TD2 : STRUCTURES DE CONTRÔLE

1. Expressions booléennes

Exercice 1. Vérité
Cet exercice est à effectuer dans la console Python.
Quelle est la valeur des expressions suivantes :

```
>>> True
>>> 1 + 1 == 2
>>> False
>>> 1 + 1 == 3
>>> type(True)
>>> type(False)
>>> type(1 + 1 == 2)
>>> type(True) != type(2)
```

Quelle est la valeur de vérité des expressions suivantes :

```
>>> True and True
>>> True and False
>>> False and False
>>> False and True
>>> True or True
>>> True or False
>>> False or False
>>> False or True
>>> True and (True or False)
>>> not False
>>> not True
```

Exercice 2. Expressions conditionnelles
Pour chaque expression suivante, prédire si sa valeur est `True` ou `False`, puis vérifier le résultat dans l'interpréteur.

```
>>> 3 == 4
>>> 3 != 4
>>> 3 <= 4
>>> 3 + 4 >= 7
>>> 3 + 4 == 7 or 3 <= 7 - 5
>>> 24 % 2 == 0 and 24 % 3 == 0
```

Exercice 3. Écriture de conditions
Initialiser les variables 'a', 'b' et 'c' respectivement aux valeurs 2, 5 et 7. Déterminer si les tests suivants sont vrais ou faux en utilisant les opérateurs booléens et les opérateurs de comparaison correspondants :

- a égale à b
- (a + b) différent de c
- non (a égale à c)
- ((a + b) supérieur ou égal à c) et (a inférieur ou égal à (c - b))
- ((a + c) > b) ou ((a + c) < b)

2. Instruction if

Exercice 4. La moyenne
Rappel : `x = float(input('message'))` *permet de faire saisir un nombre à l'utilisateur et d'en récupérer la valeur dans la variable 'x'.*
Écrire un programme qui demande une note à l'utilisateur et affiche soit "Vous avez la moyenne" soit "Vous n'avez pas la moyenne" selon que la note est supérieure ou non à 10.
Modifiez ce programme pour qu'il affiche "La note est invalide" si la note entrée n'est pas entre 0 et 20.

Exercice 5. Table d'exécution
Soit le programme suivant :

```
x = int(input('x ? '))
y = int(input('y ? '))
z = x + y
if z % 2 == 0:
  print('x + y est pair')
else:
  print('x + y est impair')
```

→ Écrire une table d'exécution du programme lorsque l'utilisateur entre 2 puis 3.
→ Écrire une table d'exécution du programme lorsque l'utilisateur entre 5 puis 7.

Exercice 6. Programme attentionné
Écrire un programme python qui pose la question "Bonjour, comment ça va ?". Si l'utilisateur répond "Bien" alors il devra afficher "Super ! Je suis content pour vous." Si la réponse est différente de "Bien" il affichera "Mince... courage !".

Exercice 7. Au cinéma
Écrire un programme qui demande son âge à l'utilisateur : "Quel est votre âge ?" et s'il oui ou non il souhaite des popcorns : "Souhaitez-vous des popcorns ?". Le programme affiche alors le prix de sa séance de cinéma. Si la personne est mineure, le prix d'une place est de 7€, si la personne est majeure le prix est de 12€. Le popcorn est à 5€.

Exercice 8. La machine qui rend la monnaie
Vous travaillez pour une boulangerie qui souhaite mettre en place une machine à rendre la monnaie.

(1) Écrire un programme `monaie.py` qui lit une somme exprimée en euros, sans centimes, et affiche le nombre minimal de billets et de pièces de 2 et 1 euros nécessaires pour la composer.

Il s'agit pour le moment de réaliser un script sans boucles ni instruction conditionnelle. Vous avez donc le droit de donner des réponses maladroites, pourvu qu'elles soient justes.

