TD 2 : STRUCTURES DE CONTRÔLE

1. Programmes avec des affectations, des lectures et des écritures

Exercice 1. La machine qui rend la monnaie

(1) Ecrivez un programme 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
$
```

Indications. Si `x` et `y` sont des expressions entières (de type `int` ou `long`), alors `x//y` vaut le quotient entier (on dit aussi quotient « par défaut ») et `x%y` le reste de la division de x par y.

(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
$
```

(3) Réécrivez une nouvelle version de ce programme en réduisant significativement le nombre de ses lignes par l'utilisation d'une une liste constante représentant les coupures disponibles et une instruction de boucle *for*.

Exercice 2. Jour de la semaine d'une date donnée

(1) Écrivez 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

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

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).

2. Programmes avec des instructions conditionnelles *if*

Exercice 3. La date du lendemain

(1) 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
annee? 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.

3. Programmes avec des boucles *while*

Exercice 4. Moyenne d'une suite de nombres lus successivement

(1) Ecrivez 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
? 12
? 8
? -1
moyenne: 10
$
```

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)

Exercice 5. Tester la fonction random

(1) La fonction `random()` du module random (voyez http ://docs.python.org/library/random.html) 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 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. Etant 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

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

Note. Si la fonction random était parfaite et le nombre de tirages infini, la moyenne vaudrait $\frac{1}{2}$ et l'écart-type $\frac{1}{\sqrt{12}} = 0,28867513459481292$

4. Programmes avec des boucles *for*

Exercice 6. Calcul de la factorielle

(1) Écrivez un programme qui calcule itérativement (c'est-à-dire selon un algorithme basé sur une boucle, ici une boucle for) la valeur de la factorielle d'un nombre donné. Exemple :

```
$ python factorielle.py
n ? 50
50 ! = 30414093201713378043612608166064768844377641568960512000000000000
$
```

Rappel. La factorielle d'un nombre n vérifiant $n \geq 0$ est le nombre

$$n! = n \times (n-1) \times (n-2) \times \ldots \times 3 \times 2 \times 1$$

Exercice 7. Tabulation d'une fonction

(1) 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
```

Exercice 8. Tables de multiplication

(1) Ecrivez un programme qui affiche les tables de multiplications des entiers de 1 à 10.