BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°03

DUREE DE L'EPREUVE : 1 heure

Le sujet comporte 2 pages numérotées de 1 / 3 à 3 / 3
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (4 points)

Le codage par différence (*delta encoding* en anglais) permet de compresser un tableau de données en indiquant pour chaque donnée, sa différence avec la précédente (plutôt que la donnée elle-même). On se retrouve alors avec un tableau de données assez petites nécessitant moins de place en mémoire. Cette méthode se révèle efficace lorsque les valeurs consécutives sont proches.

Programmer la fonction `delta` qui prend en paramètre un tableau non vide de nombres entiers et qui renvoie un tableau contenant les valeurs entières compressées à l'aide cette technique.

Exemples :

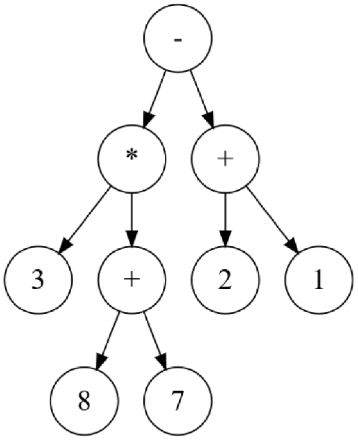
```
>>> delta([1000, 800, 802, 1000, 1003])
[1000, -200, 2, 198, 3]
>>> delta([42])
42
```

EXERCICE 2 (4 points)

Une expression arithmétique ne comportant que les quatre opérations $+, -, \times, \div$ peut être représentée sous forme d'arbre binaire. Les nœuds internes sont des opérateurs et les feuilles sont des nombres. Dans un tel arbre, la disposition des nœuds joue le rôle des parenthèses que nous connaissons bien.

En parcourant en profondeur infixe l'arbre binaire ci-contre, on retrouve l'expression notée habituellement :

$3 \times (8 + 7) - (2 + 1)$.



La classe `Noeud` ci-après permet d'implémenter une structure d'arbre binaire.

Compléter la fonction récursive `expression_infixe` qui prend en paramètre un objet de la classe `Noeud` et qui renvoie l'expression arithmétique représentée par l'arbre binaire passé en paramètre, sous forme d'une chaîne de caractères contenant des parenthèses.

Résultat attendu avec l'arbre ci-dessus :

```
>>> e = Noeud(Noeud(Noeud(None, 3, None), '*', Noeud(Noeud(None, 8, None),
'+', Noeud(None, 7, None))), '-', Noeud(Noeud(None, 2, None), '+',
Noeud(None, 1, None)))

>>> expression_infixe(e)
'((3*(8+7))-(2+1))'
```

```
class Noeud:
```

```
    '''
    Classe implémentant un noeud d'arbre binaire disposant de 3
attributs :
    - valeur : la valeur de l'étiquette,
    - gauche : le sous-arbre gauche.
    - droit : le sous-arbre droit.
    '''
    def __init__(self, g, v, d):
        self.gauche = g
        self.valeur = v
        self.droit = d

    def est_une_feuille(self):
        '''Renvoie True si et seulement si le noeud est une feuille'''
        return self.gauche is None and self.droit is None


def expression_infixe(e):
    s = ...
    if e.gauche is not None:
        s = s + expression_infixe(...)
    s = s + ...
    if ... is not None:
        s = s + ...
    if ...:
        return s

    return '('+ s +')'
```

```python
from doctest import testmod


def delta(tab: list) -> list:
    """ Codage par différence du tableau tab.

    Args:
        tab (list): tableau non vide de nombres entiers

    Returns:
        list: tableau de valeurs entières compressées par différence

    Tests et Exemples:!
    >>> delta([1000, 800, 802, 1000, 1003])
    [1000, -200, 2, 198, 3]
    >>> delta([42])
    [42]
    """
    diff = [tab[0]]

    n = len(tab)
    for i in range(1, n):
        delta = tab[i] - tab[i - 1]
        diff.append(delta)

    return diff



testmod()


# upylab

from random import randint
tab = [randint(0, 1000),randint(0, 1000),randint(0, 1000),randint(0, 1000),randint(0
, 1000),randint(0, 1000),randint(0, 1000),randint(0, 1000),randint(0, 1000),randint(
0, 1000)][:randint(1,10)]

print(tab)
```

```python
from doctest import testmod


class Noeud:
    def __init__(self, g, v, d):
        self.gauche = g
        self.valeur = v
        self.droit = d

    def __str__(self):
        return str(self.valeur)

    def est_une_feuille(self):
        '''Renvoie True si et seulement si le noeud est une feuille'''
        return self.gauche is None and self.droit is None


def expression_infixe(e):
    """Parcours infixe d'un arbre pour afficher les noeuds
    séparés par des parenthèses.

    Args:
        e (Noeud): racine de l'arbre

    Returns:
        str: expression bien formée par des parenthèses de l'abre e

    Tests et Exemples:
    >>> e = Noeud(Noeud(Noeud(None, 3, None), '*', Noeud(Noeud(None, 8, None), '+',
Noeud(None, 7, None))), '-', Noeud(Noeud(None, 2, None), '+', Noeud(None, 1, None)))
    >>> expression_infixe(e)
    '((3*(8+7))-(2+1))'
    """
    # initialisation de la chaine à renvoyer
    s = ""
    if e.gauche is not None:
        # appel récursif pour afficher d'abord le sous arbre gauche
        s = s + expression_infixe(e.gauche)
    # ajout de la valeur du noeud
    # en utilisant la méthode dédiée
    s = s + str(e)
    if e.droit is not None:
        # appel récursif pour afficher d'abord le sous arbre gauche
        # s'il est non vide
        s = s + expression_infixe(e.droit)

    # cas de base : le noeud est une feuille
    if e.est_une_feuille():
        # afficher la valeur sans les parenthèses
        return s


    return '('+ s +')'


if __name__ == '__main__':
    testmod()

    e = Noeud(Noeud(Noeud(None, 3, None), '*', Noeud(Noeud(None, 8, None), '+', Noeu
d(None, 7, None))), '-', Noeud(Noeud(None, 2, None), '+', Noeud(None, 1, None)))
    assert expression_infixe(e) == '((3*(8+7))-(2+1))'
```