Les trucs en plus

Afin de faire des exercices d'entraînement plus variés, nous allons utiliser des opérateurs et des instructions nouvelles. Nous vous les présentons car, de toute façon, ils font partie des classiques de la programmation.

1. Les paramètres optionnels de la fonction print

Lorsque la fonction `print()` reçoit plusieurs arguments, par exemple `print("un","deux")`, elle les affiche en les séparant par un espace. Qu'elle reçoive un ou plusieurs arguments, voire aucun, cette fonction positionne ensuite le curseur à la ligne suivante produisant un retour à la ligne.

Il est possible de modifier ces deux comportements en utilisant des paramètres optionnels :

- Le paramètre `sep` permet de changer le séparateur inséré entre les arguments de la fonction `print()`. Par défaut, c'est un espace qui les sépare.
- Le paramètre `end` permet de définir la chaîne qui sera affichée une fois l'affichage des arguments terminés. On peut ainsi contourner le passage à la ligne automatique en fournissant une chaîne vide comme argument. Par défaut, le paramètre `end` correspond au caractère spécial `/n` codant un retour à la ligne.

Voici un exemple :

```
print("Hibou","Caillou")
print("Toto",end="##",sep="AA")
print("Titi","Tata","Tutu",end="KK",sep="BB")
print("============")

>> Hibou Caillou
>> Toto##TitiBBTataBBTutuKK============
```

Le premier `print()` a deux arguments. Ainsi, les deux chaînes "Hibou" et "Caillou" sont séparées par un espace et un retour à la ligne est mis en place après la chaîne "Caillou".

Le deuxième `print()` n'utilise pas le paramètre `sep`, car une seule chaîne est passée en argument. Le paramètre `end` correspondant à l'argument "##". On ajoute cette chaîne à la suite de "Toto" et ainsi le retour à la ligne par défaut est désactivé.

Le troisième `print()` reçoit trois chaînes de caractères "Titi", "Tata" et "Tutu". Elles vont être reliées en utilisant la chaîne "BB" donnée en argument au paramètre `sep`, produisant ainsi l'affichage : TitiBBTataBBTutu. Le paramètre `end` étant fixé à "KK", cette instruction `print()` affiche finalement : TitiBBTataBBTutuKK sans provoquer un retour à la ligne.

Le dernier `print()` affiche une série de `======` accolés à la dernière chaîne.

2. L'opérateur modulo %

L'expression a%b (a modulo b) entre deux entiers a et b retourne le reste de la division euclidienne de a par b. Ainsi, lorsqu'on divise 7 par 4, on obtient 1 et il reste 3. C'est cette valeur 3 qui est retournée par l'opérateur modulo.

L'opérateur modulo peut s'avérer pratique dans certaines situations très précises. Son utilisation est assez

amusante. En fait, si l'on examine la suite i%7, on obtient une série de valeurs 0, 1, 2, 3, 4, 5 et 6 bouclant à l'infini :

```
0%7 = 0     1%7 = 1     2%7 = 2     3%7 = 3     4%7 = 4     5%7 = 5     6%7 = 6
7%7 = 0     8%7 = 1     9%7 = 2    10%7 = 3    11%7 = 4    12%7 = 5    13%7 = 6
14%7 = 0    15%7 = 1    16%7 = 2    17%7 = 3    18%7 = 4    19%7 = 5    20%7 = 6
```

Une particularité de l'opérateur modulo est que le résultat de a%b est compris entre 0 et b-1, ceci quelle que soit la valeur de a, même négative.

Pour calculer de tête le résultat de a%b, utilisez les règles suivantes :

- Si a est compris entre 0 et b-1, le résultat vaut a.
- Si a ≥ b, il faut soustraire plusieurs fois b à la valeur a jusqu'à ce que a<b. Comme dans ce cas, on a aussi a≥0, le résultat vaut a. Ainsi, pour calculer 20%7, on effectue 20-7 → 13 toujours supérieur à 7, donc on continue en calculant : 13-7 → 6, ce qui correspond à la réponse.
- Si a < 0, il faut ajouter plusieurs fois b à la valeur a jusqu'à ce que a≥0. Comme dans ce cas on a aussi a<b, le résultat vaut a. Ainsi, pour calculer -5%7, on effectue -5+7 → 2, ce qui correspond à la réponse.

Ces règles viennent du fait qu'en ajoutant ou en retranchant la valeur b au nombre actuel, on ne change pas la valeur du modulo. Ainsi, par ces manipulations, on cherche à amener la valeur courante de a dans l'intervalle 0,…,b-1 car cela correspond à la plage de valeurs du modulo.

Trouvez les valeurs de :

4%8=	-3%8=	8%8=	15%8=	17%8=
55%100=	-10%100=	188%100=	250%100=	540%100=

Dans le désordre, voici les valeurs à trouver : 55, 4, 7, 88, 5, 90, 0, 50, 40 et 1.

3. L'opérateur puissance

Sa syntaxe est la suivante : `x**n`. Ainsi x^2 s'écrit dans le langage Python : `x**2`.

4. Les opérateurs +=, -= et *=

Dans les pages précédentes, nous avons écrit plusieurs fois : `i = i + 1` ou `a = a * 4`. Maintenant que vous connaissez ces syntaxes, on vous conseille de les remplacer par leur version plus compacte :

- `a = a + 1` → `a += 1`
- `b = b - 4` → `b -= 4`
- `c = c * 7` → `c *= 7`

Cette écriture améliore grandement la lisibilité du programme. Utilisez-la autant de fois que possible.

Nous ne listons par tous les opérateurs compacts disponibles, mais il en existe quasiment un pour tous les

opérateurs disponibles, comme /=.