

Exercice 1. Fonctions avec boucles for

Question 1 – Il faut utiliser un `range(1, n+1)`.

Question 2 – Le nombre exact de tours de boucle est n . Le nombre de tours de boucle est fini et toutes les opérations sont élémentaires donc la fonction termine pour toute entrée $n \in \mathbb{N}$.

Question 3 – Notre but est de montrer que la fonction renvoie $\sum_{k=1}^n k$. Les tours de boucle sont numérotés de $i = 1$ à $i = n$. Pour $i \in \llbracket 1, n \rrbracket$, on note s_i la valeur de la variable `s` à la fin du tour de boucle et :

$$(\mathcal{P}_i) : s_i = \sum_{k=1}^i k.$$

À la fin du premier tour de boucle, on a $s_1 = 1$, $i = 1$ et donc $\sum_{k=1}^1 k = 1$. Ainsi, (\mathcal{P}_1) est vrai.

Soit $i \in \llbracket 2, n \rrbracket$. On suppose que (\mathcal{P}_{i-1}) on montre (\mathcal{P}_i) . D'après la ligne 7, on a :

$$s_i = s_{i-1} + i = i + \sum_{k=1}^{i-1} k = \sum_{k=1}^i k$$

Donc (\mathcal{P}_i) est vraie.

Par itération finie, à la fin du dernier tour de boucle, la variable `s` vaut :

$$s_n = \sum_{k=1}^n k$$

On en déduit la correction de la fonction :

- Si $n = 0$, alors la fonction renvoie $0 = \sum_{k=1}^0 k$ (somme vide).
- Si $n \geq 1$, alors à la fin du dernier tour de boucle on a $i = n - 1$, donc $s = \sum_{k=0}^{n-1} k$. Ainsi, la fonction renvoie bien la somme des entiers.

Question 4 –

- * `fibo2` : `range(n-1)`.
- * `est_1er` : `range(2,n)`.
- * `cpt_1er` : `range(2,n+1)` :.

Question 5 –

- * `fibo2` : il y a exactement $(n-1)$ tours de boucle (et 0 pour $n = 0$).
- * `somme_liste` : il y a exactement `len(L)` tours de boucle.
- * `x_in_L` : il y a au plus `len(L)` tours de boucle.
- * `est_1er` : au plus $(n-2)$ tours de boucle.
- * `cpt_1er` : il y a exactement $(n-1)$ tours de boucle (et 0 pour $n = 0$).

Question 6 – Tous les invariants sont valables à la fin du tour de boucle.

★ **fibonacci** : on note $(\mathcal{F}_n)_{n \in \mathbb{N}}$ la suite de Fibonacci :

$$\mathcal{F}_0 = 0 \qquad \mathcal{F}_1 = 1 \qquad \mathcal{F}_{n+2} = \mathcal{F}_{n+1} + \mathcal{F}_n$$

Les tours de boucles sont numérotés de $i = 0$ à $i = n - 2$. Pour chaque $i \in \llbracket 0; n - 2 \rrbracket$, on note L_i la valeur de la variable L à la fin du tour de boucle numéro i . L'invariant de boucle est :

“ L_i est de taille $i+3$ et pour tout $k \in \llbracket 0; i + 2 \rrbracket : L_i[k] = \mathcal{F}_k$ ”.

On en déduit la correction de la fonction :

- Si $n = 0$, la fonction renvoie $0 = \mathcal{F}_0$.
- Si $n = 1$, la fonction renvoie $1 = \mathcal{F}_1$.
- Si $n \geq 2$, alors à la fin du dernier tour de boucle, on a $i = n - 2$. Ainsi, L est de taille $n + 1$ et pour tout $k \in \llbracket 0; n \rrbracket : L_i[k] = \mathcal{F}_k$. Finalement, la fonction renvoie $L[n]$ qui existe et vaut bien \mathcal{F}_n .

