Modularité

Lorsque l'on apprend à programmer, les codes sources sont relativement simples et on écrit ces derniers très souvent de manière linéaire. Si l'on veut réutiliser du code déjà existant, on utilise la méthode magique du «copier-coller». Sur des projets avec un nombre de lignes beaucoup plus conséquent, cela pose pas mal de problèmes, notamment la difficulté de maintenantce.
Le principe de *modularité* consiste à un découpage des différents aspects du programme et des différentes tâches qui doivent être accomplies de manière que chacune puisse être réalisée indépendamment des autres.

1 Utiliser une bibliothèque

Dans Python, une *bibliothèque* est un ensemble de *modules* permettant d'ajouter des possibilités étendues dans une thématique donnée. C'est une sorte de boîte à outils qui apporte ce qui n'existe pas de manière basique dans le langage.
Un module est un simple fichier Python qui contient des collections de fonctions et de variables globales et qui a une extension `.py`.
On peut citer quelques bibliothèques communes :

- **PIL** pour la manipulation et le traitement d'images.
- **Pygame** pour la création de jeu 2D.
- **Django** pour le développement WEB.
- **Tkinter** pour l'affichage graphique.

On aurait pu en citer de nombreuses autres comme, **Math**, **Random**, **Time**, ...
Nous allons prendre des exemples dans le domaine du traitement d'image et nous intéresser à la bibliothèque PIL. La première chose à faire est de se renseigner sur les posibilités proposées par cette bibliothèque et pour cela, il faut **lire la documentation** de cette dernière. Ce document est souvent en anglais et possède un nombre important de pages, cependant c'est la seule solution pour pouvoir utiliser correctement la bibliothèque.
Dans notre cas, la documentation se trouve sur le site internet suivant :

`https://pillow.readthedocs.io/_/downloads/en/stable/pdf/`

Nous allons illustrer quelques exemples de cette bibliothèque à l'aide de l'image `tiger.jpg` que l'on souhaite transformer en niveau de gris et enregistrer en format PNG :

Essayons de décompser nos contraintes pour chercher dans la documentation les informations souhaitées :

- Comment charger une image ?
- Existe-t-il une fonctionnalité pour convertir une image en nuance de gris ?

- Comment sauvegarder une image et choisir son format ?

Après lecture de la documentation, nous avons nos réponses :

- Au début du tutoriel, page 12, dans la rubrique «*Using the Image class*», on nous informe que la lecture d'une image se fait à l'aide de la fonction `open()` du module `Image`.

```
from PIL import Image
img_depart = Image.open('tiger.jpg')
```

- Dans le paragraphe suivant «*Reading and writing images*», on nous apprend que la classe `Image` a une méthode `save()`. De plus, lors de la sauvegarde des fichiers, le nom devient important. A moins que vous ne spécifiez le format, la bibliothèque utilise l'extension du nom de fichier pour découvrir quel format de stockage de fichier utiliser.

```
img_arrivee.save('tiger_ndg.png')
```

- Dans la section «*2.2.5 Color transforms*» la méthode `convert()` est présentée. Elle permet de convertir des images dans différentes représentations. Son utilisation est détaillée dans le chapitre «*3.1.3 The Image Class*» ; la nuance de gris est obtenue à l'aide du mode `L`.

```
img_arrivee = img_depart.convert('L')
```

```
from PIL import Image

img_depart = Image.open('tiger.jpg')

# on converti notre image de depart en niveau de gris
img_arrivee = img_depart.convert('L')

# on enregistre notre nouvelle image
img_arrivee.save('tiger_ndg.png')

# on libere la memoire
img_depart.close()
img_arrivee.close()
```

2 Utiliser une API

Une API, de l'anglais «*Application programming interface*», est une *interface de programmation d'application*, c'est-à-dire un moyen mis en place par une application pour que d'autres applications puissent interagir simplement avec. Nous utilisons alors les fonctions et les méthodes publiques.
Tout comme les bibliothèques, il existe de nombreuses API, parmi elles, les API Web sont de plus en plus nombreuses. Nous allons nous intéresser plus particulièrement à l'API *Openweathermap*, un service concernant le climat et la météo que l'on trouve à l'adresse suivante :

`https://openweathermap.org/api`

Pour l'utiliser, il faut d'abord obtenir une clef d'authentification sur la page ci-dessous :

`https://openweathermap.org/price`

La pluspart sont payantes, la gratuite sera largement suffisante pour nous, il faut cependant s'enregistrer pour l'obtenir.
Nous aimerions, à l'aide d'*Openweathermap*, donner la météo du jour de Strasbourg. Pour cela nous utiliserons «*current weather data*», donc la documentation se trouve ici :

`https://openweathermap.org/current`

Requête en Python

Pour effectuer des requêtes en Python, le plus simple est d'utiliser une bibliothèque comme `requests`.

```
import requests
```

La documentation précédente nous apprend qu'un appel à l'API se fait à l'aide de
`api.openweathermap.org/data/2.5/weather?q={city name},{state code}&{your api key}`
ce qui nous donne en Python

```
r = requests.get("https://api.openweathermap.org/data/2.5/weather?
                 q=Strasbourg,fr&appid=460d3781e4acdf238477fd60d376b9ea")
```

Les données envoyées sont au format JSON. Pour les afficher, la méthode `json()` produit une liste. Le détail du contenu de la liste peut se retrouver dans les exemples d'appel de l'API dans la documentation.

```
data = r.json()
>>> data
{'coord': {'lon': 7.74, 'lat': 48.58},
 'weather': [{'id': 800,
   'main': 'Clear',
   'description': 'clear sky',
   'icon': '01d'}],
 'base': 'stations',
 'main': {'temp': 306.44,
  'feels_like': 306.66,
  'temp_min': 306.15,
  'temp_max': 307.04,
  'pressure': 1009,
  'humidity': 38},
 'visibility': 10000,
 'wind': {'speed': 3.1, 'deg': 170},
 'clouds': {'all': 0},
 'dt': 1597932226,
 'sys': {'type': 1,
  'id': 6595,
  'country': 'FR',
  'sunrise': 1597897798,
  'sunset': 1597948505},
 'timezone': 7200,
 'id': 2973783,
 'name': 'Strasbourg',
 'cod': 200}
```

On peut par exemple obtenir la météo à Strasbourg de la manière suivante :

```
humid = data['main']['humidity']
temp = round(data['main']['temp']-273.5)
meteo = data['weather'][0]['description']
ville = data['name']

print(ville)
print('Temperature :',temp,'C')
print('Humidite :',humid,'%')
print('Meteo : '+meteo)
```

3 Créer un module

Lors de l'implémentation de structure particulières (pile, file, arbre, graphe, ...), nous allons implémenter nos propres modules. Il ne faudra surtout pas négliger la saisie de commentaire pour bien documenter son code.