Travaux pratiques d'introduction 2

Les fonctions

Informatique tronc commun MPSI et PCSI

I Présentation

I.1 Les fonctions sont des objets comme les autres

Nous connaissons par exemple les objets de type *int*

```
a=3
type(a)
```

affiche

```
int
```

De même :

```
def ma_fonction(a,b):
    somme=a+b
    return somme

type(ma_fonction) #sans parenthèses
```

affiche

```
function
```

- Il s'agit de l'objet fonction qui est un outil comme peut l'être un mixeur, une machine à expresso ou une enceinte bluetooth ! A ce stade l'outil existe, il a un nom, mais il n'a pas encore servi.
- On notera au passage le décalage de 4 espaces dans le corps de la fonction ainsi que la présence du symbole `:`
- La fin de la fonction est marquée par le retour au même niveau que le mot clé **`def`** pour l'instruction qui suit la fonction. (ici `type(ma_fonction)`)
- Le nom de la fonction est suivi d'une parenthèse contenant les variables ou paramètres de la fonction. Ici il y a deux variables, a et b.

```
res=ma_fonction(2,4)
type(res)
```

affiche

```
int
```

Cette fois, l'outil a servi, il n'a pas produit un expresso mais un nombre entier. Il pourra servir à nouveau.
Le résultat renvoyé par l'appel de la fonction est ensuite associé à la variable **res** (on peut simplifier en disant qu'il est stocké dans la variable **res** comme l'expresso est stocké dans une tasse)

```
res=ma_fonction(2.5,5.6)
```

affiche

```
8.1
```

qui, cette fois, est du type **float**. (On notera la différence entre le . et la ,)

I.2 Certaines fonctions renvoient un objet, d'autres ne renvoient rien

La fonction précédente (**ma_fonction**) renvoie un objet. Le mot clé **return** précède l'objet à renvoyer.
De même, la machine à expresso renvoie un expresso lorsqu'on l'utilise.
L'enceinte bluetooth ne renvoie pas d'objet, mais ça ne signifie pas qu'elle ne fait rien, seulement qu'il n'y pas d'objet à attraper à la sortie !
Par exemple, la fonction :

```
def imprime(phrase):
    print(phrase, phrase)
```

peut être utilisée en tapant

```
imprime('oui')
```

elle affiche

```
oui oui
```

mais ne renvoie rien, ce qu'on peut observer en tapant

```
type(imprime('oui'))
```

qui renvoie

```
NoneType
```

qui est la manière pour Python d'appeler un non-objet, ou plus simplement le 'rien'.

I.3 Une seule sortie

Une fonction s'arrête lorsqu'elle rencontre son premier **return**, par exemple :

```
def multiplie(3,2):
    x=3*2
    return x
    x=3+2
    return x
```

retourne **help(6)** seulement.

I.4 Commentons nos fonctions

Il est très important de prendre l'habitude de commenter les programmes car dans un vrai projet informatique, il faut pouvoir se relire si on reprend le projet après un moment d'interruption ou si on est amené à le faire évoluer quelques mois ou années plus tard par exemple.
On peut commenter brièvement chaque ligne à l'aide d'un **#**, ce commentaire n'est alors pas lu par la machine et la seule façon de le voir est d'afficher le programme.
On peut aussi utiliser un docstring (documentation) en début de fonction, celui ci est affiché à l'aide de la commande **help** et s'écrit entre triples guillemets : *""" mon docstring """*,
par exemple :

```
def mon_exemple(a,b):
    """ a est un entier et
    b est un entier non nul"""
    divi=a//b
    return a-b*divi
```

```
help(mon_exemple)
```

affichera

```
Help on function mon_exemple in module __main__:

mon_exemple(a, b)
    a est un entier et
    b est un entier non nul
```

Exercice 1 - usage de l'aide
Lire l'aide de la fonction **abs** et celle de la fonction **print** .

enfin, pour y voir plus clair, on peut préciser le type des paramètres et celui du résultat.

```
def mon_exemple(a:int,b:int)->int:
    """ a est un entier et
    b est un entier non nul"""
    divi=a//b
    return a-b*divi
```

Attention, ce n'est qu'indicatif et Python n'en tient pas compte

```
mon_exemple(3,2.2)
```

affiche

```
0.7999999999999998
```

et

```
help(mon_exemple)
```

affichera

```
Help on function mon_exemple in module __main__:

mon_exemple(a:int, b:int) -> int
    a est un entier et
    b est un entier non nul
```

I.5 Paramètres optionnels

Dans la fonction suivante, les paramètres c et d sont définis implicitement, on peut les préciser ou pas lors de l'appel de la fonction.

```
def para_fac(a,b,c=3,d=6):
    total_a=c*a
    total_b=d*b
    return (total_a,total_b)
```

l'appel

```
para_fac(5)
```

donne

```
Traceback (most recent call last):

  File "<ipython-input-29-ea1f9d602597>", line 1, in <module>
    para_fac(5)

TypeError: para_fac() missing 1 required positional argument: 'b'
```

car on a oublié de définir `b` mais

```
para_fac(5,1)
```

donne

```
 (15, 6)
```

et

```
para_fac(5,1,5)
```

donne

```
 (25, 6)
```

On notera au passage que pour renvoyer plusieurs informations il suffit de les regrouper en un seul objet, ici le tuple `(total_a,total_b)`

I.6 Résumons

À retenir

1. Une fonction s'introduit par le mot clé **`def`** suivi du nom de la fonction, des variables entourées par un jeu de parenthèses et de `:` .

