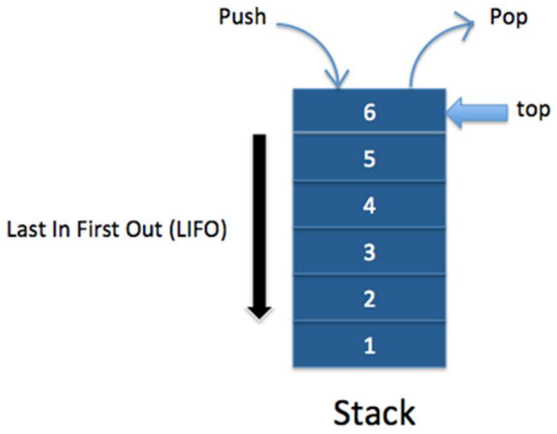
Spé NSI DS Arbres et Piles - Files

Implémentation d'une pile grâce aux listes python

On définie l'objet Pile de la manière suivante :

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

On rappelle qu'une pile obéit au principe Last Int First OUT :



1.1. Ecrivez la méthode « empiler » qui ajoute un élément « e » en haut de la pile

```
def empiler(self,e) :
    self.L.append(e)
```

1.2. Ecrivez la méthode « depiler » qui retourne l'élément en haut de la pile tout en le supprimant

```
def depiler(self) :
    return self.L.pop()
```

1.3. Ecrivez la méthode « est_vide » qui retourne vrai quand la pile est vide

```
def estVide(self) :
    return self.L== []
```

1.4. Afin de tester l'objet écrivez le bout de programme qui permet de créer une pile nommée ma_pile, d'empiler 'A', 'B', 'C' puis 'D' et de dépiler une fois en faisant afficher ce que renvoie « depiler » Donner aussi le contenu final de la pile.

```
ma_pile= Pile()
ma_pile.empiler('A')
ma_pile.empiler('B')
ma_pile.empiler('C')
ma_pile.empiler('D')
print(ma_pile.depiler())
```

2. Implémentation d'un arbre

On rappelle le code de la classe Arbre

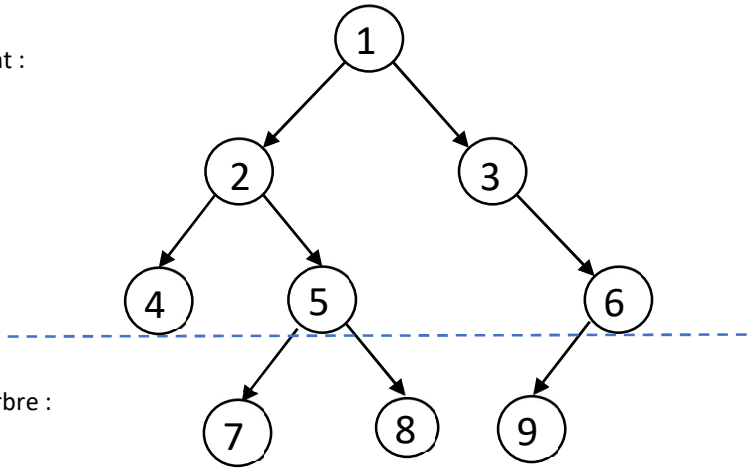
```
class Arbre :
    def __init__(self,info=None,fg=None,fd=None) :
        self.info = info
        self.fg = fg
        self.fd = fd
```

2.1. Ecrivez la méthode « fils_droit » qui ajoute un fils droit à l'arbre

```
def fils_droit(self,info) :
    if self.fd == None :
        self.fd = Arbre(info)
    else :
        new = Arbre(info)
        new.fd = self.fd
        self.fd = new
```

2.2. On suppose écrite la méthode « fils_gauche » qui ajoute un fils gauche à l'arbre.

On désire bâtir l'arbre suivant :



Donner :

- La hauteur de cet arbre :

