

ALGORITHMIQUE


↔ Fonctions et procédures ↔

I. Fonctions prédéfinies

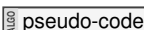
I.1. Introduction

Idée

Les **fonctions** sont un concept très important en programmation. Elles permettent notamment de *décomposer* un programme complexe en sous-programmes plus simples et d'améliorer ainsi la lisibilité d'un algorithme.

En , il existe un certain nombre de fonctions prédéfinies. Pour utiliser ces fonctions, on met le nom de la fonction, suivi des **paramètres** (ou **arguments**) de la fonction mis entre parenthèses. Les paramètres, si il y en a plusieurs, sont séparés par des virgules.




Algorithme

En , on peut utiliser des *fonctions* classiques, sans les redéfinir, mais uniquement en précisant brièvement leur rôle, comme par exemple :

```
Algorithme : ...
Variables : ...
Fonctions utilisées : reste(a,b) = reste de la division euclidienne de a par b
...
```

 Pseudo-Code

Remarque

Par exemple,  et  sont des fonctions implémentées dans .

De nombreuses fonctions ont déjà été utilisées lors des chapitres précédents, que ce soit pour les tableaux, ou les chaînes de caractères.

Python

Par exemple, on a :


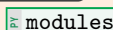


```
>>> min(3.1,5)
3.1
>>> type(42)
<class 'int'>
>>> T = sorted([1,5,2,8,7])
>>> print(T)
[1, 2, 5, 7, 8]
```

Début de la console 

Fin de la console 

I.2. Bibliothèques

Intro

Certaines fonctions de  ne sont pas directement chargées en mémoire. Elles sont regroupées dans des fichiers que l'on appelle  ou . Pour charger ces bibliothèques en mémoire, on rappelle qu'on utilise l'instruction .

Python

Pour utiliser la fonction `randint` de la bibliothèque `random`, on peut utiliser l'instruction `import` de plusieurs façons différentes :

- **Méthode 1** : en important la fonction voulue depuis (`from` en anglais) la bibliothèque;
- **Méthode 2** : en important toutes les fonctions de la bibliothèque; mais attention, si une fonction de l'espace de noms principal a le même nom que l'une des fonctions importées, cette première fonction sera écrasée par l'opération d'importation;
- **Méthode 3** : en important toutes les fonctions de la bibliothèque dans un `alias`. Pour utiliser les fonctions de la bibliothèque, il faut alors spécifier l'espace de noms dans lequel elles ont été chargées. Par défaut, ce nom est le nom de la bibliothèque. On peut aussi choisir un nom tout autre!

Il faut également noter que le fait de charger des modules complets (via `import *`) peut ralentir l'exécution.

Début de la console  python

```
>>> from random import randint      # on ne charge que la fonction randint
>>> randint(0,10)
9
>>> from random import *           # on charge tout le module random
>>> randint(0,10)
5
>>> import random                  # on importe random préfixé en random
>>> random.randint(0,10)
3
>>> import random as bob           # on importe random préfixé en bob
>>> bob.randint(0,10)
8
```

Fin de la console  python

II. Création de fonctions et procédures

II.1. Introduction

Intro

Définir une `fonction` est un peu comme écrire une recette de cuisine : on fait la liste des ingrédients (paramètres d'entrée), puis on note les instructions à effectuer avec ces ingrédients pour arriver à un résultat final. Mais une fois qu'on a fini d'écrire la recette de cuisine, on n'a toujours rien cuisiné : il faut que quelqu'un réunisse les ingrédients et effectue les instructions de la recette avec ceux-ci.

Dans notre contexte, après avoir défini une `fonction` (écriture de la recette) il faut écrire une instruction qui utilise la fonction (exécution de la recette). On parle d'`appel` à la fonction.

Attention : sans appel, aucune instruction n'est exécutée.

II.2. Premier exemple

Idée

Si on utilise régulièrement la même portion de code, on peut en faire une fonction que l'on pourra ensuite utiliser à volonté. De même, si un algorithme est très long, on peut faciliter sa lecture en le décomposant en sous-programmes simples, chacun de ces sous-programmes étant matérialisé par une fonction.

Exemple

Imaginons par exemple que l'on ait régulièrement besoin d'utiliser la fonction f définie par la formule mathématique $f(x) = x^2 + 5x + 3$. On peut alors créer une fonction `f`. Et une fois définie, on peut faire appel à la fonction `f`.

Algorithme

En `pseudo-code`, cela peut donner :

```
1 Fonction f(x) :
2 Variables : x
3 Début Fonction
4   res ← x**2 + 5*x + 3
5   Retourner : res
6 Fin Fonction
```

Pseudo-Code

Python

En `python`, cela peut donner :

```
1 def f(x):
2     """f(x) renvoie x**2 + 5*x + 3"""
3     res = x**2 + 5*x + 3
4     return res
```

Code Python

Début de la console `python`

```
>>> type(f)
<class 'function'>
>>> y = f(4)
>>> y
39
>>> type(y)
<class 'int'>
>>> print(f"f(15,1) vaut {f(15.1)}")
f(15,1) vaut 306.51
```

Fin de la console `python`

II.3. Cas général

Méthode

Pour définir une `fonction`, il faut respecter des règles précises :

- Les parenthèses après le nom de la fonction sont *obligatoires*. Si une fonction ne nécessite pas de paramètre, on ne met rien entre les parenthèses.
- En `python`, on commence en général la fonction par une ligne de documentation écrite entre triples guillemets, on parle de `docstrings`. Cette ligne n'est pas obligatoire, mais documenter ses fonctions est une bonne habitude à prendre.
- L'instruction `Retourner` ou `return` précède la liste des valeurs renvoyées par la fonction.

