Description du projet et de sa fonctionnalité

Le projet d'une calculatrice simple, qui permet à l'utilisateur de réaliser des calculs simples tout en gérant l'historique des calculs. Le logiciel tient compte des priorités des opérations. L'utilisateur peut également supprimer les entrées ou effacer l'entrée et l'historique entière.

L'historique des opérations est sauvegardée dans un fichier json (history.json par défaut). L'utilisateur peut sauvegarder l'historique dans un fichier sous un autre nom pour créer un log personnalisé. Il peut également supprimer ou consulter les logs directement dans le programme.

La structure du programme

1. Calcul des expressions
2. Gestion des entrées utilisateur
3. Gestion de l'historique

1. Calcul des expressions

   La fonction principale: def caclculate_expression(expression): cette fonction évalue une expression mathématique sous forme de chaîne de caractères. Elle comprend deux fonctions supplémentaires def apply_operator(operators, values) et def precedence(op).

   La fonction apply_operator(operators, values) prend le dernier opérateur de la liste des opérateurs et l'applique aux deux dernières valeurs de la liste des valeurs, en effectuant une opération arithmétique en fonction de signe d'opérateur.

   La fonction precedence(op) établit la priorité des opérateurs (*, /) sur les opérateurs (+, —)

   L' algorithme principale:

On initialise les liste vides ‘operators’ pour stocker les opérateurs et ‘values’ pour stocker les valeurs. On parcourt notre expression mathématique. S’il on croise ‘(‘, on le stocke immédiatement dans la liste des opérateurs. Si on croise un nombre, on le met dans la liste des valeurs. Si on croise un opérateur mathématique, avant de l’ajouter à la liste des opérateurs, on compare sa priorité avec la priorité du dernier opérateur de la liste. Si sa priorité est supérieure à celle du dernier opérateur de la liste, l’opérateur courant est ajouté à la liste. Si sa priorité est égale ou plus basse que celle du dernier opérateur, c’est ce dernier qui est exécuté (pour l’exécuter on appelle la fonction apply_operator()), et l’opérateur courant est ajouté à la liste des opérateurs seulement après cette comparaison. Si l’opérateur courant est ‘)’, on commence à exécuter tous les opérateurs précédents dans l’ordre dans lequel ils sont stockés dans la liste, tant qu’on n’arrive pas à l'opérateur ‘(‘. Ainsi, on assure que les opérateurs entre () en priorités. A la fin de l’algorithme, on exécute les opérateurs restants.

Voici un exemple comme fonctionne notre algorithme:

1. Initialisation
operators = []
values = []

2. Lecture de l'expression caractère par caractère
Caractère 3 :

C'est un nombre, il est placé dans values.
values = [3]
Caractère + :

C'est un opérateur. Comme la liste operators est vide, il est ajouté.
operators = [+]
Caractère 5 :

C'est un nombre, il est placé dans values.
values = [3, 5]
Caractère * :

C'est un opérateur. Sa priorité (* a une priorité de 2) est plus élevée que celle de + (priorité de 1). Il est ajouté à la liste operators.
operators = [+, *]

Caractère 2 :

C'est un nombre, il est placé dans values.
values = [3, 5, 2]
3. Fin de l'expression
À ce stade, tous les caractères ont été lus. On passe à l'évaluation.

Étape 3 : Évaluation des listes
Évaluation de * :

L'opérateur * est en haut de la liste operators. On applique l'opération sur les deux derniers nombres de la liste values :
$5 \times 2 = 10$

values = [3, 10]
operators = [+]
Évaluation de + :

L'opérateur + est à la fin de la liste operators. On applique l'opération sur les deux derniers nombres de la liste values :
$3 + 10 = 13$

values = [13]
operators = []
Résultat: 13

Gestion des Entrées Utilisateur

Fonction Principale de l'Interface Utilisateur

calculator() :
Cette fonction sert d'interface principale de la calculatrice. Elle offre à l'utilisateur un menu interactif lui permettant d'entrer des valeurs, d'effectuer des opérations, d'afficher l'historique, de supprimer des entrées ou d'enregistrer et d'ouvrir des fichiers d'historique.
Options Disponibles pour l'Utilisateur

Entrée de calcul :

L'utilisateur peut entrer des nombres ou des opérateurs (+, -, *, /, (,)). Les calculs peuvent être finalisés en entrant '=', ce qui calcule le résultat et l'ajoute à l'historique.

Effacer une entrée :

L'utilisateur peut supprimer le dernier caractère de l'entrée en cours en utilisant la touche 'd'.

Réinitialiser l'entrée et l'historique :

En utilisant la touche 'c', l'utilisateur peut effacer l'entrée en cours ainsi que l'historique des calculs.

Afficher l'historique :

En appuyant sur la touche 'h', l'utilisateur peut consulter l'historique des calculs effectués.

Ouvrir un fichier d'historique :

L'utilisateur peut ouvrir un fichier d'historique existant avec la touche 'o' en spécifiant le nom du fichier.

Enregistrer l'historique dans un fichier sous un nom choisi par l'utilisateur :

L'utilisateur peut sauvegarder l'historique actuel dans un fichier avec la touche 's' en spécifiant un nom de fichier.

Supprimer un fichier d'historique :

En appuyant sur 'r', l'utilisateur peut supprimer un fichier d'historique existant en spécifiant le nom du fichier.

Gestion de l'Historique

Variables Globales

history_file: Nom du fichier où l'historique des calculs est sauvegardé (par défaut "history.json").
history: Liste contenant les calculs effectués sous forme de chaînes de caractères.
current_entry: Représente l'entrée actuelle de l'utilisateur, qui est en cours de saisie.

Fonctions Associées

load_history_from_file() :

Charge l'historique à partir du fichier JSON. Si le fichier n'existe pas, l'historique est initialisé comme une liste vide. Si le fichier est corrompu, il réinitialise l'historique.

save_history_to_file() :

Sauvegarde l'historique actuel dans le fichier history.json.

get_history() :

Récupère l'historique complet des calculs.

delete_last_history_entry() :

Supprime la dernière entrée de l'historique et la sauvegarde.

reset_history() :

Réinitialise l'historique et l'entrée actuelle, puis sauvegarde l'historique vide.