Exercice 1

Construire un arbre

Un système de fichiers contient des fichiers et des dossiers. La structure montre une hierarchie qui permet de construire un arbre.

1. Construire l'arbre du système ci-dessous :

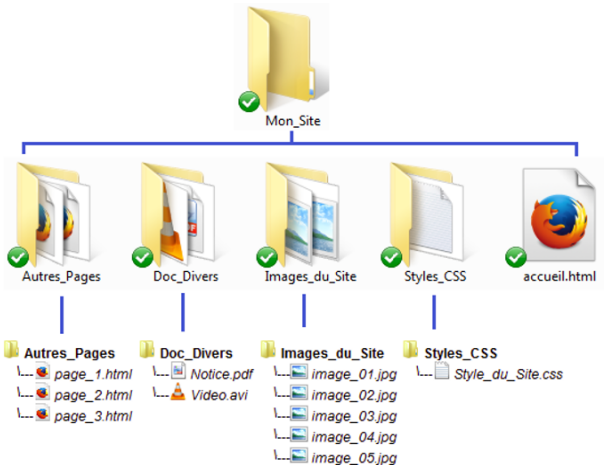


FIGURE 1 – systeme fichiers sous windows

Les noeuds fils de la racine `Mon_Site` se situent à une profondeur de 1 dans l'arbre.

2. Entourer tous les noeuds de profondeur 1.
3. Entourer tous les noeuds de profondeur 2 (utiliser une nouvelle couleur)
4. Quelle est la hauteur de cet arbre ?
5. Un arbre montre une organisation *récursive*. Entourer le sous arbre de racine `Autres_Pages`. Quelle est la hauteur de ce sous-arbre ?

Exercice 2

Parcours du système de fichiers

La fonction `fils` retourne la liste des noms des fichiers contenus dans le répertoire. Le nom est placé en paramètre. L'expression `os.listdir(p)` retourne un tableau de noms de tous les fils d'un repertoire spécifié par son chemin d'accès `p`, mais pas ses autres descendants.

```
def fils(p,L=[]):
    if os.path.isfile(p):
        return ""
    for y in os.listdir(p):
        L.append(p+'/'+y)
    return L
```

Avec le système de fichiers précédent, la fonction `fils` retourne la liste suivante :

```
>>> fils(Mon_Site)
```

```
['Mon_Site/Doc_Divers',
 'Mon_Site/Autres_Pages',
 'Mon_Site/Images_du_Site',
 'Mon_Site/Styles_CSS',
 'Mon_Site/accueil.html']
```

1. Recopier et adapter le script de **manière recursive** pour qu'il retourne la liste suivante :

```
>>> fils_recursif('Mon_Site')
['Mon_Site/Doc_Divers',
'Mon_Site/Doc_Divers/Notice.pdf',
'Mon_Site/Doc_Divers/Video.avi',
'Mon_Site/Autres_Pages',
'Mon_Site/Autres_Pages/page_1.html',
'Mon_Site/Autres_Pages/page_2.html',
'Mon_Site/Autres_Pages/page_3.html',
'Mon_Site/Images_du_Site',
'Mon_Site/Images_du_Site/image_01.jpg',
'Mon_Site/Images_du_Site/image_02.jpg',
'Mon_Site/Images_du_Site/image_03.jpg',
'Mon_Site/Images_du_Site/image_04.jpg',
'Mon_Site/Images_du_Site/image_05.jpg',
'Mon_Site/Styles_CSS',
'Mon_Site/Styles_CSS/Style_du_Site.css',
'Mon_Site/accueil.html']
```

