Initiation interactive à Python 2 : Calculatrice, variables, IO

Aller vers Initiation interactive à Python 1 : Présentation, instructions

Super calculatrice

Un interpréteur de langage informatique contient généralement tout ce qui est nécessaire pour *calculer*. C'est le cas de l'interpréteur Python qui contient, entre autres choses, les opérations arithmétiques usuelles :

Opération	**Symbole**
Addition	+
Soustraction	-
Multiplication	*
Division	/
Division entière	//
Reste de la division	%
Exponentiation	**

Tout le monde doit connaître les 4 premières opérations, ce sont les mêmes que sur une calculatrice de poche. La division entière et le reste de la division fonctionnent ensemble : ce sont les résultats qu'on trouve lorsqu'on *pose* une *division euclidienne*. Par exemple, en divisant 25 par 7, on trouvera que le résultat (le quotient) vaut 3 et qu'il reste 4. En Python : 25//7 vaudra 3 et 25%7 vaudra 4.

L'exponentiation correspond à l'opérateur puissance. Multiplier le nombre 3 avec lui même quatre fois 3x3x3x3 s'écrit 3^4 et se lit *3 à la puissance 4*. En Python, on écrit cette opération 3**4.

On a souvent besoin de ranger des résultats intermédiaires de calculs dans des *variables*. En Python, les *variables* sont des noms qu'on peut donner aux objets, que ce soient des valeurs numériques, ou autre chose (une image, une chaîne de caractères, un fichier....).

La notion de variable est **fondamentale**. Nous en parlons maintenant, mais tant que nous programmerons, nous allons continuer à l'utiliser.

Voici un premier exemple d'utilisation : nous avons deux nombres et nous voulons connaître leur somme et leur produit. Nous pouvons écrire :

```
print(3456712 + 1243567, 3456712 * 1243567)
```

Ce court programme (vous pouvez l'essayer en le copiant dans une zone d'édition interactive) affiche

effectivement la somme et le produit des deux nombres. Il comporte toutefois deux gros défauts :

- si on désire changer l'un ou l'autre des nombres, par exemple 1243567 en 1234567, il faut modifier le code à deux endroits différents
- et comme il faut modifier le code à deux endroits différents, on a deux fois plus de chances de se tromper que s'il y avait une seule modification à faire.

Enfin, pour être sûr que le programme calcule bien la somme et le produit des deux mêmes nombres (quels qu'ils soient), il faut vérifier très attentivement qu'on n'a pas fait de faute de frappe dans le calcul de la somme ou dans le calcul du produit.

Pour éviter ces défauts, on peut donner des noms aux deux nombres (par exemple a et b), puis écrire le reste du code en utilisant les noms plutôt que les nombres :

Après l'avoir exécuté, modifiez le programme pour qu'il calcule la somme et le produit de 3456712 (le nombre ne change pas) avec 1234567 (celui-là a a changé).

Prenez bien conscience des avantages de cette écriture :

- si l'un des deux nombres change, le code n'est modifié qu'à un seul endroit
- il est très facile de vérifier que la somme et le produit concerne les deux mêmes nombres (on lit `a + b` et `a * b`...)

À partir de maintenant nous allons parfois utiliser des variables. Si vous n'avez pas bien compris ce qui précède, relisez... ou demandez des compléments.

Questionner l'utilisateur

Dans ce qui précède, les données d'entrée (les nombres à ajouter et multiplier) doivent être modifiées dans le programme lui-même. Il est parfois souhaitable que l'utilisateur final n'ait pas besoin d'aller modifier le code source (le texte du programme). Dans ce cas, il faut que les données d'entrée lui soient demandées par le programme lui-même.

Voici un exemple, que vous pouvez directement exécuter :

Dans le programme qui précède, nous utilisons la variable n pour désigner le nom que l'utilisateur aura entré. La possibilité d'entrer son nom est donnée par la fonction `input` qui affiche le texte qu'on lui donne en paramètre, puis attend que l'utilisateur entre quelque chose. La fonction `input` *renvoie* le texte qui sera entré par l'utilisateur et en écrivant `n = ` nous appelons ce texte n.