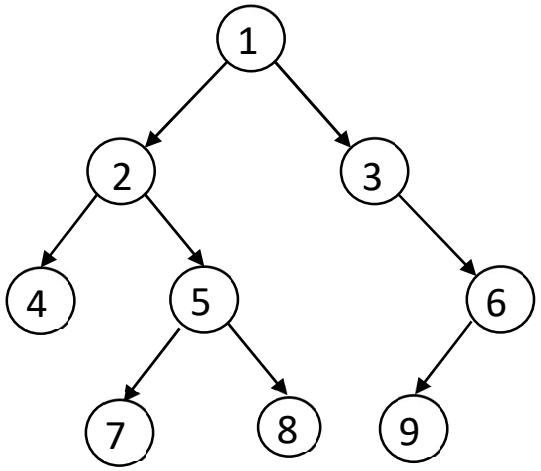
4

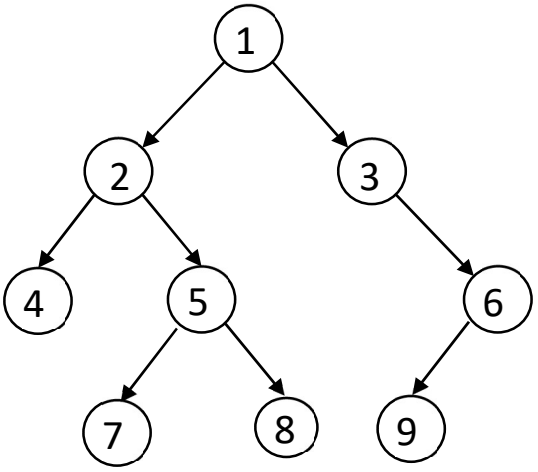
- Le parcours en profondeur postfixe de cet arbre (fils gauche, fils droit, nœud)

Vous pouvez entourer les « sous arbres »

sur lesquels s'appliqueront le parcours

de manière récursive





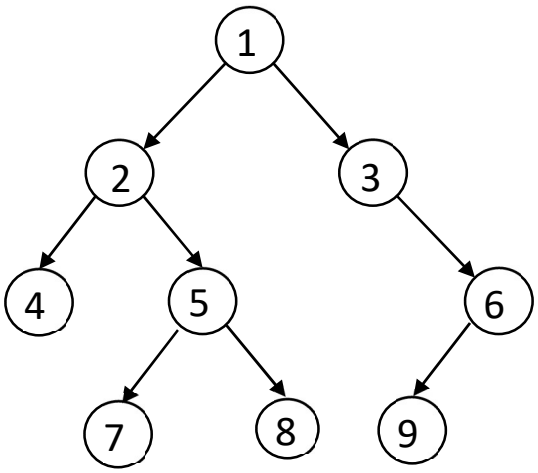
4 7 8 5 2 9 6 3 1

- Le code qui dans le programme principal crée un arbre « A » égal à celui de dessus jusqu'aux pointillés

```
B =Arbre(1)
B.insert_gauche(2)
B.fg.insert_gauche(4)
B.fg.insert_droite(5)
B.insert_droite(3)
B.fd.insert_droite(6)
```

3. On désire parcourir l'arbre en profondeur avec un parcours (préfixe) : nœud, fils gauche, fils droit

- Donner pour l'arbre de la question précédente le résultat du parcours préfixe.



1 2 4 5 7 8 3 6 9

Spé NSI DS Arbres et Piles - Files

On utilise une pile pour effectuer le traitement suivant

temps	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
Etat de la Pile	1	2 3	4 5 3	5 3	7 8 3	8 3	3	6	9	
Affichage		**1**	**2**	**4**	**5**	**7**	**8**	**3**	**6**	**9**

A T1 que met-on dans la pile ?

L'arbre de l'exemple

De T1 à T2 :

Quelle est la nature de « 1 » que possède-t-il ? Et que fait-on de « 1» Que range-t-on alors dans la pile ?

1 est un nœud qui possède fd et fg. On le dépile. On empile dans la pile le fils droit puis le fils gauche

De T2 à T3

Quelle est la nature de « 2 » que possède-t-il ? Que range-t-on alors dans la pile ? Comment vérifier cela ? Et que fait-on de « 2 »

2 est un nœud qui possède fd et fg. On le dépile. On empile dans la pile le fils droit puis le fils gauche

De T3 à T4

Quelle est la nature de « 4 » que possède-t-il ? Que range-t-on alors dans la pile ? Comment vérifier cela ? Et que fait-on de « 4 »

4 est un nœud qui ne possède pas de fd et de fg. On le dépile.

Rappel page précédente

temps	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
Etat de la Pile	1	2 3	4 5 3	5 3	7 8 3	8 3	3	6	9	
Affichage		**1**	**2**	**4**	**5**	**7**	**8**	**3**	**6**	**9**

En procédant de cette façon, on retrouve donc le résultat du parcours préfixe (nœud, fils gauche, fils droit). Utiliser cette méthode, dite itérative pour compléter le code ci-dessous de la méthode de classe ParcoursPrefixe() qui permet d'afficher un parcours Préfixe de l'arbre. Compléter les blancs. Vous disposez des méthodes pour l'objet Pile de la question 1.

```
def ParcoursPrefixe(self) :
        p = Pile()
        p.empiler(self)
        while not p.estVide()          :
            elt = p.depiler()
            print(p)
            print(f" {elt.info}  ",end='')
            if elt.fd       != None : p.empiler     (elt.fd   )
            if elt.fg       != None : p.empiler     (elt.fg   )
```