

Labo d'introduction à l'informatique

pour les mathématiques

Yann Thorimbert



**UNIVERSITÉ
DE GENÈVE**

CENTRE UNIVERSITAIRE
D'INFORMATIQUE

Semaine 11

Modules et packages
Déballages de séquences
Séquences en intention



Partie 1 - Modules et packages

Modules

- En python, tout script (fichier dans lequel vous écrivez du code) est automatiquement un "module" accessible pour d'autres fichiers.
- Exemple :
 - Je crée un fichier "monmodule.py" dans lequel je déclare une fonction `ma_fonction` et une variable `ma_variable`.
 - Dans un fichier voisin, je peux importer `monmodule` et invoquer `ma_fonction` ou `ma_variable` en précédant leur nom par le nom du module et un point (comme une méthode d'un objet).

```
ma_variable = 4

def ma_fonction(x):
    print(ma_variable * x)
```

Fichier monmodule.py

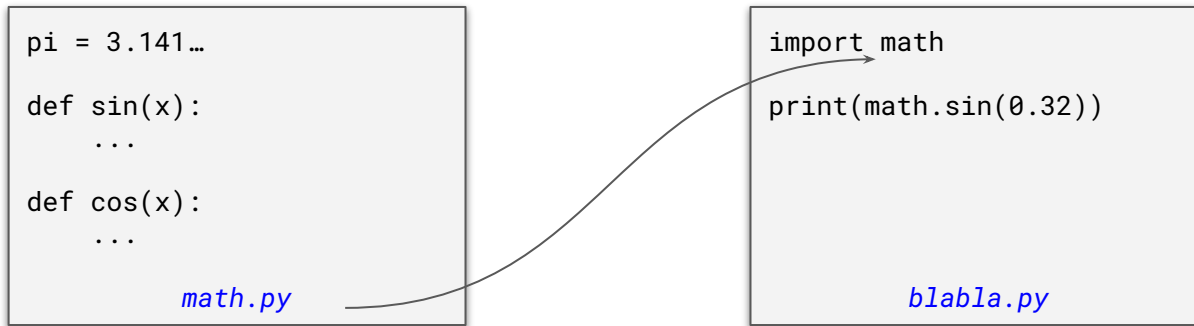
```
import monmodule

monmodule.ma_fonction(3) #affiche '12'
monmodule.ma_variable += 1
monmodule.ma_fonction(3) #affiche '13'
```

Fichier blabla.py

Modules et packages | L'exemple de 'math'

- Une collection de modules se nomme un package, ou une librairie.
- Vous avez déjà utilisé des packages/modules : matplotlib, math, ...
- Python vient avec des packages par défaut (*built-in*) comme math.
- D'autres packages sont à installer, comme matplotlib.



Différentes façons d'importer

- On peut importer tout un module et s'y référer :

```
import math  
print(math.sin(math.pi))
```
- On peut sélectionner des objets spécifiques à importer :

```
from math import sin, pi  
print(sin(pi))
```
- Ou encore tout importer dans le namespace courant (déconseillé) :

```
from math import *  
print(sin(pi))
```

Différentes façons d'importer

- Il est possible de définir des alias du module à l'importation :

```
import math as m  
print(m.sin(m.pi))
```
- Vous l'avez déjà fait par le passé :

```
import matplotlib.pyplot as plt
```
- Pour importer un package, il faut qu'il soit connu par l'interpréteur Python :
 - Soit parce qu'il fait partie des packages installés (voir "*Python path*") ;
 - Soit parce qu'on indique explicitement où se trouve le code correspondant (le cas de la création de package n'est pas traité dans ce cours) ;
 - Soit parce que le fichier correspondant est dans le dossier courant.

Installer de nouveaux packages

- Lorsqu'une librairie n'est pas built-in, il faut au préalable l'installer.
- Pour faciliter la gestion des packages, Python vient en général avec un outil nommé "pip".
- Assurez-vous que pip soit installé conjointement à Python, au moment d'installer ce dernier sur votre propre machine.
- Commande à lancer dans un terminal :
`pip install <le-nom-de-la-librairie>.`
- La façon d'ouvrir un terminal dépend beaucoup de votre environnement $\Rightarrow$ à vous de trouver l'information pertinente.



