CONCEPTION ET ECRITURE D'UN PROGRAMME

L'écriture d'un programme consiste à utiliser ou à créer du code pour répondre à un problème spécifique. En règle générale, on utilise du code déjà bien écrit, bien documenté et validé et on ajoute son propre code si besoin, en veillant à bien le documenter. **Lors de l'écriture d'un programme, il faut donc éviter de réécrire en moins bien ce qui existe déjà.**

MODULES

Un programme python commence souvent par des lignes d'importation contenant le mot clé **`import`**.

par exemple : **`from random import*`**.

Le mot clé **`import`** est aussi utilisé en **Java**. En **C++** on utilise le mot clé **`include`**.

En Python, un module peut être un simple fichier avec l'extension **`py`**. Ce fichier contient des définitions de fonctions, des constantes et des objets destinés à être utilisés.

Par exemple, on peut créer un fichier **`aires.py`** dans lequel on précise une valeur approchée du nombre π et quelques fonctions de calcul d'aire.

```
pi = 3.1416

def aire_disque(r):
    "renvoie l'aire d'un disque"
    return pi * r * r

def aire_rectangle(l,L):
    "renvoie l'aire d'un rectangle"
    return l*L

def aire_triangle(b,h):
    "renvoie l'aire d'un triangle"
    return b*h/2
```

Pour utiliser le module aires dans un autre programme, on importe le fichier. Le plus simple est de placer les deux fichiers dans le même dossier.

Exemple d'utilisation du module aires dans un autre programme programme :

```
import aires
print (aires.aire_disque(5))
```

Le point entre **`aires`** et **`aire_disque`** signifie que l'on se réfère à **`aire_disque`** qui est défini dans le module **`aires`**.

On peut aussi écrire :

```
from aires import *
print (aire_disque(5))
```

Avec cette méthode, le risque est d'écraser une autre fonction **`aire_disque`** écrite dans le programme principal. L'écriture **`aires.aire_disque`** montre clairement que **`aire_disque`** se trouve dans le module **`aires`**.

Une dernière possibilité est d'écrire :

```
from aires import aire_disque
print (aire_disque(5))
```

Avec cette méthode on n'importe que la fonction **aire_disque**.

On accède à la spécification d'une fonction avec la fonction **help :**

```
>>> help(aire_disque)
Help on function aire_disque in module aires:

aire_disque(r)
    renvoie l'aire d'un disque
```

EXEMPLES DE MODULES

Les modules Tkinter et Pygame apportent des outils pour la création d'interfaces graphiques.

BIBLIOTHEQUES

Une **biblithèque** (**library** en anglais) est en général une collection de modules. Il en existe de nombreuse en Python, en particulier dans le domaine scientifique. Les bibliothèques Python les plus connues sont :

Matplotlib pour les graphiques.

Numpy et **Scipy** pour le calcul scientifique.

PIL pour le traitement d'images.

Pygame pour la création de jeux en 2D.

Django pour le développement WEB.

API

Une **API** (**application programming interface** en anglais), est comme son nom l'indique une interface de programmation d'application. Une API est composée de fonctions, de classes, de constantes et sert de lien entre un programme et les programmes qui vont l'utiliser.

Il existe par exemple diverses API de géolocalisation qui peuvent être intégrées à des programmes.

API WEB

Les API Web sont de plus en plus nombreuses. Les réseaux sociaux par exemple, proposent des API permettant d'accéder à des données. Les API Web peuvent être gratuites ou payantes.

Exemples :

https://openweathermap.org/api propose diverses API sur le climat et la météo (gratuites ou payantes).

https://api.gouv.fr/rechercher-api propose des API disponibles et gratuites avec la documentation.