

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

# Exercices pile - file correction

Christophe Viroulaud

Terminale - NSI

**Archi 06**

1. Exercice 1

2. Exercice 2

3. Exercice 3

4. Exercice 4

5. Exercice 5

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

# Exercise 1

Exercise 1

Exercise 2

Exercise 3

Exercise 4

Exercise 5

```
1 def creer_pile() -> list:
2     return []
3
4 def est_vide(p: list) -> bool:
5     return len(p) == 0
6
7 def empiler(p: list, e: int) -> None:
8     p.append(e)
9
10 def depiler(p: list) -> int:
11     if not est_vide(p):
12         return p.pop()
13
14 p = creer_pile()
```

Code 1 – pile

```
1 def creer_file() -> list:
2     return []
3
4 def est_vide(f: list) -> bool:
5     return len(f) == 0
6
7 def enfiler(f: list, e: int) -> None:
8     f.insert(0, e)
9
10 def defiler(f: list) -> int:
11     if not est_vide(f):
12         return f.pop()
13
14 f = creer_file()
```

Code 2 – file

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

La modification de la taille d'un tableau a un coup qui peut être linéaire.

# Sommaire

1. Exercice 1

2. Exercice 2

3. Exercice 3

4. Exercice 4

5. Exercice 5

Exercices pile - file  
correction

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

## Exercice 2

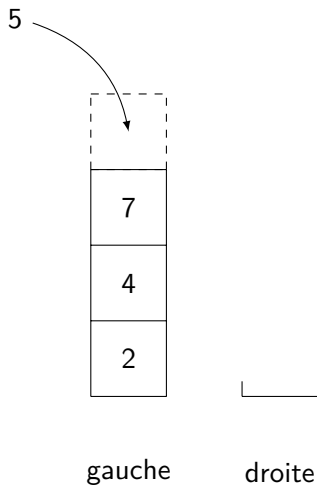


FIGURE 1 – enfiler

Le premier entré est 2.

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

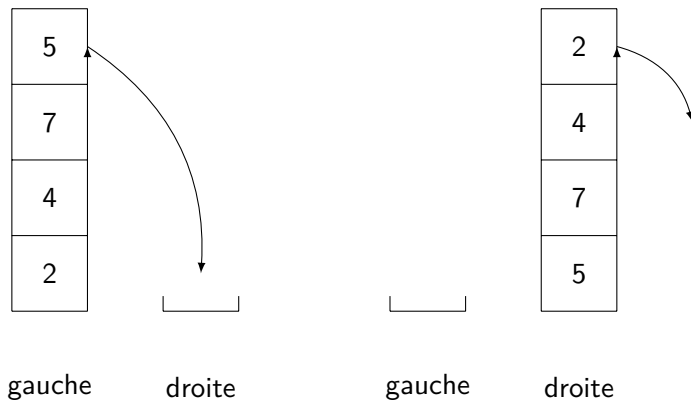


FIGURE 2 – défiler - cas 1

La pile droite est vide, on commence par dépiler celle de gauche.



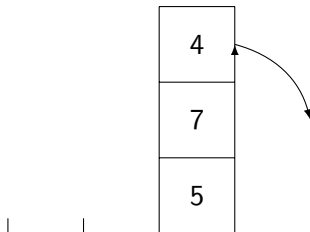
Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5



gauche

droite

FIGURE 3 – défiler - cas 2

La pile droite n'est pas vide. On dépile normalement.

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

```
1 class File2:
2     def __init__(self):
3         self.gauche = Pile()
4         self.droite = Pile()
5
6     def est_vide(self) -> bool:
7         return self.gauche.est_vide() and self.
            droite.est_vide()
```

```
1 def enfiler(self, e: int) -> None:
2     self.gauche.empiler(e)
3
4 def defiler(self) -> int:
5     if self.droite.est_vide():
6         while not self.gauche.est_vide():
7             self.droite.empiler(self.gauche.defiler())
8
9     return self.droite.defiler()
```

# Sommaire

1. Exercice 1

2. Exercice 2

3. Exercice 3

4. Exercice 4

5. Exercice 5

Exercices pile - file  
correction

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

## Exercice 3

```
1  # Création du cercle
2  soldats = File()
3
4  for i in range(1, 42):
5      soldats.enfiler(i)
6
7  # Élimination tous les 3
8  while not(soldats.est_vide()):
9      # les non-éliminés reviennent dans la
      file
10     for _ in range(2):
11         soldats.enfiler(soldats.defiler())
12
13     # soldat éliminé
14     elimine = soldats.defiler()
15
16 # dernier éliminé
17 print(elimine)
```

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

# Sommaire

1. Exercice 1

2. Exercice 2

3. Exercice 3

4. Exercice 4

5. Exercice 5

Exercices pile - file  
correction

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

## Exercise 4

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

```
1 def bien_parenthesee(code: str) -> bool:
2     parentheses = Pile()
3     i = 0
4
5     while i < len(code):
6         # empile une ouvrante
7         if code[i] == "(":
8             parentheses.empiler("(")
9         # dépile une ( quand on trouve une )
10        elif code[i] == ")":
11            # si rien à dépiler -> mauvais parenthé
12            sage
13            if parentheses.depiler() is None:
14                return False
15
16            i += 1
17
18        # si la pile n'est pas vide: il reste des (
19        return parentheses.est_vide()
```

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

```
1 def bien_parenthesee_rec(code: str, i: int, p: Pile)
  -> bool:
2     if i == len(code):
3         # si la pile n'est pas vide: il reste des (
4         return p.est_vide()
5     else:
6         if code[i] == "(":
7             p.empiler("(")
8         elif code[i] == ")":
9             if p.depiler() is None:
10                 return False
11         return bien_parenthesee_rec(code, i+1, p)
```



# Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4
5. Exercice 5

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

## Exercise 5

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 5

```
1 def polonaise(chaine: str) -> int:
2     p = Pile()
3     for e in chaine.split():
4         if e == "+" or e == "*":
5             val1 = p.depiler()
6             val2 = p.depiler()
7             if e == "+":
8                 p.empiler(val1+val2)
9             else:
10                p.empiler(val1*val2)
11         else:
12             p.empiler(int(e))
13
14     return p.depiler()
```