TD 1 : Types de données, opérateurs, structures de contrôle

Exercice 1.1 : Opérations arithmétiques et conversions

Réaliser un programme qui lit deux nombres entiers `a` et `b` du clavier avec la fonction `input()`. Ensuite, il affichera la fraction `a/b`, sa valeur numérique en division entière et sa valeur numérique en division flottante. Exemple :

```
Entrez a: 42
Entrez b: 10
42/10 = 4 en division entière et 42/10 = 4.2 en division flottante.
```

Exercice 1.2 : Types de séquence

(a) Faire apparaître sur l'écran la phrase exacte suivante :
`L'anti-slash "\" sert de caractère d'échappement.`

(b) Réaliser un programme qui demande à l'utilisateur d'entrer trois mots par le clavier et qui les affiche sur l'écran
— dans l'ordre inverse qu'ils ont été entrés
— en supprimant le premier caractère
— en supprimant tous les caractères à part le premier
— sans espaces.
Exemple :

```
Entrez le premier mot:chien
Entrez le deuxième mot:chat
Entrez le troisième mot:souris
souris chat chien
hien hat ouris
c c s
chienchatsouris
```

(c) Réaliser un programme qui demande à l'utilisateur d'entrer un jour dans l'année non bissextile (un nombre j entre 1 et 365) et le jour de la semaine qui correspond au 1er janvier (un nombre entre 1 et 7). Ensuite le programme affichera la semaine et le jour de la semaine qui correspondent au jour j (on numérote les semaines de façon que le 1er janvier est dans la semaine 1). Se servir d'une liste dont les éléments sont les chaînes de caractère `"lundi"` à `"dimanche"`. Exemple :

```
Entrez un jour entre 1 et 365: 77
Le 1er janvier tombe quel jour de la semaine (entre 1 et 7): 2
Le jour 77 est le lundi de la semaine 12.
```

Exercice 1.3 : Priorité des opérateurs

Éliminer les parenthèses superflues dans les expressions suivantes :

```
(a + b) - (2 * c)
(-a) / (-(b + c))
(2 * x) / (y * z)
(x / y) % (-z)
(x + 3) * (n % p)
(x / (y % z) > 0) and (z < 0)
```

Exercice 1.4 : Structure conditionnelle

(a) Réaliser un programme qui teste qu'une année (qui sera entrée au clavier par l'utilisateur) est bissextile. On rappelle que les années bissextiles reviennent tous les 4 ans, sauf les années séculaires, si celles-ci ne sont pas multiples de 400. Ainsi, 1900 n'était pas une année bissextile, alors que 2000 l'était.

(b) Réaliser un programme qui lit trois nombres flottants a, b, c du clavier et qui affiche le nombre des solutions réelles x de l'équation $ax^2 + bx + c = 0$. (Il n'y a pas besoin de calculer les valeurs des racines. Attention aux cas spéciaux où un ou plusieurs coefficients sont 0 !)

Exercice 1.5 : Boucles

(a) Réaliser un programme qui demande à l'utilisateur de deviner un nombre (qui sera entré par le clavier) et ne termine pas avant d'avoir eu la bonne réponse, 42. Utiliser une boucle `while`.

(b) Réaliser un programme qui lit un entier naturel n du clavier, teste que $n \geq 0$ et calcule $n!$ avec une boucle `for`.

(c) Réaliser un programme qui affiche toutes les factorielles des nombres entiers entre 0 et 20.

(d) Réaliser un programme qui lit un nombre positif k du clavier, teste que $k > 0$, et affiche tous les éléments f_i de la suite de Fibonacci inférieurs à k. (Définition : $f_0 = 0$, $f_1 = 1$, et $f_{n+2} = f_{n+1} + f_n \; \forall \, n \in \mathbb{N}$).

(e) Réaliser un programme qui lit un mot `mot` du clavier et qui indique ensuite si `mot` est un palindrome ou pas (un mot dont l'ordre des lettres reste le même qu'on le lise de gauche à droite ou de droite à gauche).

(f) Réaliser un programme qui lit une phrase `phrase` et une seule lettre `lettre` du clavier et qui affiche l'indice de la première occurrence de `lettre` dans `phrase`. Si `lettre` ne figure pas dans `phrase`, le programme terminera avec un message d'erreur. *Indication :* pour obtenir la longueur d'une chaîne de caractères `s`, utiliser la commande `len(s)`.

(g) Réaliser un programme qui lit un nombre entier `n` du clavier et calcule la somme `s` des chiffres individuelles dans `n`. Ensuite, tant que `s` > 9 le calcul est répétée avec `s` à la place de `n`. Par exemple, pour `n` $= 276$ on a $2 + 7 + 6 = 15$ et $1 + 5 = 6$.

(h) Réaliser un programme qui trouve tous les *nombres parfaits* entre 1 et 10 000. (Un nombre parfait est un nombre qui est la somme de ses propres diviseurs, p.ex. $6 = 3 + 2 + 1$ est un nombre parfait.)