V - TP 3

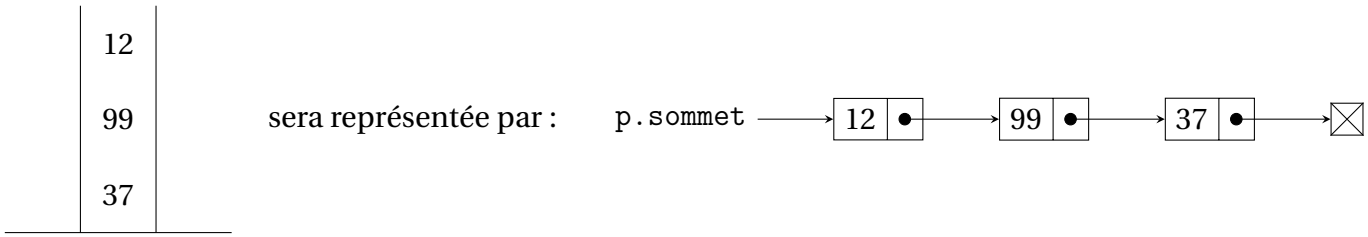
Implémentation

1 Pile

Dans cette partie, on implémente l'interface du type `Pile` à l'aide d'objets de type `Maillon`. Une pile `p` possède un unique attribut, `sommet` :

- celui-ci vaut `None` dans le cas où la pile est vide ;
- celui-ci est le premier `Maillon` de la chaîne dans le cas où la pile n'est pas vide.

Les opérations d'empilement/dépilement reviennent donc à ajouter/supprimer le premier maillon de la chaîne.



Code python

```
class Maillon:
    def __init__(self, v, n):
        self.valeur = v
        self.suivant = n

class Pile:
    """Une pile d'entiers"""
    def __init__(self):
        """ Pile -> Nonetype """
        self.sommet = None

```

1.1 Méthode `est_vide`

Écrire la méthode `est_vide` de la classe `Pile` qui renvoie `True` si et seulement si la pile `self` est vide. On rappelle qu'on a pris comme convention que l'attribut `sommet` d'un objet de type `Pile` vaut `None` dans le cas de la liste vide.

Code python

```
def est_vide(self):
    """ Pile -> bool
    Détermine si la pile est vide """
    pass
```

Code python

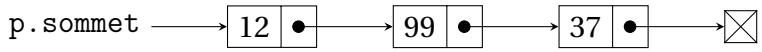
```
p = Pile()
print(p.est_vide())
```

Résultat

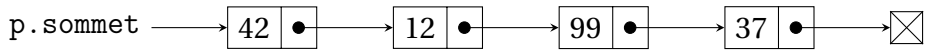
```
True
```

1.2 Méthode `empiler`

Écrire la méthode `empiler` de la classe `Pile` qui ajoute la valeur `v` au sommet de la pile `self`. Il suffit pour cela de changer le maillon de tête : si la pile correspond à la chaîne suivante :



alors après l'exécution de l'instruction `p.empiler(42)` elle doit correspondre à la chaîne :



Code python

```
def empiler(self, v):
    """ Pile, int -> Nonetype
    Empile la valeur v au sommet de la pile self """
    pass
```

Code python

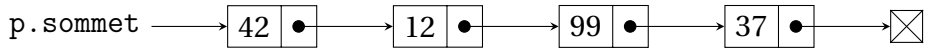
```
p = Pile()
p.empiler(37)
print(p.sommet.valeur)
```

Résultat

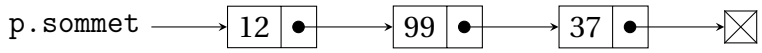
```
37
```

1.3 Méthode `depiler`

Écrire la méthode `depiler` de la classe `Pile` qui renvoie la valeur de l'élément présent au sommet de la pile, en le supprimant de celle-ci.
Par exemple, si initialement la pile `p` correspond à la chaîne :



alors l'intruction `p.depiler()` renvoie 42. De plus, à l'issue de l'exécution de cette instruction, la chaîne correspondant à `p` est :



Attention. Dans le cas où la pile `self` est vide, on *soulèvera une exception* à l'aide de l'instruction `raise IndexError("Impossible de dépiler : la pile est vide")`. Une exception est une instruction qui, si elle n'est pas traitée, arrête l'exécution du programme et affiche sur la sortie standard un message.

