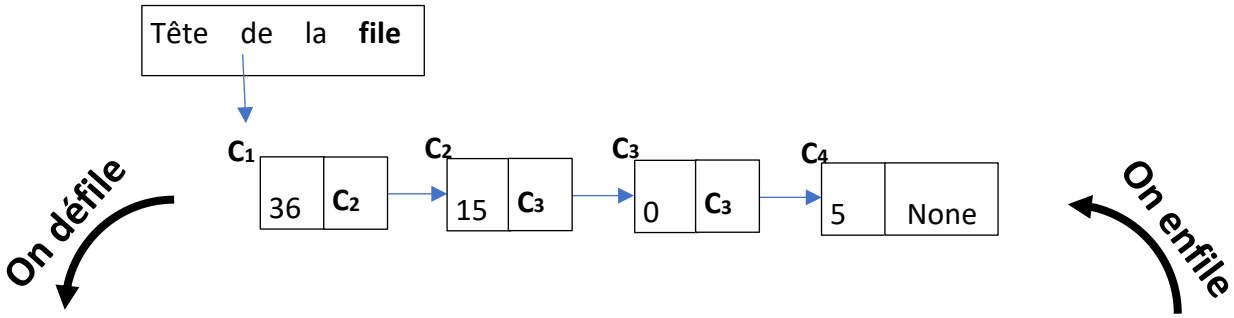
Ds B Piles et Files- correction pages 6-7-8

Exercice 1. : Implémentation d'une **FILE** en POO avec une structure du type « liste chainée »

La file F présentée ci-dessous contient les valeurs 36 , 15 , 0 et 5 . Ces valeurs sont les attributs *val* d'objets de la classe *Cellule*. La file F est une instance de la classe *File*.

La valeur 36 a été enfilée en premier. Ensuite ont été enfilés dans l'ordre les valeurs 15, puis 0 et enfin 5. Si l'on défile cette file, la valeur 36 est retirée et est renvoyée.



On donne ci-dessous une implémentation python incomplète des classes *Cellule* et *File* :

```
class Cellule :
    def __init__(self,val=None):
        self.val = val
        self.suivant  = None

class File :
    def __init__(self):
        self.tete = None


    def __str__(self) :
        message = "Etat de la file :  "
        c = self.tete
        while c != None :
            message += str(c.val) + " "
            c = c.suivant
        return message
```

On vous demande de créer, dans l'ordre, les méthodes suivantes de la classe File :

- *enfiler()* qui permet d'enfiler une valeur mise en argument en queue de la liste chainée
- *taille()* qui renvoie le nombre d'éléments de la file
- *est_vide()* qui renvoie *True* si la liste est vide, *False* sinon
- *defiler()* qui retire de la file la valeur qui est en tête de liste chainée et renvoie cette valeur

Ainsi avec le programme principal suivant :

```
#Main
F = File()
print(F)
print(f"File vide ? : {F.est_vide()}")
for elt in [36,15,0,5] :
    F.enfiler(elt)
    print(F)
print(f"Taille de la file : {F.taille()}")
print(f"Valeur défilée : {F.defiler()}")
print(F)
print(f"File vide ? : {F.est_vide()}")
```

