Solution des premiers exercices

1 Exercice : Passage de base 2 à base 10 et inversement

Question : Quelle est la représentation en base 10 de 101101 écrit en base 2 ?

Les chiffres 0 et 1 sont les coefficients des puissances successives de 2, le chiffre le plus à droite (les unités) correspondant à la puissance 0.

1	0	1	1	0	1	les bits
2^5	2^4	2^3	2^2	2^1	2^0	les puissances de 2
32	16	8	4	2	1	puissances calculées
32	0	8	4	0	1	produits (bit par puissance)

La somme des produits donne 45

Question : Quelle est la représentation en base 2 de l'entier 23 en base 10 ?

Principe : On veut écrire $n = 23$ sous la forme $b_i.2^i + ... + b_2.2^2 + b_1.2^1 + b_0.2^0$

Le coefficient le plus simple à trouver est b_0 : `n % 2` (reste de la division entière de n par 2).

On peut reformuler l'expression précédente sous la forme (factorisation de 2) : $n = (b_i.2^{i-1} + ... + b_2.2^1 + b_1.2^0) \times 2 + b_0.2^0$ qui est de la forme : $n = n' * 2 + (n$ `%` $2)$ n' est la division entière de n par 2.

On retombe sur un problème similaire : trouver le b_i pour n' (qui a un bit de moins que n).

On pourra donc refaire les mêmes opérations : $n' = n'' * 2 + (n'$ `%` $2)$ et ainsi de suite.

```
n   n % 2
23  1               23 = 11 * 2 + 1
11  1               11 =  5 * 2 + 1
 5  1                5 =  2 * 2 + 1
 2  0                2 =  2 * 1 + 0
```

```
1  1                1 =  0 * 2 + 1
0
```

La représentation de 25 est donc 10111.

2 Exercice : Expressions, opérateurs et priorité

1. Parenthéser les expressions suivantes (pour expliciter l'ordre de calcul)

```
(x != ((y * 3) + 5))
((n != 0) and ((s / n) >= 10))
(((y >= b) and ((z ** 2) != 25)) and (not (x in e)))
```

   Attention, pour la dernière expression, la priorité de `not` est 5 à ne pas confondre avec la priorité de `not in` ou `is not` qui est de 6. Le premier est l'opérateur unaire booléen de négation logique, les deux autres sont des opérateurs binaires (non appartenance d'un élément et non égalité physique).

2. Comment s'évaluent les expressions suivantes ?

```
x = (12)              # Remarque : on parenthèse l' « expression », pas la variable
y = ((6 - 3) - 2)     # donc y est associé à 1
z = (2 ** (1 ** 3))   # ** associatif à droite, z associé à 2 (en non à 8)
```

3. Que signifient les expressions suivantes ?

```
0 <= x <= 9     # équivalent à 0 <= x and x <= 9
x < 10 < y      # équivalent à x < 10 and 10 < y
x < 10 >= y     # équivalent à x < 10 and 10 >= y
x < 10 == y     # équivalent à x < 10 and 10 == y
x == y == 0     # équivalent à x == y and y == 0
```

4. Réécrire les expressions suivantes sans utiliser `not`

```
(1 > mois) or (mois > 12)              # parenthèses inutiles car or moins prioritaire que >
reponse != 'oui' and reponse != 'non'  # mais elle peuvent améliorer la lisibilité
```

   Pour 1 : On a fait `not (A and B) ~ (not A) or (not B)` puis `not (1 <= mois) ~ 1 > mois`

3 Exercice : Opérateurs artithmétiques

Pour chacune des questions, répondre avant de vérifier sur l'interpréteur Python si votre réponse est juste.

1. Quel est le résultat de 9 / 4 ? `2.25` (un nombre réel)
2. Quel est le résultat de 9 // 4 ? `2` (un entier)
3. Quel est le résultat de 9 % 4 ? `1` (le reste de la division entière)
4. Comment obtenir 2^{10} et quelle est sa valeur ? `2 ** 10` (opérateur puissance), `1024`
5. Quels sont la valeur et le type de 8 / 4 ? `2.0` (un nombre réel, `/` a toujours un résultat réel)

4 Exercice : Chiffres d'un entier

Soit un entier n, quelle expression permet d'obtenir :

1. le chiffre des unités (1 pour 421, 5 pour 35, 0 pour 0...)

   ```
   n % 10
   ```

2. le chiffre des dizaines (2 pour 421, 3 pour 35, 0 pour 0...)