Code python

```
def depiler(self):
    """ Pile -> int
    Renvoie l'élément présent au sommet de la pile self, en le supprimant
    ↪ de la pile """
    pass
```

Code python

```
p = Pile()
p.empiler(42)
print(p.depiler())
print(p.depiler())
```

Résultat

```
42
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call
↪ last)
Input In [15], in <cell line: 4>()
      2 p.empiler(42)
      3 print(p.depiler())
----> 4 print(p.depiler())

Input In [1], in Pile.depiler(self)
     23 """ Pile -> int
     24 Renvoie l'élément présent au sommet de la pile self,
     25 en le supprimant de la pile """
     26 if self.est_vide():
---> 27     raise IndexError("Impossible de dépiler la pile vide")
     28 val = self.sommet.valeur
     29 self.sommet = self.sommet.suivant

IndexError: Impossible de dépiler la pile vide
```

1.4 Affichage d'une pile

Écrire le code de la méthode spéciale `__str__` afin que, lorsque l'on exécute l'instruction `print(p)` une représentation compréhensible de la pile soit affichée sur la sortie standard.
Pour cela on initialisera une chaîne de caractère `s` commençant par `"[Sommet]"` et on parcourra tous les maillons de la chaîne, en ajoutant successivement la chaîne de caractère représentant la valeur du maillon courant à `s` (on utilisera des espaces pour séparer les valeurs entres elles).

Code python

```python
def __str__(self):
    """ Pile -> str
    Construit la chaîne de caractère représentant la pile self """
    pass
```

Code python

```python
p = Pile()
p.empiler(42)
p.empiler(24)
print(p)
```

Résultat

```
[Sommet] 24 42
```

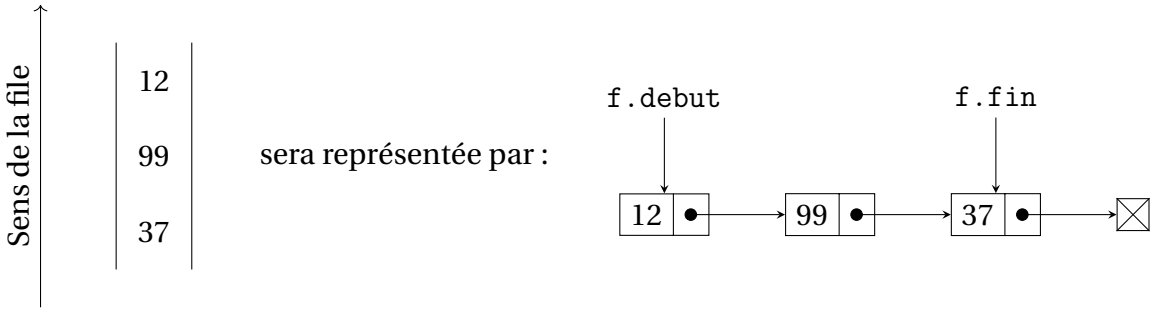
Question 1. **1.** Quelle est la complexité de la méthode `__str__` en fonction du nombre n d'éléments présents dans la pile `self` ?

2. La méthode `__str__` mute-t-elle l'objet auquelle elle s'applique ?

2 File

Dans cette partie, on implémente l'interface du type `File` à l'aide d'objets de type `Maillon`. Une file `f` possède deux attributs :

- l'attribut `debut` qui fait référence au début de la file ;
- l'attribut `fin` qui fait référence à la fin de la file.