Partie 2 - Déballage de séquences



Accès aux éléments d'une séquence (méthode classique)

Méthode déjà connue :

```
texte = "Jean Dupont A8x4592"  
infos = texte.split()  
prenom = infos [0]  
nom = infos [1]  
identifiant = infos [2]
```



Accès aux éléments d'une séquence (déballage)

Raccourci syntaxique :

```
texte = "Jean Dupont A8x4592"  
infos = texte.split()  
prenom, nom, identifiant = infos
```

Fonctionne à condition que le nombre de variables corresponde !

Permet de simplifier en : `prenom, nom, identifiant = texte.split( )`



Déballage de tuples

Fonctionne sur les listes aussi bien que les tuples :

```
infos_geneve = ("Suisse", 530_000, 400, "français")  
pays, population, altitude, langue = infos_geneve
```



Déballage et boucles

Pratique très courante : déballer un tuple au sein d'une variable de boucle :

```
coords = [(2,-5), (-14, 8), (4,-3), (8,8)]
```

#chaque élément de notre liste est un tuple :

```
for x,y in coords:  
    distance_eucl_carre = x**2 + y**2  
    if distance_eucl_carre > 100 :  
        print(f"{{x,y}} est à une distance > 10 de l'origine")
```



Partie 3 - Séquences en intentions

Extension vs intension en mathématiques

Un ensemble se définit explicitement :

$$E = \{0, 1, 4, 9, 16, 25, 36, 49, 64\}$$

Ou en donnant la règle qui permet de l'obtenir :

$$E = \{x^2 \mid x < 9, x \in \mathbb{N}\}$$

Certains langages de programmation permettent de procéder de la même façon.

Extension vs intension en Python

Première méthode en Python :

```
E = [0, 1, 4, 9, 16, 25, 36, 49, 64]
```

Seconde méthode en Python :

```
E = []  
for i in range(9):  
    E.append(i**2)
```

Ou encore (nouveau pour nous) :

```
E = [i**2 for i in range(9)]
```

Listes en intention (*list comprehension*)

Certains types de traitement sont si courants qu'on leur a donné une syntaxe raccourcie :

```
noms = ["James", "Jennifer", "Michael", "Linda", "David", "Mary"]
for n in noms:
    if len(n) == 5:
        noms_de_5_lettres.append(n)
```

Peut se formuler en intention (*comprehension* en anglais) comme :

```
noms = ["James", "Jennifer", "Michael", "Linda", "David", "Mary"]
noms_de_5_lettres = [n for n in noms if len(n) == 5]
```



Listes en intention (*list comprehension*) avec déballage

```
coords = [(2, -5), (-14, 8), (4, -3), (8, 8)]
```

```
mes_coords_x = [x for x, y in coords if y < 0]
```

```
mes_coords_xyz = [(x, y, (x+y)/2) for x, y in coords]
```

À noter que la condition n'est pas obligatoire :

```
mes_coords_y = [y for x, y in coords]
```



Autres cas d'utilisation

```
mes_nombres = [2, 3, 4, 5, 7, 12, 34, 231, 234]
```

```
#l'utilisation conjointe avec des fonctions permet de simplifier:  
somme_pairs = sum([n for n in mes_nombres if n%2 == 0])
```

```
#si je définis une fonction mon_filtre qui retourne un booléen :  
mes_nombres_filtres = [n for n in mes_nombres if mon_filtre(n)]
```

```
#liste des nombres premiers plus petits que 100 :  
premiers_petits = [n for n in range(100) if est_premier(n)]
```



Autres séquences en intention

```
a = tuple(i for i in range(14, 50, 3))
```

```
noms = ["James", "Jennifer", "Michael", "Linda", "David", "Mary"]  
b = {nom : nom.count("e") for nom in noms}
```

```
c = [[k for k in range(i)] for i in range(3,6)]  
# c = [[0, 1, 2], [0, 1, 2, 3], [0, 1, 2, 3, 4]]
```