1ère NSI — Exercices d'Entraînement

Algorithmique

Lycée Fustel de Coulanges, Massy

Marc Biver, février 2024, *v0.1*

Exercice 1: Docstring & Tests

Soit la fonction de calcul de puissance suivante (dont on a étudié l'algorithme en cours) :

```
1 def Puissance(x, n):
2     r = 1
3     for i in range(n):
4         r = r * x
5     return r
```

a. Rédiger la docstring d'une telle fonction ;

b. Déterminer un jeu de tests pour cette fonction – en d'autres termes, compléter le tableau suivant qui liste les couples (x, n) qu'il conviendra d'utiliser pour convenablement tester cette fonction.

x	n	Résultat attendu
2	2	4
.....	.....	.....

Exercice 2: Variant & invariant de boucle

Soit la fonction suivante :

```
1  def SommeEltLst(liste):
2      '''
3      Fonction qui prend en entrée une liste de nombres et qui renvoie la
4      somme des nombres qui la constituent.
5      '''
6      res = 0
7      for i in range(len(liste)):
8          res += liste[i]
9
10     return res
```

a. Donner un variant de boucle pour cette fonction et montrer qu'elle se termine systématiquement ;

> **Rappel:**
>
> On rappelle qu'un variant de boucle est une quantité entière positive qui décroît strictement à mesure qu'on passe dans la boucle, ce qui permet de justifier que la boucle, tôt ou tard, se terminera.

b. Donner un invariant de boucle pour cette fonction et montrer qu'elle est correcte.

> **Rappel:**
>
> On rappelle qu'un invariant de boucle est une quantité ou une propriété qui est vraie avant et après chaque itération de la boucle – et en particulier *avant* que l'on rentre dans la boucle, et *après* sa dernière itération. Il permet de justifier que le résultat voulu sera atteint. On procède pour cette technique en quatre étapes :
>
> (a) On choisit l'invariant :
>
> — Comprendre clairement le but de la boucle – qu'est-elle censée accomplir ? Quel est le résultat attendu ?
>
> — Partir "de la fin", c'est-à-dire du résultat attendu et identifier quelle quantité est "construite" au fur et à mesure des itérations de la boucle pour constituer ce résultat.
>
> — Ceci devrait vous mettre sur la voie de votre invariant – une propriété (somme d'éléments déjà traités, ordre d'éléments dans une liste...) qui ne change pas malgré les itérations de la boucle.
>
> (b) On montre que l'invariant est vérifié avant la boucle (initialisation) ;
>
> (c) On montre que si l'invariant est vérifié *avant* un passage dans la boucle, alors il est préservé *après* le passage dans la boucle ;
>
> (d) On peut conclure sur la valeur finale à la sortie de la boucle.

Exercice 3: Variant & invariant de boucle — suite...

Soit la fonction suivante :

```python
def RechercheDiviseur(nb):
    '''
    Fonction qui prend en entrée un entier strictement supérieur à 1 nb
    et renvoie son plus petit diviseur autre que 1.
    Si elle n'en trouve pas (que le nombre est donc premier), elle
    renvoie 0.
    '''
    div = 0
    i = 2
    while i < nb:
        # On vérifie si nb est un multiple de i
        if nb % i == 0:
            # Dès qu'on en a trouvé un on le renvoie
            div = i
            return div
        i += 1
    # On atteindra ce point si on n'a pas trouvé de diviseur et on renverra
    ↪  donc la valeur initiale de div, soit 0.
    return div
```

a. Donner un variant de boucle pour cette fonction et montrer qu'elle se termine systématiquement ;

b. Donner un invariant de boucle pour cette fonction et montrer qu'elle est correcte.

> **Indice:**
>
> On rappelle qu'un invariant de boucle n'est pas nécessairement une formule mathématique – il peut aussi être une propriété que l'on pourrait exprimer par une phrase du genre (où bien entendu il faut remplir les trous) : "Quel que soit l'entier k tel que $2 \leq k \leq$ ___, k n'est pas___".

c. (question bonus 1) Cette boucle "`while`" semble assez artificielle – on dirait presque qu'elle n'est là que pour vous forcer à travailler sur la notion de variant de boucle... Quelle est la boucle "`for`" équivalente qu'on utiliserait plus logiquement dans un tel cas ?

d. (question bonus 2) Cette fonction n'est pas franchement optimale... Comment remplacer la condition "`while i < nb:`" pour lui faire faire nettement moins de tests tout en gardant le bon résultat ?

Exercice 4: Variant & invariant de boucle — encore un !

Soit la fonction suivante :

```
1  def RechercheMax(Lst):
2      '''
3      Fonction qui prend en entrée une liste de nombres positifs
4      et renvoie la valeur du plus grand d'entre eux.
5      '''
6      res = 0
7      for i in range(len(lst)):
8          if lst[i] > res:
9              res = lst[i]
10
11     return res
```

Donner un invariant de boucle pour cette fonction et montrer qu'elle est correcte.