Chapitre 1 : Séquentialité et variables - TD

Exercice 1 : Séquences d'instructions*

Qu'affichent les programmes suivants ?

```
In [ ]: print(1)
        print(2)
        print(3)
```

```
In [ ]: print(2)
        print(1)
        print(3)
```

Exercice 2 : Noms de variables*

Parmi ces exemples, seuls certains sont des noms de variable valides. Lesquels ?

- `x`
- `x1`
- `X1`
- `toto`
- `éric`
- `_eric`
- `t_42`
- `42_t`

Exercice 3 : Séquentialité et modifications successives des valeurs des variables*

Qu'affiche le programme suivant ? Pour répondre à la question, dessiner les cases mémoire à chaque étape de l'exécution du programme.

```
In [ ]: toto = 4
        print(toto)
        toto = 5 + 5
        print(toto)
        tata = toto + 4
        print(tata)
        tata = tata + 5
        print(toto)
        tata = tata + (toto*2)
        print(tata)
```

Exercice 4 : Séquentialité et valeur des variables*

Qu'affiche le programme suivant ?

```
In [ ]: a=1
        b=2
        a=b
        b=a
        print(a)
        print(b)
```

Qu'affiche le programme suivant ?

```
In [ ]: a=1
        b=2
        b=a
        a=b
        print(a)
        print(b)
```

Exercice 5 : Echange des valeurs de deux variables*

Que manque-t-il aux programmes précédents pour réaliser un échange entre les valeurs des variables a et b ?
Écrire un programme qui réalise un tel échange.

Exercice 6 : Types numériques*

A chaque ligne du programme suivant, donner la valeur et le type des variables à gauche de l'opérateur d'affectation.

```
In [ ]: a = 1.0
        b = 2
        c = a + 1
        d = b + 3
        c = float(d)
        a = c / b
        a = int(c) / b
```

Exercice 7 : Types et addition*

Donner la valeur des expressions suivantes (Attention à la présence des guillemets):

- `123 + 123`
- `"123" + "123"`
- `123 + "123"`
- `"123 + 123"`

Exercice 8 : Concaténation de chaînes de caractères*

On considère le programme suivant :

```
In [ ]: cp = 93
        nom = 'Seine Saint-Denis'
        phrase = ????
        print(phrase)
```

Que mettre à la place des `?????` pour que le programme affiche: `le code postal de la Seine Saint-Denis est 93` ?

Exercice 9 : Saisie de chaînes de caractères au clavier*

Qu'affiche le programme suivant, à supposer que l'utilisateur saisisse `123` puis `456` ?

```
In [ ]: a = input()
        b = input()
        c = a + b
        print(c)
```

Quels sont les types de `a`, `b` et `c` ?

Exercice 10 : Saisie de nombres au clavier*

Comment faudrait-il modifier ce programme pour qu'il affiche `579`, à supposer que l'utilisateur saisisse `123` puis `456` ?
NB : `579` est la somme de `123` et de `456`.
Désormais, quels sont les types des différentes variables ?

Exercice 11 : Adaptation de programmes existants*

Que faut-il modifier à ce programme pour que ce soit le périmètre qui soit calculé ?

```
In [ ]: print('Quelle est la longueur du premier cote ?')
        cote1 = float (input())
        print('Quelle est la longueur du deuxieme cote ?')
        cote2 = float(input())
        surface = cote1 * cote2
        print('La surface du rectangle ainsi forme est' + str(surface))
```

Exercice 11 : Calcul de clôture*

Écrire un programme qui demande à l'utilisateur les dimensions en mètres d'une zone rectangulaire à clôturer et qui affiche un message intelligible indiquant le nombre de piquets à utiliser, sachant qu'il faut au maximum 2 mètres entre chaque piquet. On suppose que les dimensions en mètres sont des nombres entiers.

Exercice 12 : Echange de nombres sans variable intermédiaire**

Écrire un programme qui demande à l'utilisateur deux nombres, les affiche, les échange, et les réaffiche après échange, mais sans utiliser de variable intermédiaire. On pourra utiliser une addition et une soustraction pour ne rien perdre.
Remarque : Cette méthode d'échange sans variable temporaire peut paraît meilleure que la méthode classique. Elle nécessite cependant que l'ordinateur effectue des additions/soustractions en plus. Surtout elle est beaucoup moins générale car elle suppose que les variables à échanger sont de type numérique : cette méthode ne fonctionne pas sur des chaînes de caractères par exemple. D'une manière générale, il y a souvent plusieurs manières de résoudre un problème. Le rôle du programmeur consiste à identifier le meilleur algorithme (le meilleur enchaînement d'instructions), compte tenu des hypothèses qu'il lui semble raisonnable de faire.