Exercices pile - file correction

Christophe Viroulaud

Terminale - NSI

Archi 06

Sommaire

Exercice 1

```python
def creer_pile() -> list:
    return []

def est_vide(p: list) -> bool:
    return len(p) == 0

def empiler(p: list, e: int) -> None:
    p.append(e)

def depiler(p: list) -> int:
    if not est_vide(p):
        return p.pop()

p = creer_pile()
```

Code 1 – pile

```python
def creer_file() -> list:
    return []

def est_vide(f: list) -> bool:
    return len(f) == 0

def enfiler(f: list, e: int) -> None:
    f.insert(0, e)

def defiler(f: list) -> int:
    if not est_vide(f):
        return f.pop()

f = creer_file()
```

Code 2 – file

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

Sommaire

Exercice 2

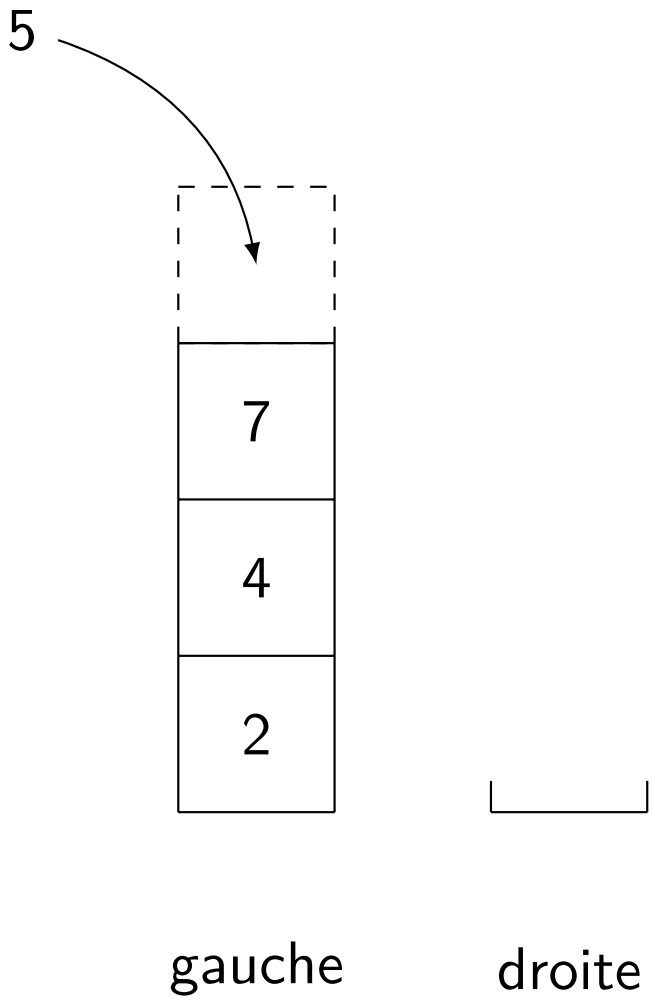

FIGURE 1 – enfiler

Le premier entré est 2.