Exemple d'exécution

```
$ python monnaie.py
Somme ? 1949
1949 = 3 x 500 + 2 x 200 + 0 x 100 + 0 x 50 + 2 x 20 + 0 x 10 + 1 x 5 + 2 x 2 + 0 x 1
$
```

Indication : Si `x` et `y` sont des expressions à valeur numérique, alors `x // y` vaut le quotient entier et `x % y` le reste de la division de x par y. Pour faciliter votre tâche, vous pouvez commencer par une machine ne contenant que des billets de 5 euros, et des pièces de 2 et 1 euros.

(2) Améliorez la solution précédemment donnée de ce problème, de manière à éviter les maladresses que nous avons constatées. On doit obtenir maintenant, par exemple :

```
$ python monnaie.py
Somme ? 1949
1949 = 3 x 500 + 2 x 200 + 2 x 20 + 1 x 5 + 2 x 2
$
```

Note : pour afficher du texte sans retour à la ligne, vous pouvez utiliser le paramètre `end=''` *de* `print`*. Par exemple,* `print('pi =', 3.14, end='')`*.*

Exercice 9. Jour de la semaine
Écrire un programme qui calcule le jour de la semaine correspondant à une date donnée, supposée correcte, exprimée sous forme de trois nombres entiers j (jour), m (mois) et a (année). Exemple

```
$ python joursemaine.py
jour? 24
mois? 9
annee? 2009
le 24/9/2009 est un jeudi
$
```

Utilisez les formules suivantes :

$$m_1 = \begin{cases} m-2 & \text{si } m \geq 3 \\ m+10 & \text{sinon} \end{cases}$$

$$a_1 = \begin{cases} a & \text{si } m \geq 3 \\ a-1 & \text{sinon} \end{cases}$$

n_s = le siècle compté à partir de 0 (c'est-à-dire tel qu'exprimé par les deux premiers chiffres de a_1), a_s = l'année dans le siècle (exprimée par les deux derniers chiffres de a_1), puis

$$f = j + a_s + \left\lfloor \frac{a_s}{4} \right\rfloor - 2 \times n_s + \left\lfloor \frac{n_s}{4} \right\rfloor + \left\lfloor \frac{26 \times m_1 - 2}{10} \right\rfloor$$

où $\lfloor \frac{a}{b} \rfloor$ correspond au quotient de la division de a par b. Dans ces conditions, le jour de la semaine est donné par le reste de la division de f par 7 ($0 \Rightarrow$ *dimanche*, $1 \Rightarrow$ *lundi*, $\ldots 6 \Rightarrow$ *samedi*).
Ci-dessus, les barres de fraction indiquent des divisions entières (c'est-à-dire des quotients par défaut).
→ Proposez des noms plus explicites pour les variables de votre programme que ceux apparaissant dans les formules.

3. Instruction while

Note : la combinaison de touches `Ctrl-C` *permet d'arrêter un programme bouclant à l'infini.*

Exercice 10. Compte à rebours
Écrire un programme qui compte à rebours de 10 à 1 et affiche "Décolage !" une fois le décompte terminé. Ce programme devra utiliser une boucle `while`.

Exercice 11. Nombre premier
Écrire un programme qui demande à l'utilisateur d'entrer un entier, puis détermine si ce dernier est *premier*, c'est à dire qu'il n'est pas divisible par autre entier que lui-même ou 1. x est divisible par y si `x % y == 0`.
→ Faire une table d'exécution pour ce programme lorsque l'utilisateur entre 4, puis lorsqu'il entre 5.