★ **x_in_L** : on numérote les tours de boucle par $i = 0, 1, 2 \dots$

“Si le tour numéro i est exécuté sans retour anticipé, alors x n'appartient pas à $L[0:i+1]$ ”.

où $i \in \llbracket 0, \text{len}(L) - 1 \rrbracket$ est le numéro du tour de boucle. On en déduit la correction de la fonction :

- Si la fonction renvoie **True**, c'est que la ligne 5 a été exécutée. Par la ligne 4 : $x == L[i]$ et donc x appartient à L .
- Si la fonction renvoie **False**, alors :
 - Si $n = 0$, alors $L = []$ et la fonction renvoie **False** ce qui est correct (aucun x n'appartient à $[]$).
 - Si $n \geq 1$, alors à la fin du dernier tour de boucle : $i = n - 1$. Donc d'après l'invariant de boucle, x n'appartient pas à $L[0:n]$. On en déduit que x n'appartient pas à L .

Finalement, la fonction renvoie **True** si x appartient à L et **False** sinon.

★ **somme_liste** : on numérote les tours de boucle de $i = 0$ à $i = n - 1$ où $n = \text{len}(L)$. Pour chaque $i \in \llbracket 0; n - 1 \rrbracket$, on note s_i la valeur de la variable L à la fin du tour de boucle numéro i . L'invariant de boucle est :

$$s_i = \sum_{k=0}^i L[k].$$

Pour la correction, on utilise le même raisonnement que pour la fonction **somme**.

★ **est_1er**. Les tours de boucle sont numérotés par $d = 2, 3, 4 \dots$. L'invariant de boucle est :

“Si le tour numéro d est exécuté sans retour anticipé, alors aucun entier dans $\llbracket 2, d \rrbracket$ ne divise n ”.

où $s \in \llbracket 2, n - 1 \rrbracket$ est le numéro du tour de boucle.

Pour la correction, on utilise le même raisonnement que pour la fonction **x_in_L**.

★ **cpt_1er**. Les tours de boucle sont numérotés de $i = 2$ à $i = n$. Pour chaque $i \in \llbracket 2; n \rrbracket$, on note s_i la valeur de la variable s à la fin du tour de boucle numéro i . L'invariant de boucle est :

“ s_i est le nombre de nombres premiers dans l'intervalle $\llbracket 2, i \rrbracket$ ”.

Pour la correction, on utilise le même raisonnement que pour la fonction **somme**.

Exercice 2. Fonctions avec boucles while

Question 1 – La quantité $v = \text{len}(L) - k - 1$ est un variant de boucle, en effet $v \in \mathbb{N}$ au début de chaque boucle (par la condition de la boucle `while`) et v décroît strictement à chaque tour. Puisqu’il existe un variant, le nombre de tours de boucle est fini. Toutes les opérations sont élémentaires, donc la fonction `ind` termine.

Question 2 – Montrons que la fonction renvoie le premier indice k tel que $L[k] = x$ et `None` si x n’appartient pas à L . Les tours de boucle sont numérotés par $i = 0, 1, 2, \dots$. Notons que la variable k vaut i au début du tour de boucle numéro i et $i+1$ à la fin.

Pour chaque $i \in \llbracket 0; n - 1 \rrbracket$, on pose :

(\mathcal{P}_i) : si le tour de boucle numéro i est exécuté, alors x n’appartient pas à $L[0:i+1]$.

Au début du premier tour de boucle : $i = 0$. Donc $L[0:i+1]$ est la liste qui ne contient que $L[0]$. Or x n’est pas égal à $L[0]$ d’après la condition de la boucle `while`. Donc (\mathcal{P}_0) est vraie au premier tour de boucle.

On suppose (\mathcal{P}_{i-1}) et on montre (\mathcal{P}_i) . Si le tour de boucle numéro i est exécuté, alors le tour de boucle numéro $i - 1$ a déjà été exécuté. Ainsi, par (\mathcal{P}_{i-1}) , x n’appartient pas à $L[0:i]$. De plus, d’après la condition de la boucle `while`, $L[i] \neq x$. On en déduit (\mathcal{P}_i) .

Soit k_0 la valeur de la variable k lorsque la boucle `while` s’arrête. Au début du dernier tour de boucle, la valeur de la variable k était $k_0 - 1$. Ainsi, d’après (\mathcal{P}_{k_0-1}) , x n’appartient pas à $L[0 : k_0]$. Notez que si $k_0 = 0$ alors (\mathcal{P}_{k_0-1}) n’est pas définie, mais on a quand même la propriété “ x n’appartient pas à $L[0 : k_0] = []$ ”.

