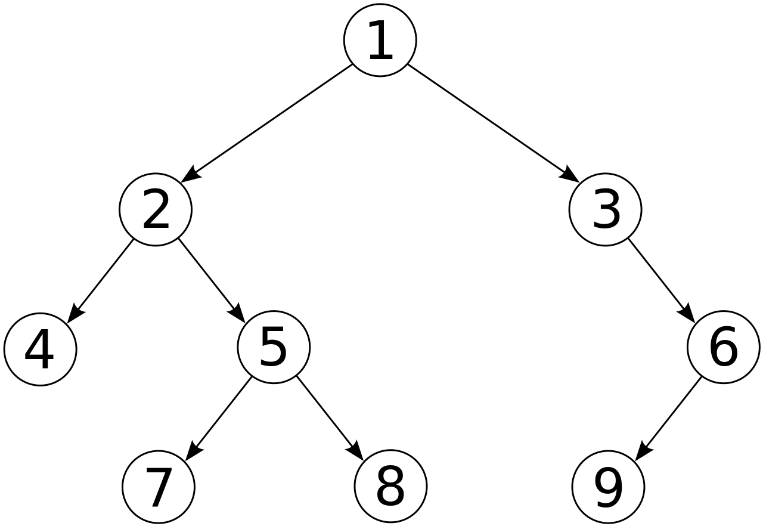
|  |  |
| --- | --- |
| Term NSI | AD1 Arbres définitions et représentations |
| AD : AD1 Arbres définitions et représentations [Extrait de pixees](https://pixees.fr/informatiquelycee/n_site/nsi_term.html), Ouvrage Ellipses et Nathan, [vidéo formation ICN](https://www.fun-mooc.fr/courses/inria/41014/session01/about) | |

***Ex 1*** : Définitions

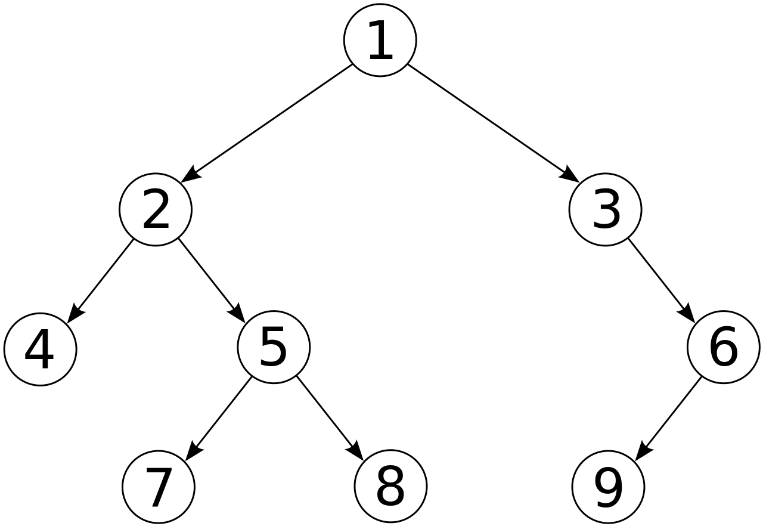
|  |  |
| --- | --- |
| Quel est le nom du **T**ype **A**bstrait de  **D**onnée ci-contre ? arbre binaire  Quel est le nœud racine ? 1  Quel est le père du nœud 5 ? 2  Quel est le fil gauche SAG de 2 ? 4  Quel est le fil droit SAD de 2 ? 5  Quelle est la hauteur de 1 ? 1  Quelle est la profondeur de 5 ? 3 | **Fig1**  Quel est le chemin de 7 ? 7 – 5 – 2 - 1  Quel est le ***chemin le plus long ?*** 4 |

**Ex2 :** **TAD** compléter la représentation de l’arbre *fig1* ci-dessus



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | Noeud1 | 1 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | sag | 2 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | Noeud2 | 2 |  |  | sad | 3 |  |  | Noeud3 | 3 |  |  |  |  |  |  |  |
|  |  |  | sag | 4 |  |  |  |  |  |  | sag | none |  |  |  |  |  |  |  |
| Noeud4 | 4 |  | sad | 5 |  | Noeud5 | 5 |  |  |  | sad | 6 |  |  |  | Noeud6 | 6 |  |  |
| sag | none |  |  |  |  | sag | 7 |  |  |  |  |  |  |  |  | sag | 9 |  |  |
| sad | none |  |  |  |  | sad | 8 |  |  |  |  |  |  |  |  | sad | none |  |  |
|  |  |  | Noeud7 | 7 |  |  |  | Noeud8 | 8 |  |  | Noeud9 | 9 |  |  |  |  |  |  |
|  |  |  | sag | none |  |  |  | sag | none |  |  | sag | none |  |  |  |  |  |  |
|  |  |  | sad | none |  |  |  | sad | none |  |  | sad | none |  |  |  |  |  |  |