Enfiler un élément revient donc à ajouter un maillon à la fin de la chaîne, défiler revient à supprimer le maillon au début de la chaîne, en en renvoyant sa valeur. La file vide est caractérisée par le fait qu'elle ne contient aucun maillon et donc que ses attributs `debut` et `fin` valent tous deux `None`.

Code python

```
class File:
    """Une file d'entiers"""
    def __init__(self):
        """File -> Nonetype"""
        self.debut = None
        self.fin = None

```

2.1 Méthode `est_vide`

Écrire la méthode `est_vide` de la classe `File` qui renvoie `True` si et seulement si la pile `self` est vide. On rappelle qu'une file `f` est vide si et seulement si ses attributs `debut` et `fin` sont valent `None`.

Code python

```
def est_vide(self):
    """ File -> bool
    Détermine si la file self est vide """
    pass
```

Code python

```
f = File()
print(f.est_vide())
```

Résultat

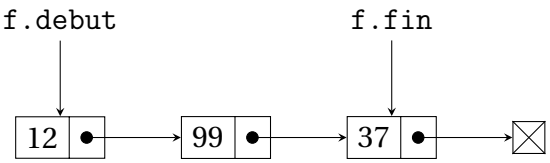
```
True
```

2.2 Méthode `enfiler`

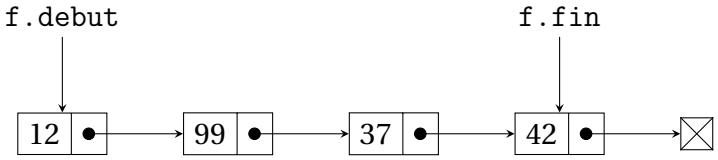
Écrire la méthode `enfiler` de la classe `File` qui ajoute la valeur `v` à la fin de la file `self`. Deux cas sont possibles :

- soit la file `self` est vide, dans ce cas on met à jour `self.debut` et `self.fin` afin qu'ils fassent référence au même maillon de valeur `v` ;
- sinon, on change l'attribut `suivant` du dernier maillon de la chaîne pour qu'il fasse référence au maillon de valeur `v` et on met à jour l'attribut `self.fin` pour qu'il fasse référence au maillon de valeur `v`.

Si la file `f` correspond à la chaîne suivante :



Alors après exécution de `f.enfiler(42)` elle doit correspondre à la chaîne :



Code python

```python
def enfiler(self, v):
    """ File, int -> Nonetype
    Ajoute l'élément v à la file self """
    pass
```

Code python

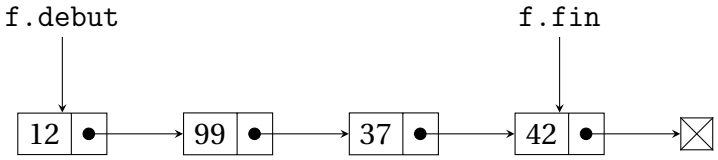
```python
f = File()
f.enfiler(42)
print(f.debut.valeur)
print(f.fin.valeur)
```

Résultat

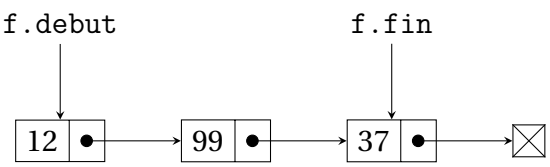
```
42
42
```

2.3 Méthode `defiler`

Écrire la méthode `defiler` de la classe `File` qui renvoie la valeur du premier élément de la file, en le supprimant de celle-ci.
Si la file `f` correspond à la chaîne :



Alors l'instruction `f.defiler()` doit renvoyer 12. De plus, à l'issue de l'exécution de cette instruction, la chaîne correspondant à `f` est :



Attention. Si la file est initialement vide, on soulèvera l'exception correspondante. Si la file est vide *après défilement* on s'assurera de modifier l'attribut `fin` de la file à `None` (car une file vide n'est constituée d'aucun maillon).