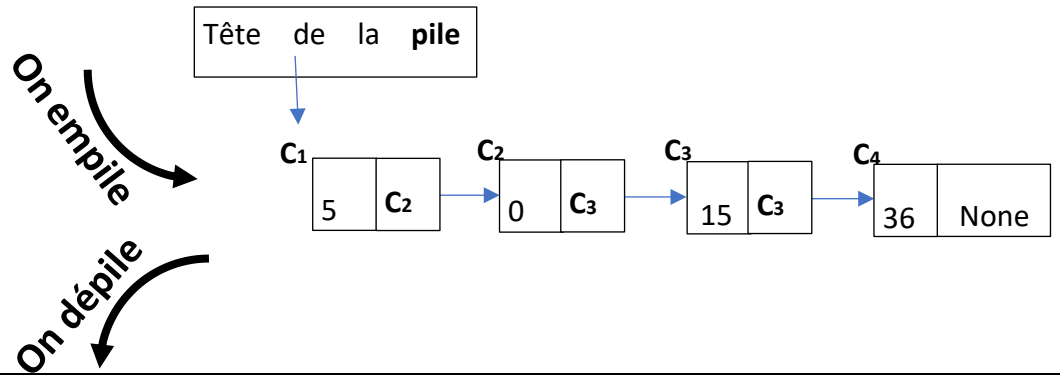
le résultat de l'exécution sera :

```
>>> (executing file "exercice1.py")
Etat de la file :
File vide ? : True
Etat de la file :  36
Etat de la file :  36 15
Etat de la file :  36 15 0
Etat de la file :  36 15 0 5
Taille de la file : 4
Valeur défilée : 36
Etat de la file :  15 0 5
File vide ? : False
```

Exercice 2. : Implémentation d'une **PILE** en POO avec une structure du type « liste chainée »

La pile P présentée ci-dessous contient les valeurs 36 , 15 , 0 et 5 . Ces valeurs sont les attributs *val* d'objets de la classe *Cellule*. La pile P est une instance de la classe *Pile*.

La valeur 36 a été enfilée en premier. Ensuite ont été enfilés dans l'ordre les valeurs 15, puis 0 et enfin 5. Si l'on défile cette file, la valeur 36 est retirée et est renvoyée.



On donne ci-dessous une implémentation python incomplète des classes *Cellule* et *Pile* :

```
class Cellule :
    def __init__(self,val=None):
        self.val = val
        self.suivant  = None

class Pile :
    def __init__(self):
        self.tete = None


    def __str__(self) :
        message = "Etat de la pile :  "
        c = self.tete
        while c != None :
            message += str(c.val) + " "
            c = c.suivant
        return message
```

On vous demande de créer, dans l'ordre, les méthodes suivantes de la classe Pile :

- *empiler()* qui permet d'empiler une valeur mise en argument en tête de la liste chainée
- *taille()* qui renvoie le nombre d'éléments de la file
- *est_vide()* qui renvoie *True* si la liste est vide, *False* sinon
- *depiler()* qui retire de la pile la valeur qui est en tête de liste chainée et renvoie cette valeur

Ainsi avec le programme principal suivant :

```
# Main
P = Pile()
print(f"Pile vide ? : {P.est_vide()}")
for elt in [36,15,0,5] :
    P.empiler(elt)
    print(P)
print(f"Taille de la pile : {P.taille()}")
print(f"Valeur dépilée : {P.depiler()}")
print(P)
print(f"Pile vide ? : {P.est_vide()}")
```

le résultat de l'exécution sera :

```
>>> (executing file "exercice2.py")
Pile vide ? : True
Etat de la pile :  36
Etat de la pile :  15 36
Etat de la pile :  0 15 36
Etat de la pile :  5 0 15 36
Taille de la pile : 4
Valeur dépilée : 5
Etat de la pile :  0 15 36
Pile vide ? : False
```

Exercice 3. : Implémentation d'une **PILE** en POO en utilisant une liste Python

La pile P présentée ci-dessous contient les valeurs 36 , 15 , 0 et 5 . Ces valeurs sont les attributs *val* d'objets de la classe *Cellule*. La pile P est une instance de la classe *Pile*.

La valeur 36 a été enfilée en premier. Ensuite ont été enfilés dans l'ordre les valeurs 15, puis 0 et enfin 5. Si l'on défile cette file, la valeur 36 est retirée et est renvoyée.