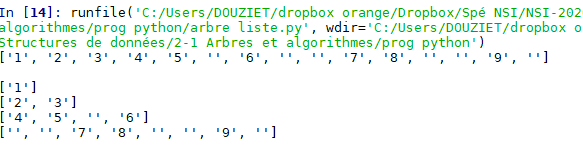
**Ex3 : TAD** représentation par une liste l’arbre fig1 ci-dessus



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** |  | **6** |  |  | **7** | **8** |  |  | **9** |  |
| **Niv0** | **Niv1** | | **Niv2** | | | | **Niv3** | | | | | | | |

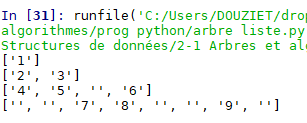
Créer l’objet liste « tableau » dans python , algorithme :

|  |  |
| --- | --- |
| Créer la liste ci-dessus  Afficher la liste complète Afficher le **niveau 0**  Afficher le **niveau 1** Afficher le **niveau 2** Afficher le **niveau 3** | **arbre = ["1","2","3","4","5","","6","","","7","8","","","9",""]**  **print(arbre)**  **print(arbre[0:1])**  **print(arbre[1:3])**  **print(arbre[3:7])**  **print(arbre[7:15])** |

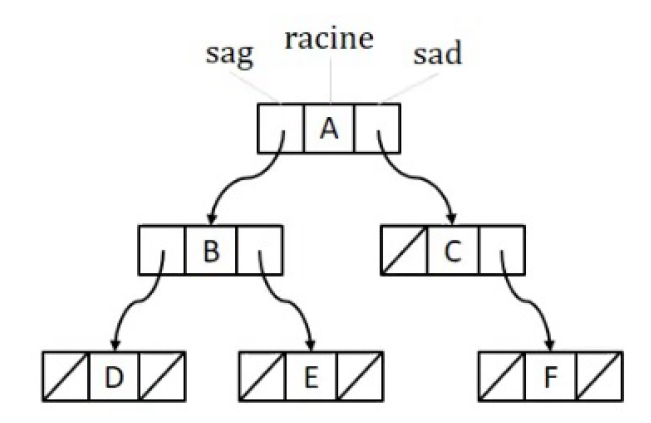


Algorithme plus « élégant »

|  |  |
| --- | --- |
| Créer la liste ci-dessus  Pour niv =0 to 3 inclus   afficher la liste de   ( | **arbre = ["1",  "2","3",  "4","5","","6",   "","","7","8","","","9",""]**  **for i in range (4):**  **print(arbre[2\*\*i-1:2\*\*i-1+2\*\*i])** |



**Ex4 : TAD** représentation « ***objet » , nous allons progressivement créer*** l’arbre ci-dessous (python)



Voir <https://courspython.com/classes-et-objets.html>

|  |  |
| --- | --- |
| 1. Créer dans python la class Nœud   Initialisation :   donnée = A , gauche =, droit =  Créer un objet N-A (nœud A) nommé A  Afficher « nœud »  Afficher N\_A.donnée  Afficher "="  Afficher N\_A | class Noeud :  # Constructeur  def \_\_init\_\_(self, valeur) :  self.donnée = valeur  self.gauche = None  self.droit = None  N\_A=Noeud('A')  print("noeud")  print(N\_A.donnée)  print("=")  print(N\_A) |

|  |  |
| --- | --- |
|  |  |

Nous avons créé un objet :

|  |  |  |
| --- | --- | --- |
| Nom de la class | 🡪 | N\_A |
| Attributs | 🡪 | donnée = 'A'  gauche = None  droit = None |
| Méthode | 🡪 | ?????? |

Pour connaître la valeur d’un attribut :  
dans le programme ***N\_A.donnée***, dans la class ***self.donnée***

|  |  |
| --- | --- |
| 1. ***Amélioration de l’affichage print***   Ajouter les lignes suivantes dans la class  Puis tester votre programme | class Noeud :  [...]  # Affichage  def \_\_str\_\_(self) :  return str(self.donnée)  [...] |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| 1. ***Amélioration de l’affichage***   ***print (plus complet)***  Modifier les lignes  def \_\_str\_\_(self) : dans la class  Puis tester votre programme | **# Affichage**  **def \_\_str\_\_(self) :**  **return "({},{},{})".format(self.gauche, self.donnée, self.droit)** |

|  |  |
| --- | --- |
|  | **SAG SAD** |

|  |  |
| --- | --- |
| 1. ***Création de la méthode SAG***   Ajouter les lignes **def SAG** dans la class  Créer un objet N-B (nœud B) nommé B  Affecter le N\_B au SAG de N\_A  Puis tester votre programme | class Noeud :  [...]  **# Création du sous-arbre gauche**  **def SAG(self,valeur):**  **self.gauche = valeur**  **N\_B=Noeud('B') N\_A.SAG(N\_B) print("noeud",N\_A.donnée,"=",N\_A)** |