Attention, les instructions après le `Retourner` ou le `return` ne sont pas effectuées.

Algorithme

Voici à quoi ressemble une fonction définie en pseudo-code. Noter, de même qu'en `python`, les indentations nécessaires à la bonne lecture et à la bonne compréhension du bloc :


```
Fonction nom_fonction(parametre_1, parametre_2, ...) :
Variables : ...
Début Fonction
    instructions
    Retourner : valeur_1, valeur_2, ...
Fin Fonction
```

Pseudo-Code




Python

En  , cela peut donner :

```
1 def nom_fonction(parametre_1, parametre_2, ...) :
2     """Aide de la fonction"""
3     instructions
4     return valeur_1, valeur_2, ...
```

 Code Python


Remarque

Il est à noter qu'une  fonction est « typée », et que son type est définie par l'objet retourné !
Une  fonction pourra donc être (ré)utilisée et manipulée à l'aide des outils disponibles sur son  type.

III. Appel d'une fonction ou d'une procédure

III.1. Pour une fonction

Méthode

En général, on utilise une fonction en affectant la valeur renvoyée par celle-ci à une variable que l'on peut ensuite afficher ou réutiliser, grâce à  $a \leftarrow \text{fonction}(\text{parametres})$ ou  $a = \text{fonction}(\text{parametres})$.
Si on ne veut pas la réutiliser, on peut également directement afficher la valeur retournée par une fonction.

III.2. Pour une procédure

Intro

Une fonction peut ne rien retourner (en  , elle ne contient alors pas de  return, et pas de  Retourner en  pseudo-code). On parle alors de  procédure.

En  , si on ne met pas de  return, la fonction retournera en fait l'objet  None.

Puisqu'une procédure ne retourne aucun résultat, l'affectation de variable  $a = \text{procedure}(\text{parametres})$ n'a aucun sens :  a ne contiendra rien ( None en  .

Pour utiliser une procédure, on utilise la syntaxe « directe »  $\text{procedure}(\text{parametres})$!

Python

On peut illustrer, en   :

```
>>> a = print("1")      # stocke None dans la variable a
1
>>> print(a)
None
>>> print("1")         # utilisation correcte de print
1
```

Fin de la console  python

III.3. Exemples de fonctions et procédures


Exemple

On va définir :

- la fonction cube;
- une procédure affichage de la somme de deux nombres;
- une fonction (inutile) renvoyant 42;
- une fonction retournant plusieurs valeurs :

Python

En , cela peut donner :

 Code Python

```

1 def Cube(x) :
2     """Cube(x) renvoie le cube de x"""
3     res = x**3
4     return res
5
6 def aff_somme(x,y) :
7     """aff_somme(x,y) affiche la valeur de x+y"""
8     print(f"{x}+{y} vaut {x+y}")
9
10 def Reponse() :
11     """fonction sans argument renvoyant 42"""
12     return 42
13
14 def nb_et_carre(x) :
15     """fonction retournant le paramètre et son carré"""
16     return x, x**2
  
```

Début de la console 

```



>>> a = Cube(3)           # on appelle Cube et on stocke
>>> a
27
>>> Cube(4)
64
>>> Cube(5)
125
>>> aff_somme(3,2)
3+2 vaut 5
>>> Reponse()
42
>>> nb_et_carre(5)
(5, 25)
>>> a,b = nb_et_carre(12)
>>> a
12
>>> b
144
>>> nb_et_carre(12)[0]    # nb_et_carre(...) est une liste, ...[0] est le 1er élément !
12
>>> nb_et_carre(12)[1]    # nb_et_carre(...) est une liste, ...[1] est le 2nd élément !
144
  
```

Fin de la console 


IV. Exercices

Exercices

Exercice 1.



Écrire en  `pseudo-code` et en  `python` une fonction prenant en paramètres d'entrée deux nombres réels et retournant leur produit.

Exercice 2.




Écrire en  `python` une procédure prenant en paramètres d'entrée une phrase et l'affichant trois fois.

Exercice 3.

La température en degrés Fahrenheit est égale à 32 plus 1,8 fois la température en degrés Celsius.



- Écrire en  `pseudo-code` et en  `python` une fonction prenant en paramètre une température en degrés Celsius et renvoyant cette température en degrés Fahrenheit.
- Écrire une fonction faisant la conversion inverse.


Exercice 4.

Écrire en  `pseudo-code` et en  `python` une procédure  `conv_duree` prenant en paramètre une durée en secondes puis affichant cette même durée en heures, minutes et secondes.

Par exemple  `conv_duree(7422)` affiche "2 heures, 3 minutes et 42 secondes".

Exercice 5 (jeu de calcul mental).

Écrire en  `pseudo-code` et en  `python` une procédure prenant en paramètre un entier n et qui :

- tire au hasard deux entiers, notés x et y , entre 1 et n ;
- affiche ces deux entiers;
- demande de saisir le produit de ces deux entiers et stocke le résultat dans une variable  `reponse`,
- calcule puis affiche la variable erreur, valant la valeur absolue de $(\text{reponse} - x * y)$.