On donne ci-dessous une implémentation python incomplète de la classe *Pile* :

```
class Pile :
    def __init__(self):
        self.pile = []
```

On vous demande de créer, dans l'ordre, les méthodes suivantes de la classe Pile :

- *__str__()* qui permet d'utiliser la fonction print() sur les instances de cette classe
- *empiler()* qui permet d'empiler une valeur mise en argument
- *taille()* qui renvoie le nombre d'éléments de la file
- *est_vide()* qui renvoie *True* si la liste est vide, *False* sinon
- *depiler()* qui retire de la pile la valeur et renvoie cette valeur

Ainsi avec le programme principal suivant :

```
# Main
P = Pile()
print(f"Pile vide ? : {P.est_vide()}")
for elt in [36,15,0,5] :
    P.empiler(elt)
    print(P)
print(f"Taille de la pile : {P.taille()}")
print(f"Valeur dépilée : {P.depiler()}")
print(P)
print(f"Pile vide ? : {P.est_vide()}")
```

le résultat de l'exécution sera :

```
>>> (executing file "exercice3.py")
Pile vide ? : True
Etat de la pile :  36
Etat de la pile :  36 15
Etat de la pile :  36 15 0
Etat de la pile :  36 15 0 5
Taille de la pile : 4
Valeur dépilée : 5
Etat de la pile :  36 15 0
Pile vide ? : False
```

Exercice 1

```
class Cellule :
    def __init__(self,val=None):
        self.val = val
        self.suivant  = None

class File :
    def __init__(self):
        self.tete = None


    def __str__(self) :
        message = "Etat de la file :  "
        c = self.tete
        while c != None :
            message += str(c.val) + " "
            c = c.suivant
        return message


    def enfiler(self,valeur) :
        cNew = Cellule(valeur)
        if self.tete == None : self.tete = cNew
        else :
            c = self.tete
            while c.suivant != None :
                c = c.suivant
            c.suivant = cNew


    def taille(self) :
        n = 0
        c = self.tete
        while c != None :
            n += 1
            c = c.suivant
        return n


    def defiler(self) :
        cel = self.tete
        if cel != None :
            self.tete = cel.suivant
            return cel.val
        return None


    def est_vide(self) :
        return self.tete == None

#Main
F = File()
print(F)
print(f"File vide ? : {F.est_vide()}")
for elt in [36,15,0,5] :
    F.enfiler(elt)
    print(F)
print(f"Taille de la file : {F.taille()}")
print(f"Valeur défilée : {F.defiler()}")
```

```
print(F)
print(f"File vide ? : {F.est_vide()}")
```

Exercice 2

```
class Cellule :
    def __init__(self,val=None):
        self.val = val
        self.suivant  = None

class Pile :
    def __init__(self):
        self.tete = None


    def __str__(self) :
        message = "Etat de la pile :  "
        c = self.tete
        while c != None :
            message += str(c.val) + " "
            c = c.suivant
        return message


    def empiler(self,valeur) :
        cNew = Cellule(valeur)
        cNew.suivant = self.tete
        self.tete = cNew

    def depiler(self) :
        cel = self.tete
        if cel != None :
            self.tete = cel.suivant
            return cel.val
        return None


    def est_vide(self) :
        return self.tete == None


    def taille(self) :
        n = 0
        c = self.tete
        while c != None :
            n += 1
            c = c.suivant
        return n

# Main
P = Pile()
print(f"Pile vide ? : {P.est_vide()}")
for elt in [36,15,0,5] :
    P.empiler(elt)
    print(P)
print(f"Taille de la pile : {P.taille()}")
print(f"Valeur dépilée : {P.depiler()}")
print(P)
print(f"Pile vide ? : {P.est_vide()}")
```

Exercice 3

```
class Pile :
    def __init__(self):
        self.pile = []


    def __str__(self) :
        message = "Etat de la pile :  "
        for elt in self.pile :
            message += str(elt) + " "
        return message


    def empiler(self,valeur) :
        self.pile.append(valeur)

    def depiler(self) :
        return self.pile.pop()


    def est_vide(self) :
        return self.pile == []


    def taille(self) :
        return len(self.pile)

# Main
P = Pile()
print(f"Pile vide ? : {P.est_vide()}")
for elt in [36,15,0,5] :
    P.empiler(elt)
    print(P)
print(f"Taille de la pile : {P.taille()}")
print(f"Valeur dépilée : {P.depiler()}")
print(P)
print(f"Pile vide ? : {P.est_vide()}")
```