Exercice 12. Moyenne d'une suite de nombres lus successivement
Écrire un programme qui lit au clavier une suite de nombres positifs ou nuls (tapés successivement, à raison d'un par ligne) et en calcule la moyenne. On ne sait pas à l'avance combien il y a de nombres, c'est la frappe d'une valeur négative qui indique la fin de la suite. Exemple

```
$ python moyenne.py
? 10
? 8
? -1
moyenne: 9
$
```

Indication. Sous son air innocent, cet exercice est très important car il illustre une situation que vous rencontrerez souvent : traiter une suite de données dont on ne connaît pas à l'avance le nombre, acquises successivement, jusqu'à la rencontre d'une donnée invalide qui indique la fin de la suite.
Attachez-vous à écrire un programme obéissant au schéma universel et fiable suivant :

— acquérir une donnée x
— tant que x est valide :
— traiter x (cela dépend du problème particulier considéré)
— acquérir une donnée x

La plupart des langages actuels disposent d'une instruction d'abandon de boucle, comme `break` en Python. Un autre schéma universel et fiable est alors celui-ci :

— répéter indéfiniment :
— acquérir une donnée x
— si x est invalide : abandonner la boucle
— traiter x (cela dépend du problème particulier considéré)

→ Remplir une table d'exécution pour l'exemple donné ci-avant (l'utilisateur entre 10, 8, -1).

Exercice 13. Calcul d'une suite
Soit la suite définie par :

$$U_0 = 1$$
$$U_1 = -1$$
$$U_n = \frac{U_{n-1}}{2} + 2U_{n-2}$$

Écrire un programme qui calcule et affiche U_n pour un n entré par l'utilisateur itérativement, c'est à dire en utilisant une boucle.

Exercice 14. Tables de multiplication
Écrire un programme qui affiche les tables de multiplications des entiers de 1 à 10.

Exercice 15. Tabulation d'une fonction
Écrivez un programme qui affiche les valeurs du sinus des angles de 0 à 90 degrés par pas de 15 degrés. On souhaite un affichage respectant strictement la forme suivante :

```
sin(0) = 0.000000
sin(15) = 0.258819
sin(30) = 0.500000
...
sin(90) = 1.000000
```

Note : vous pouvez calculer le sinus d'un angle avec la fonction `math.sin(angle)` *ou l'angle passé en paramètre est en radians. Pour que cette fonction soit disponible, votre programme doit commencer par* `import math`.

Exercice 16. La date du lendemain
Écrivez un programme qui acquiert au clavier une date, exprimée sous forme de trois nombres entiers `j` (jour), `m` (mois) et `a` (année) et qui affiche la date du lendemain.
Exemple :

```
$ python lendemain.py
jour? 30
mois? 9
année? 2009
aujourd'hui: 30 9 2009
demain: 1 10 2009
$
```

Vous pouvez supposer que la date saisie est correcte. Suggestion. Vous aurez fait le plus gros du travail quand vous aurez écrit une expression logique, portant sur les nombres `j`, `m` et `a`, qui traduit exactement la proposition « `(j, m, a)` est le dernier jour du mois ».
Rappel. Les années bissextiles sont les années multiples de 4 qui ne sont pas multiples de 100, et les années multiples de 400.

Exercice 17. Tester la fonction random
La fonction `random()` du module random [1] renvoie, chaque fois qu'on l'appelle, un nombre flottant x vérifiant $0 \leq x < 1$, la suite des nombres successivement produits étant pseudo-aléatoire. Pour afficher une valeur aléatoire, on peut procéder de la façon suivante :

```
import random
x = random.random()
print(x)
```

Pour tester (très succinctement) cette fonction, écrivez un programme qui fait un grand nombre d'appels de cette fonction et qui calcule la moyenne et l'écart-type des valeurs obtenues.
Rappels. Étant donnée une suite de n nombres x_i $0 \leq i < n$, sa moyenne est définie par

$$\overline{x} = \frac{1}{n} \sum_{i=0}^{n-1} x_i$$

et son écart-type par

$$\rho = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} x_i^2 - \overline{x}^2}$$

Si la fonction random était parfaite et le nombre de tirages infini, la moyenne vaudrait $\frac{1}{2}$ et l'écart-type $\frac{1}{\sqrt{12}} \simeq 0,28867513459481292$
Note : la racine carrée se calcule avec la fonction `math.sqrt` *du module* `math`.

1. Voir `http://docs.python.org/library/random.html`