Lab 2 : Calculatrice en Python en architecture MVC

Objectif du Lab 2

- **Concevoir** une application calculatrice en mode console en suivant une architecture MVC simplifiée.
- **Séparer** le code en modules distincts pour la **logique métier (modèle)**, l'**affichage (vue)** et le **contrôle (contrôleur)**.
- **Respecter** les bonnes pratiques de codage en Python (PEP8, KISS, DRY).

Structure du Projet

Organisez votre projet avec la structure suivante :

```
calculatrice_mvc/
├── model.py          # Contiendra les fonctions de calcul (modèle)
├── view.py           # Contiendra les fonctions d'affichage et de saisie (vue)
├── controller.py     # Contiendra la logique de coordination entre la vue et le modèle (contrôleur)
└── main.py           # Point d'entrée de l'application
```

Explication de l’Architecture MVC (sans OOP)

- **Modèle (model.py)** : Ce module contient les fonctions qui effectuent les calculs de la calculatrice (addition, soustraction, multiplication, division).
- **Vue (view.py)** : Ce module gère l'affichage des menus et la saisie des données par l'utilisateur.
- **Contrôleur (controller.py)** : Ce module coordonne les entrées de la vue, appelle les fonctions du modèle et gère la logique de l’application.
- **main.py** : Le script principal qui initialise l'application en appelant le contrôleur.

Détails des Bonnes Pratiques

- **PEP8** : Suivez les conventions PEP8 (indentation de 4 espaces, noms de variables en snake_case, etc.).
- **KISS** (Keep It Simple, Stupid) : Gardez les fonctions et le code simples et directs.
- **DRY** (Don't Repeat Yourself) : Évitez la duplication de code en factorisant les parties communes dans des fonctions ou modules.

Solution Proposée

1. Module model.py

Ce module contient les fonctions de calcul.

```python
# model.py
"""
Module Model : contient les fonctions de calcul de la calculatrice.
"""

def addition(a, b):
    """Renvoie la somme de a et b."""
    return a + b

def soustraction(a, b):
    """Renvoie la différence de a et b."""
    return a - b
```

```python
def multiplication(a, b):
    """Renvoie le produit de a et b."""
    return a * b

def division(a, b):
    """Renvoie la division de a par b. Gère la division par zéro."""
    if b == 0:
        return "Erreur : Division par zéro"
    return a / b
```

2. Module view.py

Ce module gère l'interface console (affichage et saisie).

```
# view.py
"""
Module View : gère l'affichage des menus et la saisie des données utilisateur.
"""

def afficher_menu():
    """Affiche le menu principal de la calculatrice et retourne le choix de l'utilisateur."""
    print("\n=== Calculatrice MVC ===")
    print("1. Addition")
    print("2. Soustraction")
    print("3. Multiplication")
    print("4. Division")
    print("5. Quitter")
    choix = input("Choisissez une opération (1-5): ")
    return choix
```

```python
def saisir_valeurs():
    """Demande à l'utilisateur de saisir deux nombres et retourne ces valeurs."""
    try:
        a = float(input("Entrez le premier nombre : "))
        b = float(input("Entrez le deuxième nombre : "))
        return a, b
    except ValueError:
        print("Entrée invalide. Veuillez saisir des nombres.")
        return None, None

def afficher_resultat(resultat):
    """Affiche le résultat de l'opération."""
    print(f"Le résultat est : {resultat}")

def message_erreur(message):
    """Affiche un message d'erreur."""
    print(f"Erreur : {message}")
```

3. Module controller.py

Ce module coordonne la vue et le modèle.

```
# controller.py
"""
Module Controller : coordonne la logique de la calculatrice.
"""

from model import addition, soustraction, multiplication, division
from view import afficher_menu, saisir_valeurs, afficher_resultat, message_erreur
```

```python
def lancer_calculatrice():
    """Fonction principale qui lance la calculatrice."""
    while True:
        choix = afficher_menu()
        if choix == '5':
            print("Au revoir !")
            break

        a, b = saisir_valeurs()
        if a is None or b is None:
            continue  # Si la saisie est invalide, redemander les valeurs

        if choix == '1':
            resultat = addition(a, b)
        elif choix == '2':
            resultat = soustraction(a, b)
        elif choix == '3':
            resultat = multiplication(a, b)
        elif choix == '4':
            resultat = division(a, b)
        else:
            message_erreur("Choix invalide")
            continue

        afficher_resultat(resultat)
```

4. Module main.py

Le script d'entrée qui lance l'application.

```
# main.py
"""
Point d'entrée de l'application calculatrice MVC.
"""

from controller import lancer_calculatrice

if __name__ == "__main__":
    lancer_calculatrice()
```

Instructions pour l'Exécution

1. Créez la structure du projet comme indiqué ci-dessus.
2. Créez un environnement virtuel et installez les dépendances si nécessaire (bien que ce projet utilise uniquement des modules Python intégrés).
3. Exécutez le script principal :
   - Sur Windows : `python main.py`
   - Sur Mac/Linux : `python3 main.py`
4. Suivez les instructions affichées dans la console pour utiliser la calculatrice.

Références et Ressources

- PEP 8 – Style Guide for Python Code
- Principes KISS et DRY dans la programmation
- Documentation officielle de Python : Python.org
- Introduction to MVC architecture in Python

Ce projet vous permet de comprendre comment structurer un projet Python en utilisant le modèle MVC sans recourir à la POO, tout en appliquant les bonnes pratiques de développement. Bon travail et bonne révision !

« Vous avez une force intérieure et un potentiel illimité. Croyez en vous et atteignez la grandeur. »

« Les petits pas mènent aux grands résultats. Continuez à avancer et faites confiance à votre dévouement. »

« Les défis sont des opportunités de croissance. Acceptez le voyage; votre détermination vous guidera. »

Bon courage à tous !