TP sur les Piles et les Files(Correction)

A.MAZZA
CPGE Oujda
mazza8azzouz@gmail.com

2 février 2025

Table des matières

1 Problème 1 : Vérification des Expressions Parenthésées

1.1 Analyse du Problème

Dans une expression mathématique contenant des parenthèses `( )`, des crochets `[ ]` et des accolades `{ }`, on veut vérifier que chaque symbole ouvrant est bien associé à un symbole fermant correspondant.

Exemples :

— `74*(1-9/(7+1.2))` → Correct
— `74*(1-9/7+1.2))` → Incorrect
— `74*(1-9/(7+1.2)` → Incorrect

1.2 Méthode et Implémentation

On utilise une pile :

1. Lire l'expression caractère par caractère.
2. Empiler les symboles ouvrants `(, [, {`.
3. Lorsqu'un symbole fermant `), ], }` est rencontré :
   — Vérifier si la pile est vide (si oui, erreur).
   — Dépiler et vérifier si le dernier élément correspond bien au symbole fermant.
4. À la fin, la pile doit être vide.

```
def verifier_parentheses(expression):
    pile = []

    for char in expression:
        if char in "({[":
            pile.append(char)
        elif char in ")}]":
            if not pile:
                return False
            dernier = pile.pop()
            if (char == ')' and dernier != '(') or \
               (char == ']' and dernier != '[') or \
               (char == '}' and dernier != '{'):
                return False

    return len(pile) == 0
```

Listing 1 – Vérification des parenthèses en Python

2 Problème 2 : Inversion d'une Phrase avec une Pile

2.1 Analyse du Problème

On souhaite écrire une fonction qui inverse l'ordre des mots d'une phrase en utilisant une pile.

Exemple : Entrée : `"Je suis étudiant"` Sortie : `"étudiant suis Je"`

2.2 Méthode et Implémentation

1. Découper la phrase en mots.
2. Empiler chaque mot.
3. Dépiler les mots pour reconstruire la phrase inversée.

```python
def inverser_phrase(phrase):
    pile = phrase.split()
    phrase_inverse = []

    while pile:
        phrase_inverse.append(pile.pop())

    return '␣'.join(phrase_inverse)

print(inverser_phrase("Je␣suis␣ tudiant "))
# R sultat attendu : " tudiant  suis Je"
```

Listing 2 – Inversion d'une phrase avec une pile

3 Problème 3 : Simulation d'une File d'Attente

3.1 Analyse du Problème

On veut simuler une file d'attente où les clients arrivent aléatoirement et sont servis selon leur ordre d'arrivée.

3.2 Méthode et Implémentation

1. Modéliser une file avec une liste.

2. À chaque itération :
   — Ajouter un client avec une probabilité de 50%.
   — Servir un client avec une probabilité de 50%.
3. Afficher l'état de la file après chaque opération.

```
import random
import time

class File:
    def __init__(self):
        self.elements = []

    def ajouter(self, element):
        self.elements.append(element)

    def retirer(self):
        if not self.est_vide():
            return self.elements.pop(0)
        return None

    def est_vide(self):
        return len(self.elements) == 0

    def afficher(self):
        print(" tat de la file :", self.elements)

file_attente = File()

for i in range(10):
    if random.random() < 0.5:
        client = f"Client {i}"
        file_attente.ajouter(client)
        print(f"{client} arrive.")

    if random.random() < 0.5 and not file_attente.
       est_vide():
        print(f"{file_attente.retirer()} est servi.")

    file_attente.afficher()
    time.sleep(1)
```

Listing 3 – Simulation d'une file d'attente