La clause `import as`

Complément - niveau intermédiaire

Rappel

Jusqu'ici nous avons vu les formes d'importation suivantes (rappel de la 3ème semaine):

Importer tout un module sous son nom canonique

D'abord pour importer tout un module

```
import un_module
```

qui est pour simplifier équivalent à

```
un_module = __import__('un_module')
```

Importer un symbole dans un module, sous son nom canonique

Et dans la vidéo nous venons de voir qu'on peut aussi faire

```
from un_module import un_symbole
```

qui est grosso-modo équivalent à

```
temporaire = __import__('un_module')
```

un_symbole = temporaire.un_symbole del temporaire

Pour mémoire, le langage permet de faire aussi des `import *`, qui est d'un usage déconseillé en dehors de l'interpréteur interactif à cause du risque de collisions non controllées des espaces de nommage.

`import as`

Tout un module

Dans chacun de ces deux cas, on n'a pas le choix du nom de l'entité importée, et cela pose parfois problème.

Il peut arriver d'écrire un module sous un nom qui semble bien choisi, mais on se rend compte au bout d'un moment qu'il entre en conflit avec un autre symbole.

Par exemple, vous écrirez un module dans un fichier `globals.py` et vous l'importez dans votre code

```
import globals
```

Puis un moment après pour débugger vous voulez utiliser la fonction *builtin* `globals`. Sauf que, en vertu de la règle LEG, le symbole `globals` se trouve désigner votre module, et non la fonction.

À ce stade évidemment vous pouvez (devriez) renommer votre module, mais cela peut prendre du temps parce qu'il y a de nombreuses dépendances. En attendant vous pouvez tirer profit de la clause `import as` dont la forme générale est

```
import un_module as mon_nom_de_module
```

ce qui, toujours à la grosse louche, est équivalent à

```
mon_nom_de_module = __import__('un_module')
```

Un symbole dans un module

On peut de même importer un symbole spécifique d'un module, sous un autre nom que celui qu'il a dans le module. Ainsi

```
from un_module import un_symbole as mon_nom_de_symbole
```

qui fait quelque chose comme

```
temporaire = __import__('un_module')
```

mon_nom_de_symbole = temporaire.un_symbole del temporaire

Quelques exemples

Vous vous souvenez que nous avons dans notre environnement les modules

- `un_deux`
- `un_deux_trois`
- `un_deux_trois_quatre`

qui définissent les symboles `un`, `deux`, `trois` et `quatre`

```
import un_deux as autre_module
autre_module.un()


from un_deux_trois import un as autre_fonction
autre_fonction()
```

```
from un_deux_trois_quatre import un as u, deux, trois as t


u()
deux()
t()
```

Pour en savoir plus

Vous pouvez vous reporter à la section sur l'instruction `import` (https://docs.python.org/2.7/reference/simple_stmts.html#the-import- statement) dans la documentation python.