D’après la condition de la boucle `while`, on sort de la boucle `while` pour l’une des deux raisons suivantes :

- $k_0 = \text{len}(L)$. Dans ce cas, l’invariant de boucle indique que x n’appartient pas à $L[0 : \text{len}(L)] = L$. La fonction renvoie `None` qui est le résultat attendu.
- $L[k_0] = x$. D’après (\mathcal{P}_{k_0-1}) , k_0 est le premier indice tel que $L[k_0] = x$. La fonction renvoie k_0 qui est le résultat attendu.

En conclusion la fonction est correcte.

Question 3 – La ligne 5 est `while i < len(L)`.

Question 4 – La quantité $v = \text{len}(L) - i$ est un variant de boucle.

Question 5 – On numérote les tours de boucle de $j = 0$ à $j = \text{len}(L)-1$. Notons que la variable i vaut j au début du tour de boucle numéro j et $j+1$ à la fin.

Pour chaque $j \in \llbracket 0, \text{len}(L) - 1 \rrbracket$, on note s_j la valeur de la variable s à la fin du tour de boucle d’indice j . La proposition suivante est un invariant de boucle :

$$s_j = \sum_{k=0}^j L[k]$$

Exercice 3. Fonctions récursives

Question 1 –

```

1 | # On suppose que n >= 0
2 | def nb_chiffres(n):
3 |     """(n: int) -> int"""
4 |     if n == 0:
5 |         return 0
6 |     else:
7 |         return 1 + nb_chiffres(n//10)

```

Question 2 –

Terminaison. On le montre par récurrence sur $n \in \mathbb{N}$.

★ Si $n = 0$ alors l'appel à `nb_chiffres(n)` termine d'après les lignes 4 et 5.

★ Si $n > 0$, on suppose que l'appel à `nb_chiffres(m)` termine pour tout $m < n$. L'appel à `nb_chiffres(n)` fait un appel à `nb_chiffres(n//10)` qui termine par hypothèse de récurrence. Toutes les autres opérations sont élémentaires, donc l'appel termine. En conclusion, l'appel à `nb_chiffres(n)` termine pour tout $n \in \mathbb{N}$.

Correction. On le montre par récurrence sur $n \in \mathbb{N}$.

★ Si $n = 0$ alors l'appel à `nb_chiffres(n)` renvoie 0 d'après les lignes 4 et 5 ce qui est correct.

★ Si $n > 0$, on suppose que l'appel à `nb_chiffres(m)` est correct pour tout $m < n$. L'appel à `nb_chiffres(n)` renvoie le nombre de chiffres dans $n//10$ par hypothèse de récurrence. On note c ce nombre de chiffres, alors l'appel principal renvoie $c + 1$ qui est bien le nombre de chiffres dans n . En conclusion, l'appel à `nb_chiffres(n)` renvoie le nombre de chiffres dans n pour tout $n \in \mathbb{N}$.

Exercice 4. Miroir d'une liste

Question 1 – On fait des exemples pour trouver la formule générale :

- `len(L) = 2` \rightsquigarrow `range(0, 1)`
- `len(L) = 3` \rightsquigarrow `range(0, 1)`
- `len(L) = 4` \rightsquigarrow `range(0, 2)`
- `len(L) = 5` \rightsquigarrow `range(0, 2)`
- `len(L) = 6` \rightsquigarrow `range(0, 3)`

Formule générale : `range(0, len(L)//2)`.

Question 2 – Le nombre de tours de boucle est :

$$\left\lfloor \frac{\text{len}(L)}{2} \right\rfloor.$$

La fonction `swap` est composée d'opérations élémentaires donc elle termine sous l'hypothèse que i et j sont des indices valides pour L . Le nombre de tours de boucle dans la fonction `miroir` est fini, cette fonction fait appel à `swap` qui termine et les autres opérations sont des opérations élémentaires, donc la fonction `miroir` termine pour toute entrée L .

Question 3 – On remarque que la fonction `swap` inverse les éléments d'indices i et j dans L . Pour $i \in \{0, \dots, \text{len}(L)//2 - 1\}$, on note L_i la valeur de la variable L à la fin du tour de boucle. On note (\mathcal{P}_i) la proposition suivante :

(\mathcal{P}_i) : La liste L_i est égale à L_0 où pour tout $k \in \llbracket 0, i \rrbracket$ les éléments d'indices k et $(\text{len}(L) - k - 1)$ ont été échangés.

