mettre tout d'abord ce fichier au format CSV en retirant les en-têtes.
- Avec la commande `wc`, compter le nombre de ligne du fichier.
- Localiser ensuite les colonnes pour les éléments orbitaux suivants : demi grand-axe (noté a), excentricité (e) et inclinaison de l'orbite (i).
- Avec la commande `awk`, extraire du fichier les colonnes des données a et i et copier le résultat dans un nouveau fichier.
- Modifier le traitement pour ne retenir que les astéroïdes avec a<6
- Ecrire un script Python produisant un fichier SVG contenant pour chaque astéroïde un affichage de i=f(a), comme sur la figure de droite. On peut aussi étudier e=f(a).



Bonus : Produire un graphique comme celui qui illustre les "lacunes de Kirkwood" dans Wikipedia. Etudier `gnuplot` comme alternative pour le tracé de courbes sous Unix.