   ```
   (n % 100) // 10      # solution 1
   (n // 10) % 10       # solution 2
   ```

3. le chiffre des dizaines **et** le programme ne doit utiliser que 10 comme constante littérale

   ```
   (n // 10) % 10       # remarque : on pourrait ne pas mettre les parenthèses
   ```

4. le chiffre des centaines

   ```
   ((n // 10) // 10) % 10  # marche aussi sans les parenthèses
   ```

5 Exercice : Calculer x^5

Soit x un nombre réel, calculer x^5 (on l'appellera x5) :

```
x = 3       # par exemple
```

1. en utilisant l'opérateur puissance de Python

   ```
   x5 = x ** 5
   ```

2. en n'utilisant que l'opérateur multiplication

   ```
   x5 = x * x * x * x * x     # fastidieux de multiplier 5 fois !
   ```

3. en n'utilisant que l'opérateur multiplication et en faisant au plus 3 multiplications

   ```
   x2 = x * x
   x5 = x2 * x2 * x
   ```

   On a bien fait 3 multiplications !

6 Exercice : Afficher le résultat d'opérations

Soit a = 503 et b = 17, afficher, en utilisant a et b et aucune constante littérale, le texte suivant « 503 / 17 = 29.58823529411765 » en utilisant `print` :

1. avec plusieurs paramètres (séparés par des virgules).

```
a = 503 ; b = 17
print(a, '/', b, '=', a / b)
```

2. avec la méthode `format`.

```
a = 503 ; b = 17
print('{} / {} = {}'.format(a, b, a / b))
```

3. en utilisant les chaînes formatées

```
a = 503 ; b = 17
print(f'{a} / {b} = {a / b}')
```

7 Exercice : Améliorer l'affichage des nombres réels

Reprendre l'exercice précédent en affichant le résultat de la division avec 3 chiffres après la virgule.

```
a = 503 ; b = 17
print(a, '/', b, '=', int(a / b * 1000) / 1000)
print(a, '/', b, '=', round(a / b, 3))
print('{} / {} = {:.3f}'.format(a, b, a / b))
print(f'{a} / {b} = {a / b:.3f}')
```

La fonction `round` prend deux paramètres, d'abord le nombre à arrondir et ensuite le nombre de chiffre souhaité après la virgule.

Quand on ne met pas le nombre de position, c'est comme si on avait mis 0 ou 1. `.3f` est identique à `1.3f`.

8 Exercice : afficher plusieurs objets

On considère l'instruction `print(1, 2, 3, 4, 5, 6)`.

1. Quel est l'effet de l'instruction précédente ?

Il y aura une espace entre chaque objet (1, 2, ..., 6) et un retour à la ligne à la fin.

```
1 2 3 4 5 6
```

2. Compléter l'instruction après le paramètre 6 pour obtenir l'affichage suivant :

```
1 -> 2 -> 3 -> 4 -> 5 -> 6...
```

Entre chaque entier (objet), on a `->`. C'est le séparateur (`sep`). À la fin on a `...` et certainement un retour à la ligne. C'est la valeur à donner au paramètre `end` de `print`.

```
print(1, 2, 3, 4, 5, 6, sep=' -> ', end='...\n')
```

Les deux caractères `\n` correspondent à un caractère retour à la ligne.

9 Exercice : Réaliser un affichage tabulé

Soit les variables suivantes : `nom = 'Paul' ; note = 15.5 ; matiere = 'programmation'`. Écrire dans l'ordre le nom sur 7 positions, la note sur 6 positions avec 2 chiffre après la virgule et la matière sur 10 positions (« `Paul 15.50 programmation` ») en utilisant :

1. la méthode `format` de `str`

```
nom = 'Paul' ; note = 15.5 ; matiere = 'programmation'
print("{:7} {:6.2f} {:>10}".format(nom, note, matiere))
nom = 'Lucie' ; note = 18.0 ; matiere = 'chimie'
print("{:7} {:6.2f} {:>10}".format(nom, note, matiere))
```

Le `>` permet d'aligner le texte à droite. Un `<` aligne à gauche et `^` centre.

2. les chaînes formatées

```
nom = 'Paul' ; note = 15.5 ; matiere = 'programmation'
print(f"{nom:7} {note:6.2f} {matiere:>10}")
nom = 'Lucie' ; note = 18.0 ; matiere = 'chimie'
print(f"{nom:7} {note:6.2f} {matiere:>10}")
```

10 Exercice : Saisie au clavier

On veut avoir le dialogue suivant avec l'utilisateur. Après les points d'interrogation apparaît ce qui a été saisi par l'utilisateur.

```
Nom ? Paul
Note ? 15.5
Coefficient ? 4
```

1. Réaliser cette saisie sachant que l'on veut initialiser les variables :
   - `nom` : une chaîne de caractère,
   - `note` : un réel et
   - `coefficient` : un entier.

```
nom = input("Nom ? ")
note = float(input("Note ? "))
coefficient = int(input("Coefficient ? "))

print(f"{nom} a eu {note} coefficient {coefficient}.")
```

2. Que se passe-t-il si l'utilisateur répond « dix » pour la note ?

   Le programme se termine sur l'exception `ValueError` car la fonction `float` ne sait pas convertir la chaîne `"dix"` on un réel.

3. Que se passe-t-il si l'utilisateur répond « 4.5 » pour le coefficient ?

   Le programme se termine sur l'exception `ValueError` car la fonction `int` ne sait pas convertir la chaîne `"4.5"` en un entier.