

## Exercices Python Janv1

1. Quel est le gros problème avec la fonction `pyramide_inversee` ?

```
def pyramide_inversee(n):
    print(' '*(20-n) + '*'*(2*n+1))
    pyramide_inversee(n-1)
```

2. La fonction `saute` a pour but de faire sauter un mouton d'un troupeau (liste de moutons) à un autre.

Bizarrement, on retrouve des `None` inattendus dans le troupeau et on cherche à comprendre d'où vient l'erreur. Quelle ligne peut-on rajouter dans le code ci-dessous qui va interrompre l'exécution du programme si le mouton récupéré dans `troupeau1` est égal à `None` ?

```
def saute(troupeau1: list, troupeau2: list):
    mouton = troupeau1.pop()
    troupeau2.append(mouton)
```

3. Trouver le bug dans ce code :

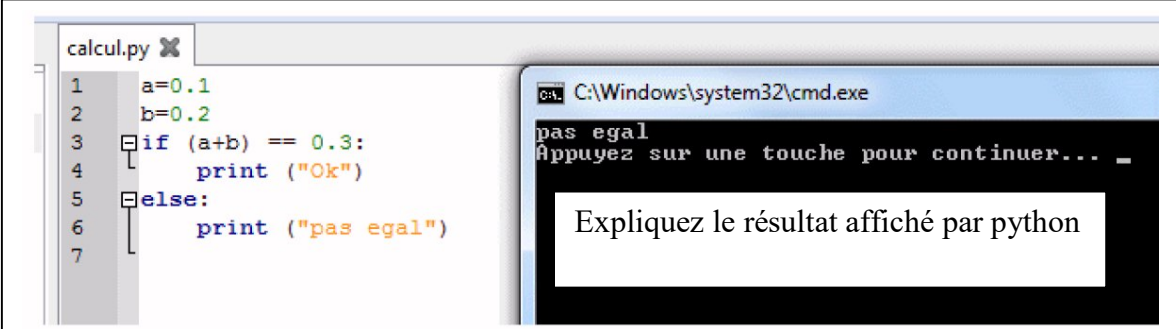
```
def element_central(lst):
    "précondition : lst est une liste non vide"
    n = len(lst)
    return lst[n/2]
```

4. Lors du parcours de deux listes Python avec le code ci-dessous s'affiche l'erreur "IndexError: list index out of range". Proposer une modification du code pour éviter cette erreur.

```
i = 0
while i < len(tab1) or i < len(tab2):
    tab1[i] = tab1[i] - tab2[i]
    i += 1
```

5. Corriger l'erreur dans le code suivant, la fonction permet d'indiquer la présence ou pas d'un élément dans une liste Python.

```
def est_present(lst, x):
    for e in lst:
        if e == x:
            return True
    return False
```

6. The screenshot shows a Python script named 'calcul.py' with the following code:  
1 a=0.1  
2 b=0.2  
3 if (a+b) == 0.3:  
4 print ("Ok")  
5 else:  
6 print ("pas egal")  
7  
The script is executed in a Windows command prompt, showing the output: 'pas egal' followed by a prompt 'Appuyez sur une touche pour continuer...'. A white box with the text 'Expliquez le résultat affiché par python' is overlaid on the command prompt window.

**Correction Exercices Python Janv 1**

1. C'est une fonction récursive et il manque la condition d'arrêt :

```
def pyramide_inversee(n):  
    if n >= 0:  
        print(' '*(20-n) + '*'*(2*n+1))  
        pyramide_inversee(n-1)
```

On peut aussi rajouter `if n == -1: return` en première ligne de la fonction.

2. On rajoute une assertion :

```
def saute(troupeau1: list, troupeau2: list):  
    mouton = troupeau1.pop()  
    assert mouton is not None # ou assert mouton != None  
    troupeau2.append(mouton)
```

3. `n/2` est un flottant et les indices des tableaux doivent être des entiers. On peut corriger le bug en écrivant `n//2`.

4. On remplace **or** par **and**.

5. Il y a un problème d'indentation : le `return False` étant dans la boucle, l'exécution du code va s'interrompre après avoir seulement examiné le premier élément de la liste.

6. On ne doit jamais tester l'égalité entre deux flottant car le codage interne d'un nombre décimal en binaire entraîne en général une erreur d'arrondi. Cf le cours sur le transcodage des nombres décimaux en binaire.