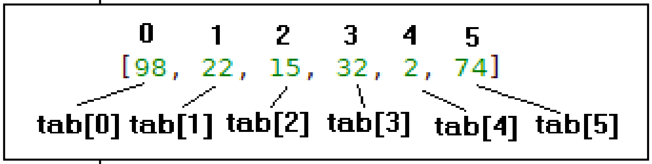
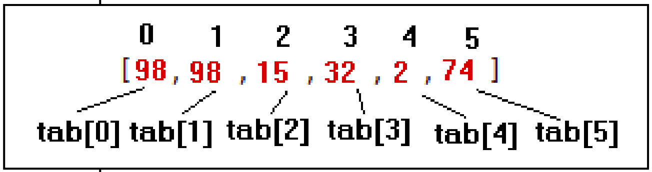
TD : Tri par insertion : les étapes du programme

```
def tri_insertion(tab):
    # parcours de 1 à longueur du tableau

    for i in range(1, len(tab)):
        1               6
        k = tab[i]
                1
             22
        j = i-1
            1
        0
        while j >= 0 and k< tab[j] : Vraie
              0          22    98
            tab[j+1] = tab[j]
               98        98
            j = j-1
           -1   0
        tab[j+1] = k
```

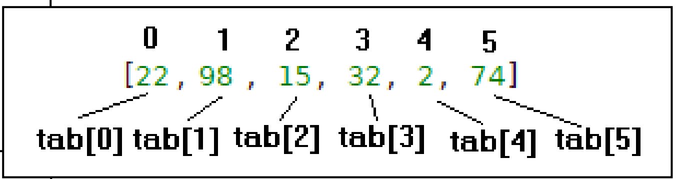




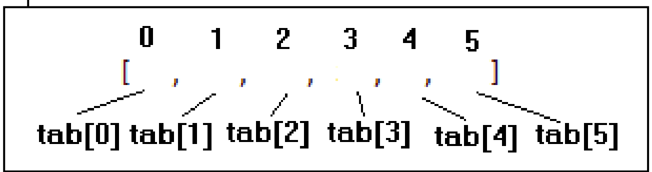

```
def tri_insertion(tab):
    # parcours de 1 à longueur du tableau

    for i in range(1, len(tab)):
        1               6
        k = tab[i]
                1
             22
        j = i-1
            1
        0
        while j >= 0 and k< tab[j] : Faux
             -1          22    -1
            tab[j+1] = tab[j]   pas exécuté

            j = j-1         pas exécuté
        on sort de la boucle
             -1
        tab[j+1] = k
             0      22
```

Compléter le tableau tab chaque fois qu'il est modifié dans le programme.

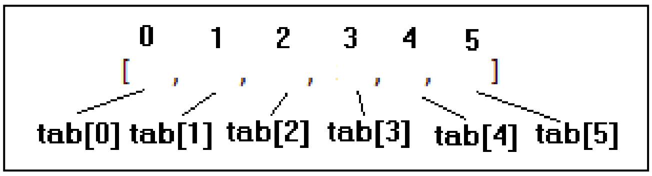
On notera que **tab[-1] ne plante pas** en python et n'engendre pas ici d'effet de bord mais ce n'est pas le cas dans d'autres langages qui interdisent les indices négatifs …



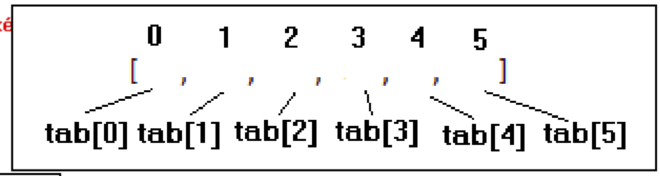

```
def tri_insertion(tab):
    # parcours de 1 à longueur du tableau

    for i in range(1, len(tab)):
        2
                2
        k = tab[i]
            15

        j = i-1
        1   2
        while j >= 0 and k< tab[j] : Vraie
              1                1
              1         1
            tab[j+1] = tab[j]
                2        98
            j = j-1
            0   1
        tab[j+1] = k
```



