Raffinages

Objectifs

— Comprendre les raffinages
— Savoir produire des raffinages
— Mettre ne pratique la méthode des raffinage
— Écrire des programmes plus ambitieux

Exercice 1 : Des raffinages vers le programme

Produire un programme à partir de ses raffinages donnés ci-après. On écrira le programme dans le fichier `gerer_compteur.py`.

```
R0 : Gérer un compteur avec un menu

R1 : Comment « Gérer un compteur avec un menu »
    compteur = 0
    choix = '1'
    while choix != '0':
        Afficher le compteur                (compteur: in)
        Afficher le menu
        Demander le choix de l'utilisateur  (choix: out)
        Traiter le choix de l'utilisateur   (compteur: in out ; choix: in)

R2 : Comment « Afficher le menu »
    print("1. Incrémenter")
    print("2. Décrémenter")
    print("0. Quitter")

R2 : Comment « Afficher le compteur »
    print()
    print("Compteur :", compteur)
    print()

R2 : Comment « Demander le choix de l'utilisateur »
    choix = input("Votre choix : ")

R2 : Comment « Traiter le choix de l'utilisateur »
    if choix == '1':
        compteur += 1
    elif choix == '2':
        compteur -= 1
    elif choix != '0':
        print("Je n'ai pas compris.")
```

Exercice 2 : Retrouver un raffinage

L'objectif de cet exercice est de retrouver les raffinages qui sont à l'origine du programme ci-après. On écrira les raffinages dans le fichier `reviser_tables_multiplications_raffinages.txt`.

```
import random

def reviser_table():
    '''
    Réviser une table de multiplication.

    Principe : On pose plusieurs multiplications à l'utilisateur sur une table puis
    on lui affiche un bilan.
    '''

    # Demander la table à réviser
    table = int(input('Table à réviser : '))

    # Poser 10 multiplications
    nb_erreurs = 0
    for _ in range(10):
        # Poser une multiplication
        #   Choisir un nombre entre 0 et 10
        nombre = random.randint(0, 10)

        #   Construire l'énoncé de la multiplication
        enonce = '{} * {} = '.format(table, nombre)

        #   Demander la réponse à l'utilisateur
        reponse = int(input(enonce))

        #   Evaluer la réponse
        if reponse == table * nombre:    # Réponse correcte ?
            print('Correct')
        else:
            print('Erreur')
            nb_erreurs += 1

    # Afficher le bilan
    #   Calculer le nombre de bonnes réponses
    nb_bonnes_reponses = 10 - nb_erreurs
    #   Afficher le nombre de bonnes réponses
    print('Nombre de bonnes réponses :', nb_bonnes_reponses)
    #   Afficher la mention
    if nb_erreurs == 0:
        print('Excellent !')
    elif nb_erreurs == 1:
        print('Très bien.')
    elif nb_erreurs <= 3:
        print('bien.')
    elif nb_bonnes_reponses >= 4:
```

```
47          print('Moyen.')
48      elif nb_bonnes_reponses == 0:
49          print("Est-ce que tu l'as fait exprès ?")
50      else:
51          print('Il faut retravailler cette table.')
52
53  reviser_table()
```

2.1. Retrouver tous les raffinages. On ne détaillera pas les dernières actions complexes, celles dont le raffinage ne contient que des actions élémentaires.

Pour cette question, il ne s'agit pas de faire preuve d'imagination : la réponse est dans le programme fourni ! Il suffit juste de retrouver les actions complexes et de les mettre en forme en respectant la présentation des raffinages utilisée dans le cours.

2.2. Ajouter les flots de données.

En analysant le code d'une action complexe, on identifie les données qu'elle manipule. Attention à ne pas confondre *donnée manipulée* par l'action (et donc exploitées par d'autres actions appartenant au même raffinage) et *variable locale* à une action complexe (utilisée seulement en interne de cette action complexe).

2.3. Indiquer les sous-programmes qu'il aurait été judicieux de faire.