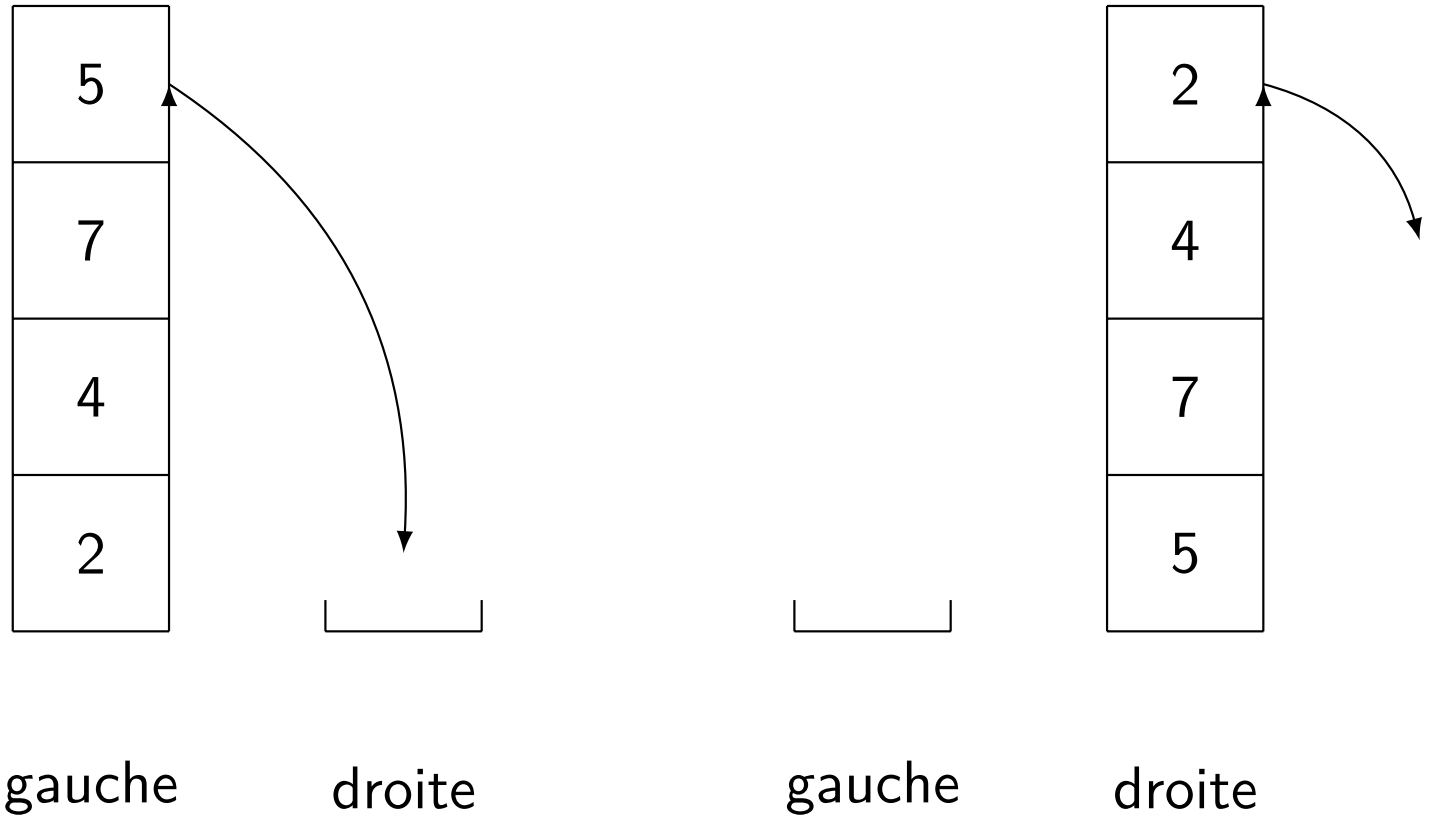

FIGURE 2 – défiler - cas 1

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

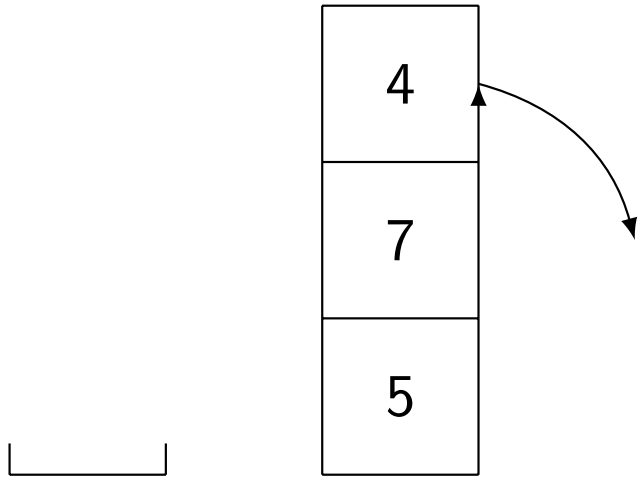

FIGURE 3 – défiler - cas 2

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

```
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.depiler())
8
9      return self.droite.depiler()
```

Sommaire

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)
```

Sommaire

Exercice 4

```python
def bien_parenthesee(code: str) -> bool:
    parentheses = Pile()

    for car in code:
        if car == "(":
            # empile une "("
            parentheses.empiler("(")
        elif car == ")":
            # dépile une "(" quand on trouve une ")"
            if parentheses.est_vide():
                # pile vide = manque une "("
                return False
            else:
                parentheses.depiler()

    # si la pile n'est pas vide: il reste des (
    return parentheses.est_vide()
```

```
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.est_vide():
0                  # pile vide = manque une "("
1                  return False
2              else:
3                  p.depiler()
4          return bien_parenthesee_rec(code, i+1, p)
```

Sommaire

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:
0                  p.empiler(val1*val2)
1          else:
2              p.empiler(int(e))
3
4      return p.depiler()
```