Fonctions

Nous avons déjà rencontré certaines fonctions déjà programmées dans PYTHON (`print()`, `input()`, `int()`). Nous avons également vu qu'il était possible de charger des modules, c'est-à-dire des bibliothèques qui contiennent d'autres fonctions préprogrammées (comme `sin()` dans le module `math`).

Mais il est également possible de programmer de nouvelles fonctions. Il y a plusieurs intérêts à cela :

- Pouvoir réutiliser plusieurs fois la même suite d'instructions sans avoir à la retaper
- Pouvoir adapter une même suite de fonctions à plusieurs situations en utilisant dans cette fonction des variables (les paramètres) que l'on pourra adapter à chaque cas particuliers rencontrés.
- Simplifier la création, la rédaction et la correction de programme en les décomposant en plusieurs sous-problèmes plus simples.

Dans la suite on distinguera une procédure, qui est une suite d'opérations regroupées sous un nom, et une fonction, qui est une procédure qui renvoie un objet informatique.

Syntaxe d'une procédure :

```
>>> def nom_de_la_procédure(liste de paramètres) :  
>>> ...  
>>> bloc d'instructions  
>>> ...
```

Rem : on peut choisir n'importe quel nom tant qu'on n'utilise pas de caractères spéciaux (sauf l'underscore) ni de mots-clés réservés.

Ne pas oublier : les deux points, l'indentation.

La liste de paramètres représentent les arguments de la procédure. Ils sont séparés par des virgules. Il peut n'y avoir aucun paramètre entre les parenthèses (mais il y a TOUJOURS des parenthèses).

Pour utiliser une procédure, il suffit de faire un appel de procédure en écrivant le nom de la procédure et en remplaçant les paramètres par les valeurs que l'on veut utiliser.

Ex :

```
>>> def somme(a,b) :
>>>     print(a+b)
```

Ex :

```
>>> def table7() :
>>>     n=1
>>>     while n<11 :
>>>         print(n*7, end=' ')
>>>         n=n+1
>>>
```

Ex :

Écrire une procédure table qui demande à l'utilisateur un entier naturel et qui donne la table de multiplication jusqu'à 10 pour ce nombre, appelé base de la table.

On peut également définir des paramètres par défaut

Ex :

```
>>> def table(base, max=10) :
>>>     n=1
>>>     while n<= max :
>>>         print(n*base, end=' ')
>>>         n=n+1
>>>
```

Si on appelle cette procédure en utilisant seulement un argument, cette valeur sera affectée à la variable base et la valeur max prendra par défaut la valeur 10. Si on utilise deux valeurs alors la première sera affecté à base et la seconde à max.

Il est possible d'utiliser une variable comme argument.

Ex : que fait le programme suivant ?

```
>>> a=1
>>> while a<11 :
>>>     table(a)
```

```
>>> a=a+1
>>>
```

Rem : on voit que le nom utiliser comme argument dans la définition de la procédure n'est pas toujours celui qu'on utilise quand on fait l'appel de la fonction. On dit que la variable utilisée dans la procédure est locale : elle n'a de sens que dans la définition, alors que la variable a dans l'exemple précédent est globale (elle est valable dans tout le programme).

Ex :

```
>>> def mask() :
>>>     p=20;q=3
>>>     print(p,q)
>>>
```

Qu'obtient-on quand on tape les instructions suivantes ?

```
>>> p=12;q=5
>>> mask()
>>> print(p,q)
```

Syntaxe pour une fonction :

```
>>> def nom_de_la_fonction(liste des arguments)
>>>     bloc d'instructions
>>>     return
```

On voit que pour PYTHON les deux objets sont identiques, seul le résultat varie : dans une fonction, l'algorithme retourne un résultat (qui pourra éventuellement être stocké dans une variable puis réutilisé).

Ex :

- écrire une fonction fact qui prend en argument un entier n et qui renvoie n !
- écrire une fonction coef_bin qui prend en argument deux entiers k et n et qui renvoie « k parmi n ».

Utilisation des fonctions dans un script

Pour utiliser les fonctions dans un script, il suffit dans un premier temps de définir toutes les fonctions dans une première partie du script, puis d'écrire le corps principal du programme.

Ainsi dans l'analyse d'un script, on commencera par le dernier bloc, et on étudiera les fonctions au fur et à mesure de leurs appels.

Exercices :

- 1) programmer la fonction f définie par $f(x)=1/(1+x^2)$ pour tout x réel.
- 2) Ecrire une fonction `max3` qui renvoie le maximum de trois nombres.
- 3) Programmer une fonction qui calcule le n -ième terme de la suite (u_n) définie par $u_0=1$ et $u_{n+1}=u_n+2/u_n$ pour tout n entier naturel.
- 4) Écrire une fonction dichotomie qui prend en argument une fonction continue f , trois nombres a , b et e et qui renvoie une approximation d'un zéro de f grâce à la méthode de dichotomie lorsque cette méthode aboutit.