

# Programme principal

Il s'agit maintenant d'articuler toutes les «briques» de notre programme.

## Étapes-clés

1. Chaque groupe doit disposer:

- d'une image de base (disponibles sur Moodle);
- d'un enregistrement sonore (disponibles sur Moodle);
- d'une vidéo et de l'audio extrait de cette vidéo (disponibles sur [https://methotapes.com/BILANS\\_CLIMATIQUES/uploads/video/](https://methotapes.com/BILANS_CLIMATIQUES/uploads/video/));

2. On commencera par créer 4 variables contenant **les noms de ces fichiers**.

3. Le protocole de création du GIF comporte 25 itérations du même procédé:

- extraire une image de la vidéo, puis sa couleur dominante;
- extraire les données du son de l'enregistrement et du son de la vidéo (qui en fait se font au préalable);
- extraire la zone de l'image de base;
- lui appliquer le filtre
- enregistrer l'image

4. Une fois les 25 images créées, lancer la création du GIF:

- redimensionner les 25 images à la bonne taille 500x500;
- créer le GIF !

## Redimensionnement des images

Le logiciel (Open-source) ImageMagick permet de faire à peu près tout ce qu'on veut comme retouche d'images... en ligne de commande dans un terminal.

Par exemple, pour redimensionner une image `monimage.png` en créant une **nouvelle** image:

### Bash

```
convert -resize 500x500 monimage.png monimageredimensionnee.png
```

Ou bien en **écrasant** l'image:

### Bash

```
mogrify -resize 500x500 monimage.png
```

## 💡 Shell et Python

En Python, on peut exécuter une ligne de commande du shell depuis un programme. Pour cela il faut utiliser le module `os` et la fonction `system`:

### Script Python

```
1 import os  
2 os.system('la commande en chaîne de caractères')
```

On s'en servira pour:

- extraire l'audio de la vidéo avec le logiciel `ffmpeg`
- redimensionner avec `ImageMagick` les images générées en 500x500 pixels pour ensuite les ajouter au GIF.

## Module code élèves



Voici le fichier/module composé de vos fonctions qu'il faudra importer dans le programme principal.

moduleprojet.py

```

1 import scipy.io.wavfile as wave
2 import math
3 import numpy
4 import imageio
5
6 ### 1. Traitement du son #####
7
8 def spectre(data: list, rate: int, debut: float, duree: float) -> list:
9     """
10     Renvoie le spectre correspondant à un intervalle du signal.
11
12     data: le signal d'un canal
13     rate: la fréquence d'échantillonnage
14     debut: le début de l'intervalle à étudier (en secondes)
15     duree: la durée de l'intervalle à étudier (en secondes)
16     """
17     start = int(debut * rate)
18     stop = int((debut+duree) * rate)
19     s = numpy.absolute(numpy.fft.fft(data[start:stop]))
20     s = s / s.max()
21     return [math.log10(i) for i in s if i != 0]
22
23
24 def volumes_min(son)->list: #Mélodie
25     """
26     Prend en paramètre un son qui correspond
27     et renvoie une liste de 25 valeurs qui
28     correspondent aux volume minimaux de
29     chaque 1/25eme de la durée totale du son.
30     """
31     rate, echantillon = wave.read(son)
32     cd = [elt[1] for elt in echantillon]
33     duree = len(echantillon)/rate
34     duree_intervalle = duree//25
35     depart = 0.0
36     volumes_mini = []
37     for k in range (25):
38         s = spectre(cd, rate, depart, duree_intervalle)
39         volumes_mini.append(min(s))
40         depart += duree_intervalle
41     return volumes_mini
42
43 def pourcentage (lst:list)-> list: #Mélodie
44     """
45     Prends en paramètre la liste des volumes minimaux
46     et renvoie la liste contenant les pourcentages de
47     chaque point dans l'intervalle minimal-maximal de
48     la liste de volumes minimaux.
49     """
50     lst_pourcentage = []
51     maxi = max(lst)
52     mini = min(lst)
53     for elt in lst :
54         p = ((elt-mini)/(maxi-mini))*100
55         lst_pourcentage.append(int(p))
56     return lst_pourcentage

```

```

57
58
59     ### 2. Traitement de l'image #####
60
61     # Extraction de la zone
62
63     def extraction_zone(img: list, S: int, Vson: float, Vvideo: float) -> list: #Manon
64         """
65             Renvoie une image constituée des pixels de l'image img, de taille W avec le décalage x,
66             y
67             donnés par les paramètres S, Vson et Vvideo.
68         """
69         W = int(500 * S/100)
70         x = int((500-W) * Vson)
71         y = int((500-W) * Vvideo)
72         #On créait une image vide pour pouvoir y placer la zone extraite
73         zone = numpy.zeros((W, W, img.shape[2]), dtype=numpy.uint8)
74         for i in range(W):
75             for j in range(W):
76                 zone[i][j] = img[i+y][j+x]
77         return zone
78
79     def extraction(image_AP: list, S: int, Vson: float, Vvideo: float) -> list: #Jules
80
81         """
82             Créer une image vide, sélectionne une zone dans l'image
83             en paramètre et copie les pixels de la zone dans l'image vide.
84         """
85
86         W = int(500 * S/100)
87         x = int((500-W) * Vson)
88         y = int((500-W) * Vvideo)
89
90         zone = numpy.zeros([W,W,3], dtype=numpy.uint8)
91
92         for i in range(W):
93             for j in range(W):
94                 zone[i][j] = image_AP[y+i][x+j]
95
96         imageio.imwrite("monimage.jpg", zone)
97         return zone
98
99     # Recherche de la couleur dominante
100
101    def color_dom(image): #Yanis
102
103        # Cherche le pixel de l'image img qui revient le plus grand nombre de fois dans img.
104        hauteur = image.shape[0]
105        largeur = image.shape[1]
106        pixels = {}
107        for i in range(hauteur):
108            for j in range(largeur):
109                if tuple(image[i][j]) in pixels:
110                    pixels[tuple(image[i][j])] += 1
111                else:
112                    pixels[tuple(image[i][j])] = 1
113
114        # recherche du pixel ayant la plus grande occurrence
115        grand = ''
116        nmax = 0
117        for k, v in pixels.items():
118            if v > nmax:
119                grand = k
120                nmax = v

