

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

# Exercices tris Correction

Christophe Viroulaud

Première - NSI

**Algo 05**

# Sommaire

1. Exercice 1

Exercice 1

2. Exercice 2

Exercice 2

3. Exercice 3

Exercice 3

4. Exercice 4

Exercice 4

5. Exercice 5

Exercice 5

6. Exercice 6

Exercice 6

7. Exercice 7

Exercice 7

8. Exercice 8

Exercice 8

# Exercice 1

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

carré du nombre éléments	$16000^2$	$1000000^2$
durée	6,8	

$$\frac{6,8 \times 1000000^2}{16000^2} = 26560s = 7h23min$$

# Sommaire

1. Exercice 1

2. Exercice 2

3. Exercice 3

4. Exercice 4

5. Exercice 5

6. Exercice 6

7. Exercice 7

8. Exercice 8

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

## Exercise 2

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

```
1 def tri_insertion(tab: list) -> None:
2     for i in range(len(tab)):
3         # mémoriser
4         en_cours = tab[i]
5         pos = i
6         # décaler
7         while pos > 0 and en_cours < tab[pos-1]:
8             tab[pos] = tab[pos-1]
9             pos = pos-1
10        # insérer
11        tab[pos] = en_cours
```

```
1 tab = [randint(0, 100) for _ in range(10)]
2 tri_insertion(tab)
3 print(tab)
```

# Sommaire

1. Exercice 1

2. Exercice 2

3. Exercice 3

4. Exercice 4

5. Exercice 5

6. Exercice 6

7. Exercice 7

8. Exercice 8

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

# Exercice 3

Exercice 1

Exercice 2

Exercice 3

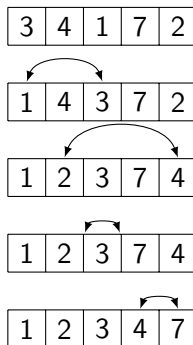
Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8



Code 1 – Tri par sélection

Exercice 1

Exercice 2

Exercice 3

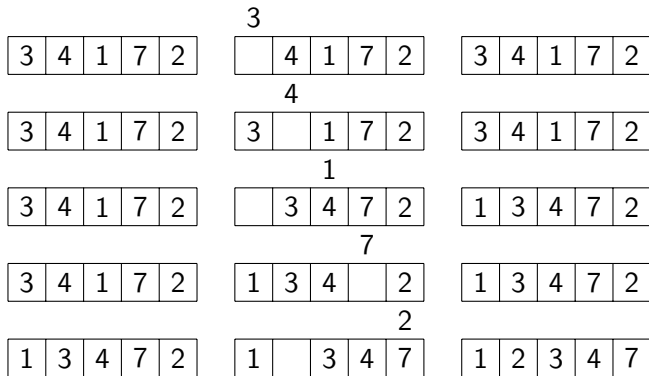
Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8



Code 2 – Tri par insertion



# Sommaire

1. Exercice 1

Exercice 1

2. Exercice 2

Exercice 2

3. Exercice 3

Exercice 3

4. Exercice 4

Exercice 4

5. Exercice 5

Exercice 5

6. Exercice 6

Exercice 6

7. Exercice 7

Exercice 7

8. Exercice 8

Exercice 8

## Exercice 4

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

```
1 def comparer(tab1: list, tab2: list) -> bool:
2     for i in range(len(tab1)):
3         if not tab1[i] == tab2[i]:
4             # stoppe à la première différence
5             return False
6     # tous les éléments ont été comparés
7     return True
```

```
1 t1 = [3, 5, 9, 0, 1, 8, 2]
2 t2 = [9, 5, 3, 2, 8, 1, 0]
3 tri_insertion(t1)
4 tri_insertion(t2)
5 print(comparer(t1, t2))
```

# Sommaire

1. Exercice 1

2. Exercice 2

3. Exercice 3

4. Exercice 4

5. Exercice 5

6. Exercice 6

7. Exercice 7

8. Exercice 8

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

# Exercice 5

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

```
1 def tri_insertion(tab: list) -> list:
2     tab_trie = []
3     for i in range(len(tab)):
4         # mémoriser
5         en_cours = tab[i]
6         tab_trie.append(en_cours)
7         pos = len(tab_trie)-1
8         # décaler
9         while pos > 0 and en_cours < tab_trie[pos-1]:
10             tab_trie[pos] = tab_trie[pos-1]
11             pos = pos-1
12         # insérer
13         tab_trie[pos] = en_cours
14     return tab_trie
```

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

```
1 t = [randint(0, 100) for _ in range(10)]
2 print(tri_insertion(t))
3 # le tableau initial n'est pas modifié
4 print(t)
```

# Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4
5. Exercice 5
6. Exercice 6
7. Exercice 7
8. Exercice 8

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

## Exercise 6

```
1 def echanger(tab: list, i: int, j: int) ->  
    None:  
2     temp = tab[i]  
3     tab[i] = tab[j]  
4     tab[j] = temp  
5  
6 def inserer(tab: list, j: int) -> None:  
7     # Le changement se fait dans la  
    comparaison  
8     while j-1 >= 0 and tab[j-1][0] > tab[j  
    ][0]:  
9         echanger(tab, j-1, j)  
10        j = j-1  
11  
12 def tri_insertion(tab: list) -> None:  
13     for i in range(len(tab)):  
14         inserer(tab, i)
```

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

# Sommaire

1. Exercice 1

2. Exercice 2

3. Exercice 3

4. Exercice 4

5. Exercice 5

6. Exercice 6

7. Exercice 7

8. Exercice 8

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8



```
1 def max_occurrences(tab: list) -> tuple:
2     tri_insertion(tab)
3     # départ 1° série
4     en_cours = tab[0]
5     serie_en_cours = 1
6     elt_max = tab[0]
7     serie_max = 1
8     for i in range(1, len(tab)):
9         # cas même élément que le précédent
10        if en_cours == tab[i]:
11            serie_en_cours += 1
12        else:
13            # vérifie alors la taille de la dernière série
14            if serie_en_cours > serie_max:
15                serie_max = serie_en_cours
16                elt_max = en_cours
17            # départ nouvelle série
18            en_cours = tab[i]
19            serie_en_cours = 1
20    return (elt_max, serie_max)
```

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

```
1 t = [randint(0, 10) for _ in range(100)]  
2 maxi = max_occurrences(t)  
3 print(f"Le nombre {maxi[0]} est apparu {maxi[1]} fois.")
```

# Sommaire

1. Exercice 1

2. Exercice 2

3. Exercice 3

4. Exercice 4

5. Exercice 5

6. Exercice 6

7. Exercice 7

8. Exercice 8

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

# Exercice 8

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

```
1 def echanger(tab: list, i: int, j: int) -> None:
2     """
3     échange deux éléments de tab
4     """
5     temp = tab[i]
6     tab[i] = tab[j]
7     tab[j] = temp
```

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

Exercice 7

Exercice 8

```
1 def tri_bulles(tab: list) -> None:
2     for i in range(len(tab)):
3         # on s'arrête avant la partie déjà triée
4         for j in range(1, len(tab)-i):
5             # remonte l'élément le plus grand
6             if tab[j-1] > tab[j]:
7                 echanger(tab, j-1, j)
```