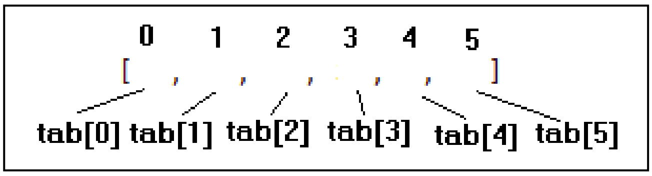

```
def tri_insertion(tab):
    # parcours de 1 à longueur du tableau

    for i in range(1, len(tab)):

        k = tab[i]

        j = i-1

        while j >= 0 and k< tab[j] :

            tab[j+1] = tab[j]

            j = j-1

        tab[j+1] = k
```



```python
def tri_insertion(tab):
    # parcours de 1 à longueur du tableau
    for i in range(1, len(tab)):

        k = tab[i]

        j = i-1

        while j >= 0 and k< tab[j] : Faux
              -1              -1
            tab[j+1] = tab[j]  pas exé

            j = j-1  pas exécuté

        tab[j+1] = k
```

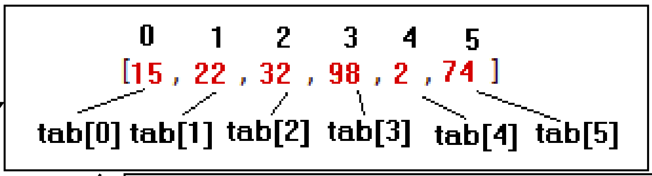


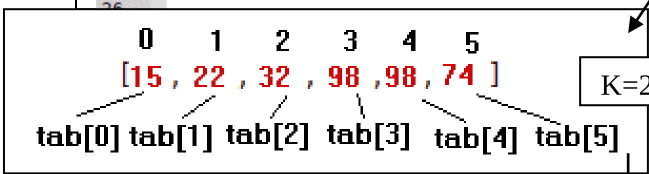

```python
def tri_insertion(tab):
    # parcours de 1 à longueur du tableau
    for i in range(1, len(tab)):

        k = tab[i]

        j = i-1

        while j >= 0 and k< tab[j] :

            tab[j+1] = tab[j]

            j = j-1

        tab[j+1] = k
```



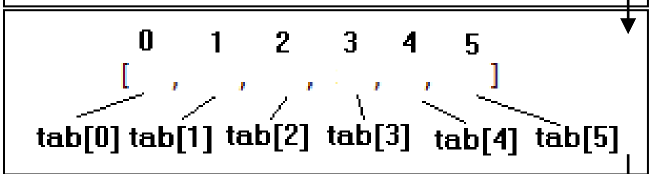

```python
def tri_insertion(tab):
    # parcours de 1 à longueur du tableau
    for i in range(1, len(tab)):

        k = tab[i]

        j = i-1

        while j >= 0 and k< tab[j] :

            tab[j+1] = tab[j]

            j = j-1

        tab[j+1] = k
```

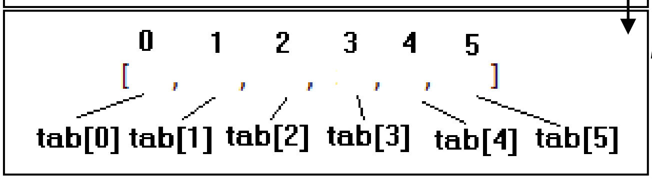
A ce stade vous devriez avoir compris le déroulement de l'algorithme, compléter les modifications du tableau jusqu'à la fin du tri

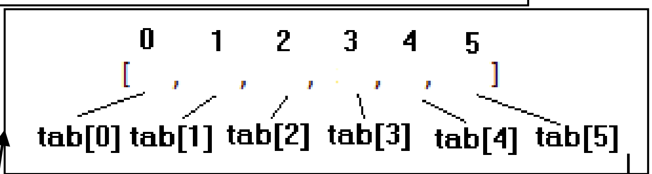


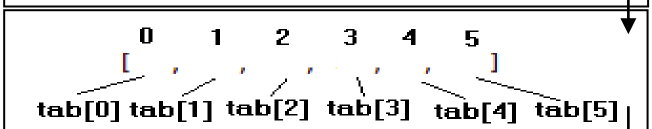


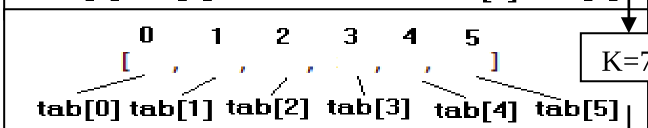
K=2











K=74