L’objet N\_A devient :

|  |  |  |
| --- | --- | --- |
| Nom de la class | 🡪 | N\_A |
| Attributs | 🡪 | donnée = 'A'  gauche = N\_B  droit = None |
| Méthode | 🡪 | .SAG |

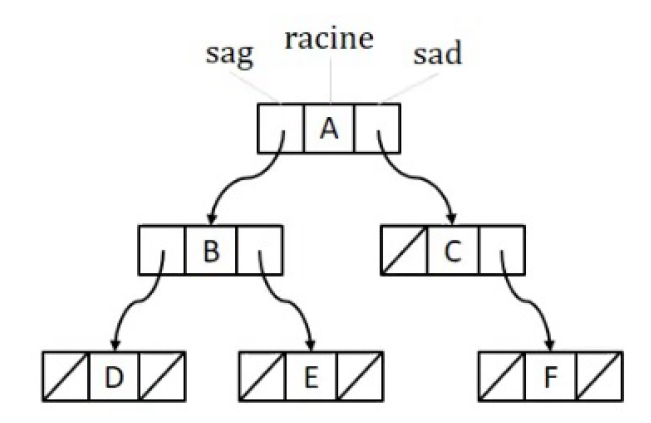
|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| ***e)Création de la méthode SAD***  Créer les lignes **def SAD** dans la class  Créer un objet N-C (nœud C) nommé C  Affecter le N\_C au SAD de N\_A  Puis tester votre programme | class Noeud :  [...]  **# Création du sous-arbre droit**  **def SAD(self,valeur):**  **self.droit = valeur** |

L’objet N\_A devient :

|  |  |  |
| --- | --- | --- |
| Nom de la class | 🡪 | N\_A |
| Attributs | 🡪 | donnée = 'A'  gauche = N\_B  droit = N\_C |
| Méthode | 🡪 | .SAG .SAD |

|  |  |
| --- | --- |
|  |  |



|  |  |
| --- | --- |
| ***F) Création de la méthode assembler (plus utiliser dans les arbres que SAG et SAD)***  Créer les lignes **def assembler** dans la class  Créer les nœuds D E et F  Assembler nœud B avec D et E  Assembler nœud C avec rien et F  Puis tester votre programme | class Noeud :  [...]  # Assembler  def assembler(self,ag,ad) :  self.gauche = ag  self.droit = ad  N\_D=Noeud('D')  N\_E=Noeud('E')  N\_F=Noeud('F')  N\_B.assembler(N\_D,N\_E)  print("noeud",N\_B.donnée,"=",N\_B)  N\_C.assembler(None,N\_F)  print("noeud",N\_C.donnée,"=",N\_C) |

L’objet N\_A devient :

|  |  |  |
| --- | --- | --- |
| Nom de la class | 🡪 | N\_A |
| Attributs | 🡪 | donnée = 'A'  gauche = N\_B  droit = N\_C |
| Méthode | 🡪 | .SAG .SAD  .assembler |

|  |  |
| --- | --- |
| ***G) amélioration de l’affichage***  Importer la bibliothèque matplot  dans la class ajouter les méthodes .dump et .\_dump  Et tester | import matplotlib.pyplot as plt  class Noeud :  [...]  # affichage de l'arbre  def dump(self):  self.\_dump(0)  def \_dump(self, level):  if self.droit:  self.droit.\_dump(level + 1)  print("{}{}{}".format(' ' \* 4 \* level, self.donnée," < "))  if self.gauche:  self.gauche.\_dump(level + 1)  N\_A.dump() |

Modifier votre programme sans utiliser les méthodes .SAG .SAD mais uniquement la méthode assembler algorithme :

|  |  |
| --- | --- |
| Créer les nœuds A,B,C,D,E et F    assembler suivant l’arbre  Afficher l’arbre | N\_A=Noeud('A')  N\_B=Noeud('B')  N\_C=Noeud('C')  N\_D=Noeud('D')  N\_E=Noeud('E')  N\_F=Noeud('F')  N\_A.assembler(N\_B,N\_C)  N\_B.assembler(N\_D,N\_E)  N\_C.assembler(None,N\_F)  N\_A.dump() |

|  |  |
| --- | --- |
|  |  |

L’objet N\_A devient :

|  |  |  |
| --- | --- | --- |
| Nom de la class | 🡪 | N\_A |
| Attributs | 🡪 | donnée = 'A'  gauche = N\_B  droit = N\_C |
| Méthode | 🡪 | .assembler |