   Les instructions sont toutes décalées de 4 espaces, on dit que le bloc d'instructions est indenté.
2. Une fonction est un objet, le nom de l'objet s'écrit sans parenthèses.
3. On utilise la fonction en donnant des valeurs aux variables entre parenthèses. Certaines variables peuvent être omises si on leur a donné une valeur implicite en définissant la fonction. On les place en dernier.
4. Une fonction peut renvoyer un (et un seul) objet grâce au mot clé **`return`**, mais elle peut aussi ne rien renvoyer si on n'utilise pas **`return`**. Pour renvoyer plusieurs objets, on les regroupe par exemple dans un tuple.
5. Au premier **`return`** rencontré, on quitte la fonction sans lire les éventuelles instructions suivantes.
6. Enfin, il ne faut pas hésiter à commenter ses fonctions, à préciser éventuellement le type attendu des variables et à proposer un docstring (= documentation = aide pour l'utilisateur).
7. Il n'y a pas de différence de nature entre une fonction que vous créez et une fonction comme **`abs`** ou **`print`**.

II Exercices

Exercice 2 - Triangle rectangle
Écrire une fonction qui renvoie l'hypoténuse d'un triangle rectangle dont les 2 autres côtés sont donnés en paramètres. Essayer cette fonction pour un triangle de côtés 3 et 4 et stocker le résultat dans une variable.

Exercice 3 - Surf vol
Écrire une fonction qui prend en paramètre un nombre positif *rayon* et renvoie la surface du disque de rayon *rayon* et le volume de la sphère de rayon *rayon*. (On importera le module `math` afin de disposer de π)
Tester la fonction sur un exemple.

Exercice 4 - Logarithme
Le logarithme usuel en math est ln (noté `log` dans le module `math`), le logarithme népérien, de base `e`, on peut aussi définir log_a, le logarithme de base $a > 0$ par $log_a(x) = \dfrac{\ln(x)}{\ln(a)}$.
Ecrire une fonction qui calcule le logarithme de base a d'un nombre $x > 0$ et qui remplace implicitement a par e si l'utilisateur ne précise pas. Tester cette fonction.

Exercice 5 - Affiche
Écrire une fonction qui affiche `Hello world`.

Exercice 6 - Mon nom
Écrire une fonction `nom_prenom` qui prend en paramètre la variable `prenom` et qui renvoie l'initiale de `prenom` et qui affiche `Je m'appelle` suivi de la valeur de `prenom`.
Prédire ce qu'affichera `initiale=nom_prenom('Ursule')` ?

Exercice 7 - Un dé
Écrire une fonction qui simule le tirage d'un dé à 6 faces non truqué. On utilisera pour cela la fonction `randint` du module `random`.

Exercice 8 - Un dé à n faces
Écrire une fonction qui simule le tirage d'un dé à n faces non truqué, par défaut n sera égal à 6. Tester cette fonction plusieurs fois et pour différentes valeurs de n.

Exercice 9 - Polynôme
Écrire une fonction `poly` qui prend la variable a et renvoie la valeur du polynôme $X^3 + 5.1X^2 + 2$ en a.

Exercice 10 - Tirages successifs
Ecrire une fonction qui additionne les valeurs obtenues en lançant deux fois un dé non truqué à 6 faces. On utilisera la fonction `de6`.

Exercice 11 - 3 lancers
Ecrire une fonction qui simule 3 lancers successifs d'une pièce et qui affiche par exemple `PFF` si on a tiré *pile* puis $face$ puis $face$. (on utilisera un test `if...else`)
On rappelle que `'P'+'F'` vaut `'PF'`

Exercice 12 - Sinus
Qu'affiche l'instruction $print("\{\} \quad etait \quad \{\} \quad fois".format('il','une'))$?
Ecrire une fonction qui prend deux lettres a et b en entrée et qui affiche la formule $sin(a + b) = sin(a)cos(b) + sin(b)cos(a)$. L'utiliser avec d'autres lettres.

Exercice 13 - Somme
Ecrire une fonction `somme` qui prend deux entiers a et b en paramètres et affiche `a+b=s` où s est la somme de a et de b et qui retourne aussi s.
Tester `somme(2,3)+somme(3,2)`.