|  |  |
| --- | --- |
| Langages et Programmation | **Modularité : Création de modules** |

1. **Pourquoi créer ses propres modules ?**

On connait déjà quelques modules existants dans Python comme ***random***, ***math***, **tkinter** , **csv** ….

Nous avons vu que les fonctions sont utiles pour réutiliser une fraction de code plusieurs fois au sein d'un même programme sans avoir à dupliquer ce code. On peut imaginer qu'une fonction bien écrite pourrait être judicieusement réutilisée dans un autre programme Python. C'est justement l'intérêt de créer un module.

On y met **un ensemble de fonctions que l'on peut être amené à utiliser souvent.**

En général, les modules sont regroupés autour d'un thème précis.

1. **Création d'un module :**

|  |  |
| --- | --- |
| En Python, la création d'un module est très simple. Il suffit **d'écrire un ensemble de fonctions** (et/ou de constantes) dans un fichier, puis d'enregistrer ce dernier avec une extension en.py : momduodule.py  (comme n'importe quel script Python).  *Exemple* *: on créé un module simple que nous enregistrerons sous le nom****:  message.py****( voir ci-contre)*  *Les chaînes de caractères entre triple guillemets en tête du module et en tête de chaque fonction sont facultatives mais elles jouent un rôle essentiel dans la documentation du code.*  ***Remarque* :** Une constante est, par définition, une variable dont la valeur n'est pas modifiée. Par convention en Python, le nom des constantes est écrit en majuscules (comme DATE dans notre exemple). |  |

1. **Utilisation de son propre module**

|  |  |  |  |
| --- | --- | --- | --- |
| il faut que le module soit dans le **répertoire courant** (dans lequel on travaille).  Ensuite, il suffit d'importer le module avec ses fonctions (et constantes) .  Le chargement du module se fait avec la commande : **import message.**  Notez que le fichier est bien enregistré avec une extension .py et pourtant on ne la précise pas lorsqu'on importe le module.  Ensuite, on appelle une fonction par nomdumodule.nomdelafonction  Pour simplifier les appels on peut écrire **import message as m** (on renomme le nom du module par l’alias m) et lancer un appel par m.nomdelafonction  ( *exemple* m.ciao) | | **Ce qui donne dans la console :** | |
| 1. **Les docstrings** :   Lorsqu'on écrit un module, il est important de créer de la documentation pour expliquer ce que fait le module et comment utiliser chaque fonction. Les chaînes de caractères entre triple guillemets situées en début du module et de chaque fonction sont là pour cela, on les appelle *docstrings*  Ces *docstrings*  sont fournis avec la commande **help()** : voir ci-contre.  On peut appeler l’aide pour une seule fonction ( ex help(message.ciao))  Les *docstrings* sont destinés aux utilisateurs du module.  Leur but est différent des commentaires qui, eux, sont destinés à celui qui lit le code (pour en comprendre les subtilités). Une bonne *docstring* de fonction doit contenir tout ce dont un utilisateur a besoin pour utiliser cette fonction :  ce **que fait la fonction**, ce **qu'elle prend en argument,** ce **qu'elle renvoie**. |  | |

1. **Visibilité des fonctions dans un module** : Les fonctions dans un même module peuvent s'appeler les unes les autres.

Les fonctions dans un module peuvent appeler des fonctions situées dans un autre module s'il a été préalablement importé. Par exemple, si la commande import autremodule est utilisée dans un module, il est possible d'appeler une fonction avec autremodule.fonction().

TP :

Exercice 1 :

1. Créer un module aires ( fichier aires.py) avec les fonctions qui permettent de calculer l’aire d’un carré, rectangle, disque, triangle et la constante PI
2. Créer un autre fichier python ( à enregistrer dans le même dossier que aires.py) qui contiendra un script permettant de tester les fonctions du module aires

Exercice 2 :

1. Construire un module joueraucarte  qui contiendra :
2. les fonctions (dont on choisira le nom et on écrira une documentation) qui :

* crée un jeu de 32 cartes sous la forme d’un tableau de tuples (nom,valeur)
* mélange un jeu de cartes
* tire la 1ere carte d’un paquet et l’enlève du paquet et renvoie la valeur de la carte
* affiche le nom d’une carte sous la forme : roi de cœur , 7 de trèfle,…..
* compare 2 cartes en renvoyant la plus forte.

1. les constantes :

figure32 = ["7","8","9","10","valet","dame","roi","as"]

Couleur= ["coeur","carreau","pique","trefle"]

1. Dans un programme principal qui utilisera le module joueraucarte, simuler le jeu de batailles

(on affichera chaque coup joué et le score des 2 joueurs)