TP Info 2 | Fonctions

Ce deuxième TP a deux objectifs : consolider les acquis sur les tests et les boucles et manipuler les fonctions.

1. Ce qu'il faut savoir

Voici quelques rappels de syntaxe.

Définir une fonction sous Python

▷ De manière générale, la syntaxe pour définir une fonction est la suivante :

```
def nom(arg1,...,argN):
    '''Notice de la fonction'''
    Instructions
    return resultat (optionnel)
    Suite des instructions
```

On n'oubliera pas les `:` et l'*indentation* qui sont obligatoires en Python.

▷ Dès que l'instruction `return resultat` est exécutée (si elle est présente), l'exécution de la fonction se termine et la fonction renvoie l'expression `resultat` ; la partie du code écrite après l'instruction `return resultat` n'est pas exécutée.

▷ L'exemple de la partie entière :

```
def pe(x):
    n=0
    if x>0:
        while n<=x:
            n=n+1
        return n-1
    else:
        while x<n:
            n=n-1
        return n
```

Les lambda-fonctions

▷ Python permet une syntaxe intéressante qui permet de définir des mini-fonctions d'une ligne à la volée. Empruntées au langage Lisp, ces fonctions dites lambda peuvent être employées partout où une fonction est nécéssaire. La syntaxe est la suivante :

`nomfonction=lambda nomvariable: expression`

▷ Exemple :

```
>> f=lambda x: x**2+1
>> f(3)
10
>> f(10)
101
```

Appel d'une fonction sous Python

Si `f` est le nom de la fonction, on obtient le résultat de `f` pour un jeu de d'arguments `arg1`, ..., `argN` par l'appel suivant, conforme à l'usage mathématique :

`f(arg1,...,argN)`

Type d'appel sous Python

▷ Sous Python, le passage des arguments d'une fonction se fait par copie sauf pour les listes. Les expressions passées en paramètres sont évaluées avant d'être passées à la fonction.

▷ Exemples :

```
>> def f(x):
     x=0
>> a=1
>> f(a),a
1
```

```
>> def g(t):
     t[0]=4
>> a=[1,1]
>> g(a),a
[4,1]
```

Un argument peut être une fonction

```
>> def somme(f,n):
   somme=0
   for k in range(n+1):
      somme=somme+f(k)
   return somme
```

```
>> f=lambda x:x
>> somme(f,100)
>> 5050
```

2. Exercices

Les difficultés sont échelonnées de la manière suivante : aucune, ♪, ♪♪, ♪♪♪ et ♪♪♪♪. Certains énoncés sont tirés des annales des concours (oral et écrit) ; leur provenance est le plus souvent précisée. Les exercices notés ♪♪♪ et ♪♪♪♪ sont particulièrement délicats.

2.1. Fonctions

1 . [***B.A.ba sur les fonctions***]
Pour chacune des fonctions `f` suivantes, déterminer la valeur de `f(4)` :

```
def f(x):
  if x%2==0:
    a=0
  elif x%4==0:
    a=1
  else:
    a=3
  return a
```

```
def f(x):
  if x%2==0:
    a=0
  if x%4==0:
    a=1
  return a
```

```
def f(x):
  if x%4==0:
    a=0
  elif x%2==0:
    a=1
  return a
```

```
def f(x):
    if x%2==0:
        return 0
    if x%4==0:
        return 1
```

2 . [***Fonctions mystères***]
Que calcule chacune des fonctions suivantes ?

a)
```
def mystere1(x):
    y=x
    z=y*x
    y=x+y+z
    z=z-y
    return y-z
```

b)
```
def mystere2(x,y):
    z=x+y
    if z+y>2*x:
        return 1
    return -1
```

c)
```
def mystere3(x,y):
    z=x+y
    if z>y-x:
        w=10
    else:
        w=y-x
    t=w-z
    if t<=841:
        return y-z
```

3 . [***Passage par valeur dans les fonctions***]
On considère les codes suivants :

```
>>> x,y=1,2
>>> tmp=x
>>> x=y
>>> y=tmp
>>> x,y
(2,1)
```

```
def swap(x,y):
    tmp=x
    x=y
    y=tmp
```

```
>>> x,y=1,2
>>> swap(x,y)
(1,2)
>>> x,y
(1,2)
```

Expliquer la différence entre l'exécution de gauche et l'exécution de droite.

4 . [*Maximum*]

a) Écrire une fonction `max2(x,y)` qui renvoie le plus grand des deux nombres x et y.

b) En utilisant la fonction `max2`, écrire une fonction `max3` qui calcule le maximum de 3 nombres.

On dispose en Python d'une fonction prédéfinie `max` qui calcule le maximum d'un nombre quelconque d'arguments. Il est bien sûr défendu de l'utiliser pour cet exercice !

5 . [*Photocopies*]
Écrire une fonction `prixPhotocopies(n)` qui affiche le prix de n photocopies sachant que le reprographe facture $0,10$ euro les dix premières photocopies, $0,09$ euro les vingt suivantes et $0,08$ euro au-delà.

