1 Créer son propre module en Python

Pour créer un module, il suffit de programmer les fonctions qui le constituent dans un fichier portant le nom du module, suivi du suffixe « `.py` ». Depuis un (autre) programme en Python, il suffit alors d'utiliser la primitive `import` pour pouvoir utiliser ces fonctions. On peut aussi définir des variables globales dans un module.

Par exemple, si l'on crée un fichier appelé `pile.py` et contenant :

```
contenu = {}

def push(e,p) :
  global contenu
  if p in contenu :
    contenu[p] = contenu[p] + [e]
  else :
    contenu[p] = [e]

def height(p) :
  global contenu
  if p in contenu :
    return(len(contenu[p]))
  else :
    return(0)

def pop(p) :
  global contenu
  if p in contenu and len(contenu[p]) > 0 :
    top = contenu[p][-1]
    contenu[p] = contenu[p][:-1]
    return(top)
  else :
    print ("pile.pop: ERREUR la pile %s est vide !" % p)

def show(p) :
  global contenu
  if p in contenu :
     for e in contenu[p] :
       print (e)
```

alors sous Python on peut écrire :

```
>>> import pile
>>> pile.contenu
{}
>>> pile.push(36,"test")
>>> pile.contenu
{'test': [36]}
>>> pile.push(2,"test")
>>> pile.contenu
{'test': [36, 2]}
>>> pile.push(2,"seconde")
>>> pile.contenu
{'test': [36, 2], 'seconde': [2]}
>>> pile.height("test")
2
>>> pile.height("seconde")
1
```

```
>>> pile.height("autre")
0
>>> pile.show("test")
36
2
>>> pile.pop("test")
2
>>> pile.show("test")
36
>>> pile.pop("test")
36
>>> pile.show("test")
>>> pile.height("test")
0
>>> pile.contenu
{'test': [], 'seconde': [2]}
```

On remarque que :

- `import` est une instruction de contrôle de Python très particulière puisqu'il n'est pas nécessaire d'écrire `pile` entre guillements pour en faire une chaîne de caractères, pourtant, `import` ne considère pas `pile` comme un nom de variable !
- On n'écrit pas le suffixe du fichier (`.py`) mais seulement le nom du module.
- Pour utiliser les fonctions et procédures du module, il faut les préfixerdu nom du module avec un point au milieu.

Python trouve ses modules dans le répertoire courant, comme on vient de le voir, mais aussi dans des répertoires prédéfinis où se trouvent des modules utiles écrits par d'autres programmeurs, comme ce fut le cas pour le module `os` dans la section précédente.

Un module peut faire appel à un autre module : il suffit qu'il contienne lui-même une instruction `import`.

Les modules dits « standard » sont documentés dans de nombreux livres sur Python et, avec un peu d'habitude, l'appel à des modules renforce considérablement la rapidité de programmation. Libre à vous d'explorer la liste presque sans fin des modules existants (et qui s'enrichit tous les jours) : vous avez maintenant toutes les bases de programmation nécessaires pour comprendre leur usage à partir de leur documentation.

Le TD n°10 sera consacré à des révisions. PRÉPAREZ VOS QUESTIONS À L'AVANCE !