La proposition (\mathcal{P}_i) se montre par itération finie sur i . On en conclut que la fonction `miroir` est correcte.

Exercice 5. Tri par sélection

Question 1 – La commande `print(tri([5,6,7,0,2]))` affiche ce que renvoie la fonction, c'est à dire `None`. Ça ne donne aucune indication sur la façon dont a été modifiée la liste. Pour tester la fonction, on peut exécuter le programme suivant :

```
1 | print(L)
2 | tri(L)
3 | print(L)
```

Question 2 – La fonction `mini` renvoie l'indice d'un élément minimum de $L[i0:]$. La fonction marche lorsque $i0$ est un indice valide pour L .

Terminaison. Cette fonction termine car la quantité $v = \text{len}(L) - i$ est un variant de boucle.

Correction. Pour $i \in \{i_0 + 1, \dots, \text{len}(L) - 1\}$, on note m_i la valeur de la variable m au début du tour de boucle et (\mathcal{P}_i) la proposition : $L[m_i]$ est le minimum de $L[i_0:i]$. Montrons que (\mathcal{P}_i) est un invariant de boucle :

- Au premier tour de boucle : $i = i_0 + 1$ et $m_i = i_0$ donc $L[m_i]$ est bien le minimum de $L[i_0:i]$ dont le seul élément est $L[i_0]$.
- Supposons (\mathcal{P}_i) et montrons (\mathcal{P}_{i+1}) . D'après les lignes 6 et 7, on a deux cas :
 - Si $L[m_i] > L[i]$ alors $m_{i+1} = i$. Or d'après (\mathcal{P}_i) , $L[m_i]$ est le minimum de $L[i_0 : i]$. On en conclut que $L[m_{i+1}]$ est inférieur strictement à tous les éléments de $L[i_0 : i]$. Donc (\mathcal{P}_{i+1}) est vraie.
 - Sinon $L[m_i] \leq L[i]$ et $m_{i+1} = m_i$. Or d'après (\mathcal{P}_i) , $L[m_i]$ est le minimum de $L[i_0 : i]$. On en conclut que $L[m_{i+1}]$ est inférieur ou égal à tous les éléments de $L[i_0 : i + 1]$. Donc (\mathcal{P}_{i+1}) est vraie.

Question 3 –

Terminaison. On remarque que la boucle `for` effectue $\text{len}(L)$ tours et que les appels aux fonctions `mini` et `swap` terminent. Donc la fonction `tri` termine.

Correction. On remarque que la fonction `swap` inverse les éléments d'indices i et j dans L . Pour $i_0 \in \{0, \dots, \text{len}(L)-1\}$, on note L_{i_0} la valeur de la variable L au début du tour de boucle d'indice i_0 et (\mathcal{P}_{i_0}) la proposition suivante :

$$\begin{cases} L_{i_0} \text{ et } L_0 \text{ contiennent les mêmes éléments} \\ L_{i_0}[0 : i_0] \text{ contient les } i_0 \text{ plus petits éléments de } L_0 \text{ triés par ordre croissant} \end{cases}$$

Montrons que (\mathcal{P}_{i_0}) est un invariant de boucle :

- Montrons (\mathcal{P}_{i_0}) pour $i_0 = 0$. $L_{i_0}[0 : i_0]$ est la liste vide donc (\mathcal{P}_0) est vraie.
- On suppose (\mathcal{P}_{i_0}) et on montre (\mathcal{P}_{i_0+1}) . D'après les lignes 4 et 5 :

$$\begin{cases} L_{i_0+1} \text{ et } L_{i_0} \text{ contiennent les mêmes éléments} \\ L_{i_0+1}[i_0] \text{ est l'élément minimum de } L_{i_0}[i_0 :] \end{cases}$$

En combinant avec (\mathcal{P}_{i_0}) , on en déduit (\mathcal{P}_{i_0+1}) .

On conclut que (\mathcal{P}_{i_0}) est un invariant de boucle.

À la sortie de la boucle `for`, la proposition $\mathcal{P}_{\text{len}(L)}$ est vraie, c'est à dire que $L[0 : \text{len}(L)]$ contient les $\text{len}(L)$ plus petits éléments de L_0 triés par ordre croissant. Finalement, la fonction `tri` est correcte.