2. On modifie légerement le script précédent, en permutant 2 lignes et on obtient alors le résultat suivant. Que s'est-il passé, quelle lignes ont été permutées ?

```
 >>> fils_recursif('Mon_Site')
['Mon_Site/Doc_Divers/Notice.pdf',
'Mon_Site/Doc_Divers/Video.avi',
'Mon_Site/Doc_Divers',
'Mon_Site/.DS_Store',
'Mon_Site/Autres_Pages/page_1.html',
'Mon_Site/Autres_Pages/page_2.html',
'Mon_Site/Autres_Pages/page_3.html',
'Mon_Site/Autres_Pages',
'Mon_Site/Images_du_Site/image_01.jpg',
'Mon_Site/Images_du_Site/image_02.jpg',
'Mon_Site/Images_du_Site/image_03.jpg',
'Mon_Site/Images_du_Site/image_04.jpg',
'Mon_Site/Images_du_Site/image_05.jpg',
'Mon_Site/Images_du_Site',
'Mon_Site/Styles_CSS/Style_du_Site.css',
'Mon_Site/Styles_CSS',
'Mon_Site/accueil.html']
```

Exercice 3

Connaissance du cours

3.1 définitions

1. Donner la définition de *hauteur* d'un arbre.
2. Donner toutes les **caractéristiques** de l'arbre suivant (taille, degré, hauteur...)

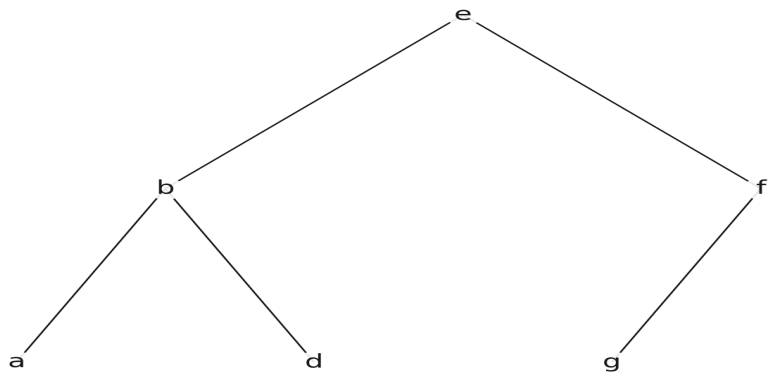


FIGURE 2 – exemple d'arbre

3. Représenter tous les arbres **binaires** de taille inférieure ou égale à 3.
4. Représenter un arbre binaire de hauteur égale à 2. (on suppose que le sommet est à la profondeur 0).
5. Quel est la taille d'un arbre binaire complet de hauteur 3 ?

Exercice 4

Parcours d'un arbre

4.1 Affichage selon le parcours

Pour l'arbre suivant :

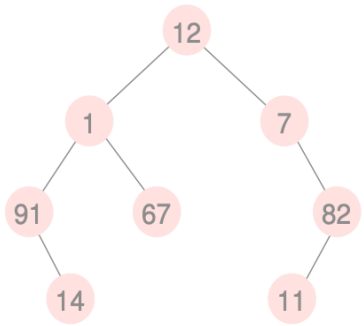


FIGURE 3 – arbre binaire

1. Dessinez le **parcours préfixe** puis donner la sortie si le traitement réalisé ne fait qu'afficher la clé. (voir dessin plus bas).

2. Dessinez le **parcours postfixe** puis donner la sortie si le traitement réalisé ne fait qu'afficher la clé.
3. Dessinez le **parcours infixe** puis donner la sortie si le traitement réalisé ne fait qu'afficher la clé.

