evalpilefile

December 1, 2024

1 Evaluation Pile File

1.1 Exercice 1

Dans un programme en Python, on peut utiliser différents signes pour définir les structures de données:

- les crochets pour les tableaux,
- les parenthèses pour les tuples,
- les accolades pour les dictionnaires.

Ces structures peuvent être imbriquées et il convient que l'environnement de développement (Thonny, Spyder, Pyzo…) vérifie que les chaînes soient bien formées. Une pile est généralement utilisée pour réaliser cette vérification. L'algorithme est le suivant:

- Parcourir la chaîne à vérifier caractère par caractère. - Si le caractère est un signe ouvrant **(** **[** **{** on l'empile.
  - Si le caractère est un signe fermant **)** **]** **}** on dépile:
  - si la pile est vide, la chaîne est mal formée,
  - si le caractère dépilé ne correspond pas à celui en cours, la chaîne est mal formée.
- À la fin du parcours, si la pile est vide, la chaîne est bien formée.

1. Écrire une classe **Cellule** qui possédera deux attributs:

- **caractere** de type **string**
- **suivant** de type **object** (en pratique une **Cellule**)

Ces deux attributs seront initialisés par deux paramètres passés au constructeur.

2. Écrire une classe Pile qui possédera:
   - un attribut sommet initialisé à None
   - trois méthodes:
     - **est_vide** qui renvoie **True** si la pile est vide, **False** sinon,
     - **empiler** qui ajoute la **Cellule** passée en paramètre, au sommet de la pile,
     - **depiler** qui dépile la Cellule au sommet et renvoie le caractère stocké ou la chaîne **"vide"** si la pile est vide.

3. Écrire la fonction bien_formee(chaine: str) -> bool qui renvoie True si la chaîne est bien formée, False sinon. La fonction respectera l'algorithme présenté dans l'énoncé et utilisera obligatoirement une pile et le dictionnaire suivant.

[1]:
```
associe = {")":"(", "]":"[", "}":"{"}
```

4. Effectuer 3 assertions qui vérifie des cas de figure différents.

1.2 Exercice 2

Écrire une file qui contiendra des entiers et dont l'interface est composée de 4 fonctions: - creer_file() -> list
- est_vide(file: list) -> bool
- defiler(file: list) -> int
- enfiler(file: list, val: int) -> None
Si la file est vide, la fonction defiler renverra -1.

2. Écrire la fonction inverser_file qui prend une file en paramètre et inverse l'ordre des éléments.

Indice

Il pourrait être intéressant d'utiliser une pile. ***

Attention

Vos fonctions doivent avoir une docstring correctement formée.
Vous devez commenter votre code et être rigoureux.