1 Compréhension du cours

1.1 Des `range` inutiles

```
def somme(liste:list[int]) -> int:
    res = 0
    for i in range(len(liste)):
        res = res + liste[i]
    return res
```

1. Récrire le code de la fonction précédente en itérant directement sur les éléments de `liste`. Quand on n'a pas besoin d'utiliser les indices de la séquence, ne pas le faire allège le code.

1.2 Des while pour coder des range

```
def un_sur_deux(chaine : str) -> str:
    res = ''
    for i in range(0, len(chaine), 2):
        res = res + chaine[i]
    return res
```

```
def inversion(liste : list[int]) -> list[int]:
    res = []
    for i in range(-1, -len(liste)-1, -1):
        res.append(liste[i])
return res
```

2. Récrire le code des fonctions précédentes en remplaçant l'itération sur les indices du paramètre par une itération sur les éléments d'une sous-séquence extraite du paramètre avec un pas (donc sans `range` mais avec `chaine[...:...:...]` et `liste[...:...:...]` respectivement).
3. Récrire le code des fonctions précédentes en remplaçant la boucle `for` par une boucle `while`.
4. Quelle version a votre préférence ?
5. Est-il toujours possible de remplacer une boucle `for` par une boucle `while` ? une boucle `while` par une boucle `for` ?

1.3 Lecture de code

La fonction suivante renvoie un couple d'entier (type `tuple[int,int]`) pour pouvoir renvoyer deux valeurs et non une, d'où le `return (nb_soustractions, reste)`. Cet type et cette syntaxe seront aussi utilisé dans la fonction `mon_zip`.

```
def foo(a : in, b : int) -> tuple[int, int]:
    """ à découvrir !
    precondition :  a >= 0 et b > 0
    """
    reste = a                                       #1
    nb_soustractions = 0                            #2
    while reste >= b:                               #3
        nb_soustractions = nb_soustractions + 1     #4
        reste = reste - b                           #5
    return (nb_soustractions, reste)                #6
```

6. Que vaut `foo(22, 5)` ?
7. Quelle relation lie les valeurs de `a`, `reste`, `nb_soustractions` et `b` ? Cette relation doit être vraie à l'initialisation, et après chaque itération.
8. À la sortie de la boucle `while`, comment exprimer en fonction de `a` et de `b` la valeur de `reste` et de `nb_soustractions` ? Compléter la docstring.
9. Que se passe-t-il si on supprime la ligne 5 et qu'on évalue `foo(22, 5)` ?

2 Exécution d'une boucle `while` et tableau de la mémoire

10. Dérouler l'état de la mémoire sur l'appel à `foo(22, 5)`, pour la fonction définie précédemment.

3 Écriture de code avec un range

11. Écrire une fonction `mon_zip` qui prend en paramètre 2 listes d'entiers et renvoie une liste contenant les couples formés par les éléments des listes partageant le même indice.

```
>>> mon_zip([1, 2, 3, 4], [4, 5, 6])
[(1, 4), (2, 5), (3, 6)]
```

4 Écriture de code avec un while

4.1 Tests

12. Quels tests prévoir quand on pense qu'on va écrire une boule `while` dans le code ?

4.2 Saisie d'un entier avec vérification de sa valeur

13. Écrire une fonction `saisie_entier_intervalle` qui prend en paramètre les bornes d'un intervalle fermé et demande à l'utilisateur la saisie d'un entier dans cet intervalle, tant que la valeur saisie est incorrecte.

On proposera 2 solutions :

- l'une qui base la condition de boucle sur un booléen (méthode du drapeau), booléen initialement faux;
- l'autre qui exprime la condition de boucle en fonction de la donnée saisie.

4.3 Codage (si si !)

La fonction `bin` de Python prend en paramètre un entier quelconque et renvoie une chaîne de caractère contenant la représentation binaire de cet entier:

```
>>> bin(15)       >>> bin(-4)
'0b1111'          '-0b100'
>>> bin(7)        >>> bin(0)
'0b111'           '0b0'
```

On souhaite utiliser l'algorithme des divisions successives vues en codage pages 33 et suivantes, avec la base 2 :

`https://www.fil.univ-lille.fr/~codagel1/portail/documents/cours2.pdf`

14. Quelle est la précondition de cet algorithme ?

15. Écrire une fonction `binaire` qui prend en paramètre un entier satisfaisant la précondition ci-dessus et renvoie une chaîne de caractères qui contient la représentation binaire de cet entier contenant uniquement les digits `0` et `1`. Bien réfléchir au(x) cas particulier(s) à tester… et traiter de manière particulière dans le code.

Par exemple : `binaire(4)` vaut `'100'`.

16. Que se passe-t-il si on déroule l'algorithme sur une donnée qui ne satisfait pas sa précondition ?

17. Écrire une fonction `mon_bin` qui imite la fonction `bin` de Python.

4.4 Racine carrée entière

La *racine carrée entière* d'un nombre entier naturel x est le plus grand entier vérifiant $n^2 \leq x$. Si on nomme r cette racine carrée entière, alors r vérifie $r^2 \leq x < (r+1)^2$.

Pour calculer cette racine carrée entière, on teste consécutivement les entiers n en commençant à 0, en s'arrêtant quand n^2 dépasse x.

18. Quelle est la racine carrée entière de `9` ? de `10` ? de `8` ?

19. Définissez une fonction `racine_carree_entiere()` qui prend en paramètre un entier positif `x` et renvoie la racine carrée entière de `x`.