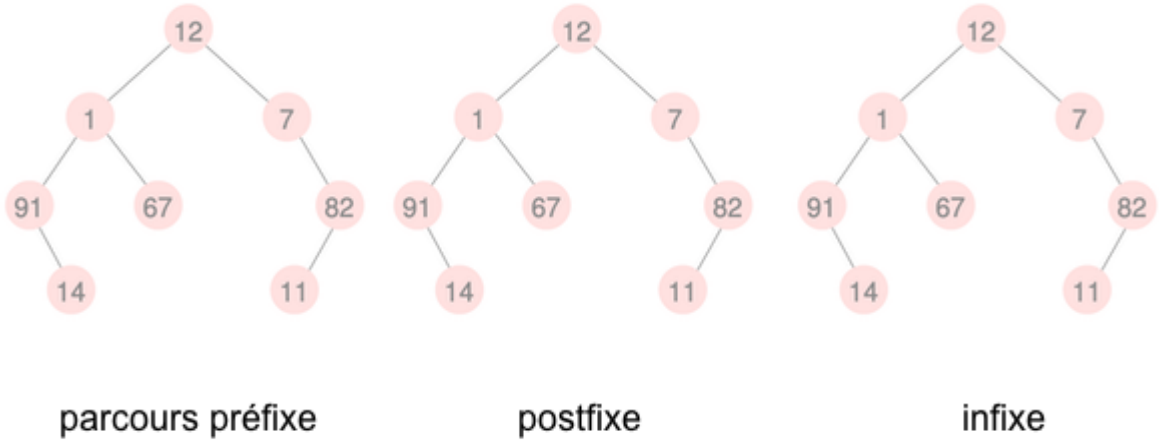


FIGURE 4 – parcours d'un arbre binaire

4. Donner la liste des clé pour un **parcours en largeur**.
5. Lequel de ces *parcours* faudra t-il choisir pour afficher les valeurs d'un arbre par profondeur croissante ?

4.2 Expression arithmetique

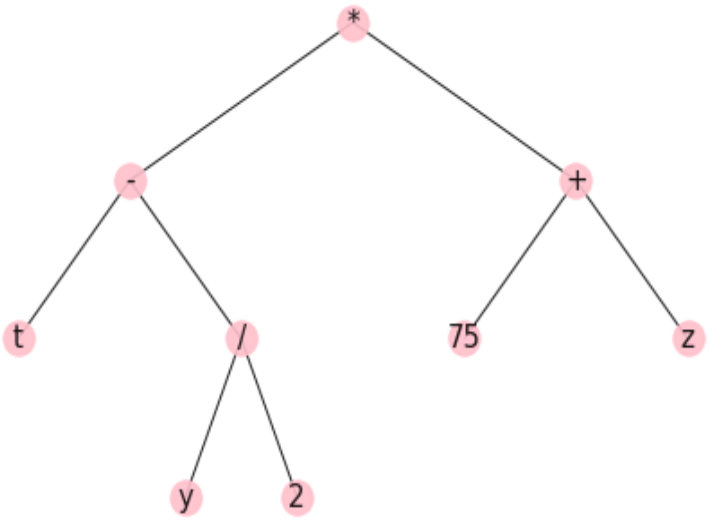


FIGURE 5 – expression arithmetique

1. Quelle est l'expression arithmetique représentée par cet arbre ? Celle-ci sera écrite en notation polonaise inversée. (valeur1 valeur2 operateur)
2. Quel doit être le parcours de l'arbre pour retourner cette expression en notation polonaise inversée.
3. Quel parcours de l'arbre va donner l'expression telle qu'on la manipule en cours de math ?

Exercice 5

Implementations d'un arbre

Les implémentations se feront pour des arbres dits binaires.

On traitera les questions pour l'arbre A suivant :

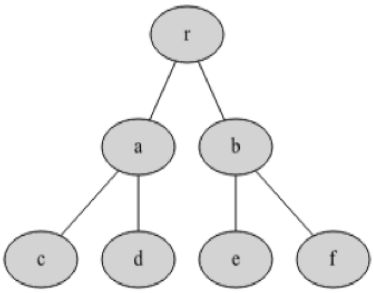


FIGURE 6 – arbre A

5.1 Listes imbriquées

1. Ecrire la liste qui représente cet arbre en utilisant la convention suivante :

```
arbre = ['r',['a',['c',None,None], ...]
```

Conseil : attention à bien écrire `['cle',None,None]` pour chacune des feuilles.