Modifiez le programme précédent :

1. afin qu'il affiche un autre message
2. afin qu'il demande l'âge en plus du nom

3. afin qu'il affiche finalement : `Bonjour <votre nom>, ça fait quoi d'avoir <votre âge> ans ?`

Si nous voulons utiliser ce que nous venons de voir pour demander à l'utilisateur qu'il entre 2 nombres à additionner, nous allons au devant de quelques surprises. Essayez le programme qui suit et entrez des nombres entiers, par exemple 123 et 65.

Pour Python, un objet de type nombre entier, comme 123 est différent d'une chaîne de caractères qui contiendrait le caractère 1, le caractère 2 et le caractère 3 (et qu'on noterait `'123'` au lieu de 123).

Pour faire fonctionner le programme qui précède, il faut indiquer que a et b sont des nombres et non des chaînes de caractères. Ce qui précède est d'autant plus déroutant que Python affiche quelque chose (12365 si vous avez entré les nombres 123 et 65). On comprend que pour Python, faire l'opération + entre des chaînes de caractères consister à les mettre bout à bout (l'opération s'appelle *concaténer*), alors que pour des nombres, l'opération + consiste bien sûr à ajouter.

Convertir une chaîne de caractères (comme `'123'`) en nombre s'appelle faire de la *conversion de type* ou du *transtypage*. La conversion en entier est réalisée en Python avec la fonction `int()`. Notons aussi la fonction `type()` qui peut indiquer le type d'un objet (`str` pour une chaîne de caractères et `int` pour un nombre entier).

Lisez attentivement le programme qui suit et exécutez le pour comprendre ce qu'il fait :

Le programme précédent contient beaucoup de difficultés. Assurez-vous de l'avoir bien compris. Par exemple il doit être clair que la ligne : `a = int(a)` est décomposable en deux parties qui signifient :

- obtenons le nombre entier qui est la conversion de a : ceci est obtenu en écrivant `… int(a)`
- puis, appelons ce nouvel objet a : ceci est obtenu en écrivant `a = …`

Modifiez le code qui suit (en utilisant `int`), pour que le programme affiche la somme des nombres, et non pas leur concaténation (il doit afficher 188 si on entre 123 et 65) :

Cherchez un peu avant de lire la solution...

Pour résoudre le problème qui précède on peut écrire par exemple !

```
a = input("Entrez le premier nombre")
```

```
a = int(a)
```

On peut aussi faire tout ça en une seule ligne :

```
a = int(input("Entrez le premier nombre"))
```

Et il faut faire la même chose pour b aussi...

Conversion d'unités

À présent, nous allons faire un peu d'arithmétique. Le programme suivant permet de convertir des degrés Celcius en Kelvins.

Les degrés Celcius et les Kelvins sont des unités de température. On peut passer d'une unité à l'autre par de simples opérations arithmétiques, de la même manière qu'on transforme des pouces en centimètres en multipliant par 2.54 (environ) ou des Dollars en Euros en divisant par 1.3.

Testez le programme :

Modifiez le programme qui précède de manière à ce que la conversion soit faite en degrés Fahrenheit plutôt qu'en Kelvins.

Pour convertir des degrés Celcius en degrés Fahrenheit il faut multiplier la valeur en degrés Celcius par 1.8 et ajouter 32. Vous devez trouver que l'eau gèle à 32 degrés Fahrenheit et bout à 212 degrés Fahrenheit

Il est possible que les résultats des conversions vous surprennent. En convertissant des degrés Celcius en Kelvin, par exemple, si vous entrez 65, vous obtiendrez peut être -208.14999999999998. Il s'agit d'un erreur d'arrondi très courante lorsqu'on fait des calculs informatiques sur des nombres à virgule. Même si cette erreur est toujours présente, une version moderne de Python installée sur votre machine personnelle vous proposerait un affichage un peu moins étonnant. Quoi qu'il en soit, ce type d'erreur ne nous dérangera pas dans la suite, mais il est très important de savoir que ça existe. Vous pourrez vous documenter sur ce sujet ici : Interstices : Pourquoi mon ordinateur calcule-t-il faux ?

Aller vers Initiation interactive à Python 3 : Les tests