```

```

121     return grand
122
123 def couleur_dominante(image) -> tuple: # Manon
124     # On parcous l'image pour savoir combien de fois les couleurs apparaissent dans le
125     # tableau.
126     dico_RGB = {}
127     for i in range(img_test.shape[0]):
128         for j in range(img_test.shape[1]):
129             # On utilise un tuple pour qu'il soit possible de parcourir l'image.
130             t = tuple(img_test[i][j])
131             if t in dico_RGB:
132                 dico_RGB[t] += 1
133             else:
134                 dico_RGB[t] = 1
135
136     # On recherche la couleur qui revient le plus.
137     clr_max_apparition = 0
138     for clr, n in dico_RGB.items():
139         if n > clr_max_apparition:
140             clr_max_apparition = n
141             couleur = clr
142
143     return couleur
144
145 # Application du filtre
146
147 """
148 la fonction f permet de superposer une teinte sur une couche
149 """
150 def f(a,b): #Florette
151     a=a/255
152     b=b/255
153     if a<0.5 :
154         r= 2*a*b
155     else:
156         r= 1-2*(1-a)*(1-b)
157     return int(r*255)
158
159 """
160 Renvoie une nouvelle image, donnée par l'application d'une filtre coloré à l'image de
161 départ img.
162 """
163 def filtre(img : list, couleur: tuple): #Florette
164     img_filtree = numpy.zeros((img.shape[0],img.shape[1],3), dtype=numpy.uint8)
165     for i in range(img.shape[0]):
166         for j in range(img.shape[1]):
167             r=img[i][j][0]
168             g=img[i][j][1]
169             b=img[i][j][2]
170             img_filtree[i][j] = (f(r,couleur[0]), f(g,couleur[1]), f(b,couleur[2]))
171     imageio.imwrite("imageavecfiltre.jpg", img_filtree)
172     return img_filtree
173
174 def f(a, b): #Dawson
175     a = a / 255
176     b = b / 255
177     if a < 0.5:
178         r = 2 * a * b
179     else:
180         r = 1-2*(1-a)*(1-b)
181     return int(r*255)
182
183
184 def filtre(img: list, couleur: tuple): #Dawson

```

```

185     """
186     Renvoie une nouvelle image, donnée par l'application d'une filtre coloré à l'image de
187     départ img.
188     """
189     img_filtree = numpy.zeros((img.shape[0], img.shape[1],3), dtype=numpy.uint8)
190     for i in range(img.shape[0]):
191         for j in range(img.shape[1]):
192             r = img[i][j][0]
193             g = img[i][j][1]
194             b = img [i][j][2]
195             img_filtree[i][j]=[f(r,couleur[0]), f(g,couleur[1]), f(b,couleur[2])]
196     return img_filtree
197
198
199 # Conversion RGB -> HSL
200
201 def convertisseur (r,g,b :int)-> int: #Safia
202     """
203     Convertit la couleur RGB en HSL
204     """
205     r, g, b = r/255, g/255, b/255
206     Cmax=max(r,g,b)
207     Cmin=min(r,g,b)
208     Cdiff=Cmax-Cmin
209     if Cmax==0:
210         T=0
211     elif Cmax == r:
212         T=(((g-b)/Cdiff)) % 6
213     elif Cmax == g:
214         T = (((b-r)/Cdiff) + 2) % 6
215     elif Cmax == b:
216         T = (((r-g)/Cdiff) + 4) % 6
217     if Cmax == 0:
218         T = 0
219     t=60*T
220     L=1/2*(Cmax+Cmin)
221     if L==1:
222         S=0
223     else:
224         S=Cdiff/(1-abs(2*L-1))
225     return t,S*100,L*100

```

## Squelette du code

### Script Python

```
1 # Import des modules
2
3
4 # Chargement des fichiers de données
5
6 video = 'data/video_station1_groupe5.mp4'
7 son = 'son_station1_groupe5.wav'
8 image = imageio.imread('images/Gr5_Sp1_Florette2.jpg')
9 audio = 'audio.wav'
10
11 # Extraction du son de la video
12
13 os.system('ffmpeg -i ' + + ' ' + )
14
15 # Extraction des volumes du son et de l'audio
16
17 v_son =
18 v_video =
19
20 # Création du reader de la vidéo
21
22 reader = imageio.get_reader(video)
23
24 # Boucle principale
25
26 for k in range(25):
27     # extraction de l'image de la video
28
29     # extraction de la couleur dominante, convertie en hsl
30
31     # extraction de la zone
32
33     # application du filtre
34
35     # enregistrement de l'image
36
37
38 # Création du GIF
39 with imageio.get_writer('GIF_ultime.gif', mode='I') as writer:
40     for k in range(25):
41         #redimensionnement de l'image en ligne de commande
42
43         #lecture de la k-ième image du gif
44
45         # ajout de l'image au writer
```