2. Ecrire la fonction de parcours *postfixe* pour parcourir des arbres représentés par une liste imbriquée.
3. Ecrire une fonction `est_egal` qui teste l'égalité de 2 arbres. Cette fonction renvoie `True` si les deux arbres ont la même structure, mêmes clés.

5.2 Classe

1. Ecrire le constructeur de la classe `ArbreBinaire` qui permettra de représenter un arbre binaire. Les fils gauche et fils droit de chaque noeud seront initialisés à `None`.
2. Ajouter les méthodes de classe `ajoute_fils_gauche` et `ajoute_fils_droit` qui vont permettre d'ajouter un fils gauche ou fils droit à un noeud, si la place est libre.
3. Ecrire les instructions d'instanciation pour cet arbre : instancier les noeuds appelés `racine` (clé 'r'), `noeud1` (clé 'a'), `noeud2` (clé 'b'),...
4. Ecrire une méthode de classe `parcours_prefixe` qui donne la liste des clés traitées dans l'ordre prefixe. On utilise cette méthode avec l'instruction `racine.parcours_prefixe()`
5. Ecrire une méthode de classe `taille` qui retourne le nombre de noeuds dans l'arbre.

Exemple :

```
>>> racine.taille()
7
```

6. Ecrire une méthode de classe appelée `miroir` qui transforme, en place, l'arbre binaire `A` en son miroir : l'arbre symétrique dans lequel les fils gauche et droite sont échangés.

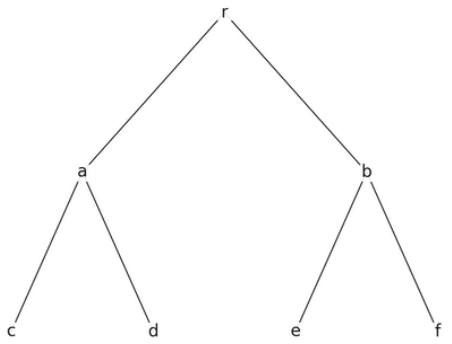
Aide : Cette fonction devra réaliser un parcours en profondeur, et executer une permutation des fils d'un noeud APRES avoir parcouru ses sous-arbre gauche et sous-arbre droit.

7. Préciser ce qui va être affiché avec `racine_miroir.infixe()`. Comparer avec `racine.infixe()`

```
from copy import deepcopy # copie par valeurs d'un objet
racine_miroir = deepcopy(racine)
racine_miroir.miroir()
racine_miroir.infixe()
```

8. Fonction `branches` : Cette fonction est utile pour tracer l'arbre (voir TP)

a. Utiliser le script de la fonction `branches` (à gauche) pour compléter celui `ArbreBinaire.branche` (à droite)




```
arbre = ('r',('a',('c',None,None),('d',None,None)),
         ('b',('e',None,None),('f',None,None)))

def branches(a, L=[]):
    """construit une liste de branches
    à partir d'une liste (ou tuple) imbriquée
    """
    if a is None:
        return []
    else:
        if not a[1] is None:
            L.append((a[0],a[1][0]))
            branches(a[1],L)
        if not a[2] is None:
            L.append((a[0],a[2][0]))
            branches(a[2],L)
    return L

-------------------------------------------------
>>> L = []
>>> branches(arbre)

[('r', 'a'), ('a', 'c'), ('a', 'd'),
 ('r', 'b'), ('b', 'e'), ('b', 'f')]
```

```
racine = ArbreBinaire('r')
noeud1 = ArbreBinaire('a')
noeud2 = ArbreBinaire('b')
...
racine.set_FG(noeud1)
racine.set_FD(noeud2)
noeud1.set_FG(noeud3)
...

class ArbreBinaire:
    """ methode de classe
    construit une liste de branches
    à partir d'un objet de la classe ArbreBinaire
    """
    def branches(self,L=[]):
        if self is None:
            return []
        else:
            if not self.FilsGauche is None:
                L.append((...
                self.FilsGauche.branches(L)
            if not self.FilsDroit is None:
                L.append((...
                self.FilsDroit.branches(L)
        return L

-------------------------------------------------
>>> L = []
>>> racine.branches()

[('r', 'a'), ('a', 'c'), ('a', 'd'),
 ('r', 'b'), ('b', 'e'), ('b', 'f')]
```

FIGURE 7 – Deux implémentations. Comparaison

b. En programmation objet : Quelle instruction va donner la liste des branches pour le sous-arbre dont la clé est 'a' ?