6 . [*La factorielle*]
Écrire une fonction `factorielle(n)` renvoyant $n!$.

7 . [*Fonction argument d'une fonction*]
Sous Python, une fonction est un objet comme un autre : il peut être donné en paramètre à une autre fonction.

a) Écrire une fonction `somme(f,a,b)` prenant en entrée une fonction f, deux entiers a et b avec $a \leqslant b$, et renvoyant

$$\sum_{k=a}^{b} f(k)$$

b) Calculer $\displaystyle\sum_{k=831}^{944} k^{10}$.

8 . [*Années bissextiles*]

a) Écrire une fonction booléenne `bissextile(annee)` qui permet de tester si une année est bissextile. On rappelle que les années bissextiles reviennent tous les 4 ans, sauf les années séculaires, si celles-ci ne sont pas multiples de 400. Ainsi, 1900 n'était pas une année bissextile, alors que 2000 l'était.

b) Écrire la même fonction en utilisant uniquement (si ce n'est déjà fait) des opérateurs logiques (sans branchement conditionnel).

9 . [*La suite de Fibonacci*]
On considère la suite $(\phi_n)_{n\geqslant 0}$ définie par :

$$\phi_0 = 0,\ \ \phi_1 = 1,\quad \forall n \in \mathbb{N},\ \ \phi_{n+2} = \phi_{n+1} + \phi_n$$

a) Écrire une fonction `fibonacci(n)` renvoyant ϕ_n.

b) Que vaut ϕ_{10} ? Idem avec ϕ_{100}.

c) Vérifier que ϕ_{10^7} est divisible par 1515.

On aura peut-être été amené à réécrire sa fonction en cours d'exercice. . .

10 . [***Suite et conjecture de Syracuse***]
La suite de Syracuse est définie par :

$$\begin{cases} u_0 & \in \mathbb{N}^* \\ u_{n+1} & = 1 \text{ si } u_n = 1 \\ & = \dfrac{u_n}{2} \text{ si } u_n \text{ est pair} \\ & = 3u_n + 1 \text{ sinon} \end{cases}$$

a) Écrire une fonction prenant en argument u_0 et n et calculant u_n.

b) On conjecture que, pour tout $u_0 \in \mathbb{N}^*$, il existe n tel que $u_n = 1$. Écrire une procédure prenant en argument u_0 et calculant le nombre minimum d'itérations nécessaires pour aboutir à 1.

11 . [***L'algorithme-mystère***]
On note a mod 10 et n/10 le quotient et le reste dans la division euclidienne d'un entier naturel a par 10. On considère l'algorithme suivant :

Algorithme 1 : Mystère

Données : Un entier naturel n
Initialisation : r ← 0, N ← n;
tant que $N > 0$ **faire**
  r ← 10r+(N mod 10);
  N ← N/10
Renvoyer r

a) Écrire une fonction `mystere(n)` correspondant à cet algorithme et la tester. Que conjecturer ?

b) Prouver votre conjecture.

2.2. Pour aller plus loin

12 . [***Jour de l'an***]
Sachant que le premier janvier 2013 est tombé un mardi, écrire une fonction `jourdelan(n)` retournant le jour de la semaine où tombe le premier janvier de l'année `n`. On prendra comme convention : 1 pour lundi, 2 pour mardi, etc.

13 . [***Les jours de la semaine***]
Écrire une fonction `jourDate(j,m,a)` prenant en entrée une date (ie un triplet d'entiers `(j,m,a)` représentant jour, mois et année) et retournant le jour de la semaine correspondant à cette date. Vérifier avec les jours de la semaine en cours.

14 . [***Largest palindome product, adapted from Euler project n^o 4***]
A palindromic number reads the same both ways. For example, 343 is a palindrome but not 344. The largest palindrome made from the product of two 2-digit numbers is 9009 (product of 91 and 99). Find the largest palindrome made from the product of two 3-digit numbers.

15 . [*Somme des décimales*]
Écrire une fonction prenant en entrée un entier n et renvoyant la somme des décimales de n.

16 . [*Nombres parfaits*]
On dit qu'un nombre n est parfait si la somme de ses diviseurs moins n vaut n, i.e.

$$n = \sum_{\substack{d|n \\ d \neq n}} d$$

Ecrire une fonction `estParfait(n)` prenant en argument un entier `n` et renvoyant sous la forme d'un booléen s'il est parfait ou non.

17 . [*Codage et décodage en base 2*]
On rappelle que 13 s'écrit 1101 en base 2 car

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13$$

a) *Décodage : de la base 2 à la base 10*. Écrire une fonction `decodage(L)` qui renvoie l'expression sous forme décimale de l'entier naturel dont les chiffres en base 2 sont contenus dans la liste `L`.

b) *Codage : de la base 10 à la base 2*. Écrire une fonction `codage(n)` qui renvoie la liste des chiffres en base 2 de l'entier naturel `n` entré sous forme décimale.