1 Bases des interactions avec l'utilisateur

1. Parmi les instructions suivantes effectuées en séquence : lesquelles

- effectuent une saisie au clavier ?
- réalisent un affichage ?
- effectuent un calcul ?
- agissent sur la mémoire ?

a. `x = 3 + 2`
b. `print(x)`
c. `x = input()`
d. `x = input("Valeur pour x ?")`

2. On souhaite :

- afficher à l'écran le message `Entrer une valeur pour x :`
- saisir au clavier une valeur
- associer cette valeur à la variable `x`
- afficher un message `La valeur de x est :'` suivi d'un espace suivi de la valeur de `x`

Pensez-vous que les instructions suivantes sont syntaxiquement correctes, si oui réalisent-elles cette spécification ?

a.
```
x = input("Entrer une valeur pour x : ")
print('La valeur de x est :', x)
```

b.
```
x = input(print("Entrer une valeur pour x : "))
print('La valeur de x est :', x)
```

c.
```
print("Entrer une valeur pour x : ")
x = input()
print('La valeur de x est :', x)
```

d.
```
input("Entrer une valeur pour x : ")
print('La valeur de x est :', x)
```

e.
```
input(x)
print('La valeur de x est :', x)
```

f.
```
x = input("Entrer une valeur pour x : ")
print('La valeur de x + 1 est :', x+1)
```

g.
```
input("Entrer une valeur pour x : ", x)
print('La valeur de x est :', x)
```

2 Sémantique des instructions réalisant une interaction avec l'utilisateur

On donne les fonctions suivantes (pour éviter les noms à rallonge dans le sujet on a utilisé exceptionnellement `aff` au lieu de `affiche` et `s` au lieu de `saisie`) :

```
def incr(y:int):
    return y+1

def aff_msg(y:int):
    print("La valeur de y est :", y)

def s_incr():
    r = input()
    return int(r)+1
```

```
def aff(y:int):
    print(y)

def saisie():
    r = input()
    return r

def s_entier_msg():
    r = input("Entrez un entier :")
    return int(r)
```

```
def aff_incr(y:int):
    print(y+1)

def s_entier():
    r = input()
    return int(r)

def aff2(ch:str):
    print(ch)
```

3. Qu'est-ce qu'une procédure ? Parmi ces fonctions, lesquelles sont des procédures ?

4. Ajouter dans les signatures des annotations de type pour les valeurs retournées par ces fonctions.
5. En représentant la mémoire du programme, l'entrée standard et la sortie standard, dérouler l'exécution des instructions ou l'évaluation des expressions suivantes (sans détailler le mécanisme de l'appel de fonction, l'objectif est de comprendre le fonctionnement des interactions avec l'utilisateur via `print` et `input`) :

1. `q = incr(1) + 1`
2. `d = incr(incr(1))`
3. `aff(2)`
4. `aff2('sel')`
5. `v = aff_incr(1)`
6. `aff(incr(1))`
7. `aff(aff(1))`
8. `n = saisie()`
9. `m = s_entier()`
10. `aff_msg(s_entier())`
11. `t = s_entier_msg()`
12. `i = s_incr(s_incr())`

6. Quelles fonctions ne respectent pas les bonnes pratiques préconisées dans l'UE ?

3 Compréhension et récriture de code

On donne le code suivant :

```
if __name__ == '__main__':
    # éxécuté qd ce module n'est pas initialisé par un import.
    entree = input("Entrer un nombre positif : ")
    nombre = int(entree)
    if nombre < 1000:
        print("Vous voulez : 1000")
    elif nombre%1000 == 0:
        print("Vous voulez :", nombre)
    else:
        print("Vous voulez :", (nombre//1000 + 1)*1000)
```

7. Expliquer pourquoi ce code n'est pas conforme aux bonnes pratiques préconisées dans l'UE.
8. Produire un nouveau code, conforme aux bonnes pratiques et correct.

4 Décomposition : Le retour des flyers

On réutilise l'exercice "Impression à tarif dégressif" de la semaine 4 (exercice de programmation).

On souhaite écrire un programme qui produit la sortie suivante :

```
Combien de flyers voulez-vous commander ? 400
Il vous sera livré 500 flyers pour un montant de 45E
```

9. Dans l'ordre :

- se rappeler quelles fonctions ont été codées en semaine 4. Réalisent-elles des affichages ? Des calculs ? Des saisies ?
- sur la base des bonnes pratiques vues à l'exercice précédent, commencer par réfléchir aux fonctions nécessaires à la réalisation de ce programme et qui n'ont pas été codées en semaine 4. Ne pas écrire ces fonctions intermédiaires.
- toujours sans écrire ces fonctions intermédiaires, écrire une fonction `main()` sans paramètres qui réalise des appels à ces fonctions. Même si la fonction `main` ne peut pas être exécutée tant que les fonctions intermédiaires ne sont pas écrites, c'est intéressant d'écrire cette sorte de scénario d'utilisation.