Exercice 6

Exercice : arbres binaires - bac 2019

Dans cet exercice on adoptera la convention suivante : la hauteur d'un arbre binaire ne comportant qu'un nœud est 1.

6.1 Déterminer la taille et la hauteur de l'arbre binaire suivant :

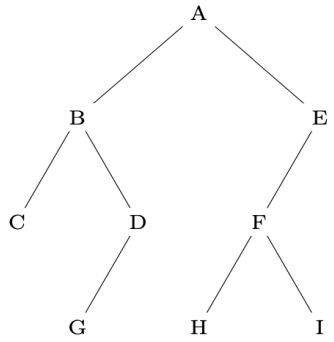


FIGURE 8 – arbre binaire alphabet

6.2 Arbre et numération binaire

On décide de numéroter en binaire les noeuds d'un arbre binaire de la façon suivante :

- la racine correspond à 1 ;
- la numérotation pour un fils gauche s'obtient en ajoutant le chiffre 0 à droite de son père ;
- la numérotation pour un fils droit s'obtient en ajoutant le chiffre 1 à droite de son père ;

Exemple :

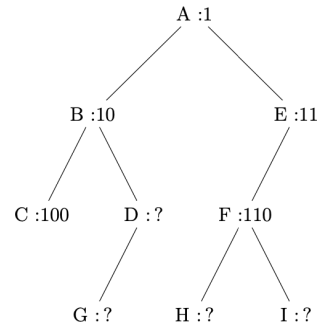


FIGURE 9 – fig 4 - arbre binaire numéroté

L'arbre présenté ici n'est pas complet.

6.2.1 Quel est le numéro en binaire associé aux noeuds H et G ?

6.2.2 Combien de noeuds différents peut-on dénombrer à l'aide de cette méthode sur le niveau le plus en bas ?

On donne l'agorithme général du parcours postfixe d'un arbre binaire :

```
ParcoursPostfixe ( Arbre binaire T de racine r )
  ParcoursPostfixe(Arbre de racine fils_gauche[r])
  ParcoursPostfixe(Arbre de racine fils_droit[r])
  Afficher clef [ r ]
```

6.2.3 Donner la succession des clés affichées lorsque l'on realise le parcours postfixe sur l' arbre binaire précédent. Vous n'écrirez que les lettres de chaque noeud.

6.3 Représentation d'un arbre

On décide de représenter un arbre binaire par un tableau (une liste) de taille n + 1, où n est la taille de l'arbre, de la façon suivante :

- La racine a pour indice 1 ;
- Le fils gauche du nœud d'indice i a pour indice 2 × i ;
- Le fils droit du nœud d'indice i a pour indice 2×i+1 ;
- On place la taille n de l'arbre dans la case d'indice 0.
- Lorsqu'il n'y a pas de valeur à renseigner pour un indice donné, mettre `None`

6.3.1 Déterminer le tableau qui représente l'arbre binaire de l'exemple précédent. (fig 4)

6.3.2 On considère le père du nœud d'indice i avec i ≥ 2. Quel est son indice dans le tableau ? Considérer les 2 cas :

- i pair
- i impair

Exercice 7

Corrections

Ex 1 : parcours recursif d'un systeme de fichiers

```
def fils_recursif(p,L=[]):
    if os.path.isfile(p):
        return ""

    for y in os.listdir(p):
        fils_recursif(p+'/'+y)
        L.append(p+'/'+y)

    return L
```