Code python

```python
def defiler(self):
    """ File -> int
    Renvoie le premier élément de la file en le supprimant de celle-ci """
    pass
```

Code python

```python
f = File()
f.enfiler(42)
print(f.defiler())
print(f.defiler())
```

Résultat

```
42
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call
 ↪  last)
Input In [79], in <cell line: 4>()
      2 f.enfiler(42)
      3 print(f.defiler())
----> 4 print(f.defiler())

Input In [76], in File.defiler(self)
     24 """ File -> int
     25 Renvoie le premier élément de la file en le supprimant de
      ↪  celle-ci """
     26 if self.est_vide():
---> 27     raise IndexError("Impossible de défiler la file vide")
     28 val = self.debut.valeur
     29 self.debut = self.debut.suivant

IndexError: Impossible de défiler la file vide
```

2.4 Affichage d'une file

On souhaite écrire la méthode spéciale `__str__` afin que, lorsque l'on exécute l'instruction `print(f)` une représentation compréhensible de la file soit affichée sur la sortie standard.
Pour cela on initialisera une chaîne de caractère `s` commençant par `"[Début]"` et on parcourra tous les maillons de la chaîne, en ajoutant successivement la chaîne de caractère représentant la valeur du maillon courant à `s` (on utilisera des espaces pour séparer les valeurs entres elles). On ajoutera `"[Fin]"` à `s` avant de la renvoyer.

Code python

```python
def __str__(self):
    """ self -> str
    Construit la chaîne de caractères représentant la file self """
    pass
```

Code python

```
f = File()
f.enfiler(11)
f.enfiler(17)
print(f)
```

Résultat

```
[Début] 11 17 [Fin]
```

Question 2. **1.** Quelle est la complexité de la méthode `__str__` en fonction du nombre n d'éléments présents dans la file `self` ?

2. La méthode `__str__` mute-t-elle l'objet auquelle elle s'applique ?

3 Piles et files bornées

On souhaite implémenter la structure de donnée "pile bornée" : il s'agit d'une structure de donnée qui possède la même interface que celle d'une pile, mais est de plus dotée d'une contenance maximale. Ainsi, il doit être impossible d'empiler un élément si la pile est remplie.
L'interface d'une pile bornée est la suivante :

Fonction	Signature	Description
`Pile(n)`	`int -> Pile`	Renvoie une pile bornée vide de capacité maximale `n`
`p.est_vide()`	`() -> Bool`	Détermine si la pile est vide.
`p.est_pleine()`	`() -> Bool`	Détermine si la pile a atteint sa capacité maximale.
`p.empiler(e)`	`int -> Nonetype`	Empile l'élément `e` au sommet de la pile `p`. Si cela n'est pas possible, soulève une erreur.
`p.depiler()`	`() -> int`	Renvoie la valeur de l'élément au sommet de la pile, en le supprimant. Si cela n'est pas possible, soulève une erreur.

Question 3. Vous écrirez et **testerez** le code d'une classe `PileBornee` implémentant ces fonctionnalités.
Les instances de cette classe possèderont trois attributs :

- un attribut `pile`, instance de la classe `Pile` ; On pourra penser à utiliser trois attributs : le premier étant une pile (classique), le second
- un attribut `contenance`, de type `int` (le nombre maximum d'éléments que peut contenir la `pile`) ;
- un attribut `n`, de type `int` (le nombre d'éléments actuellements présents dans la `pile`).

On écrira les méthodes de la classe `PileBornee` en faisant appel à celle de la classe `Pile` et en vérifiant qu'une opération d'empilement ne provoque pas un dépassement de capacité de la `pile`.
À chaque opération d'empilement et de dépilement, on mettra à jour la valeur de l'attribut `n`.

Question 4. Écrire l'interface d'une classe `FileBornee`, l'implémenter et la tester.