Arithmétique, variables….

Présentation

Le langage de programmation Python permet d'interagir avec la machine à l'aide d'un programme appelé interprète Python. On peut l'utiliser de deux façons différentes.

- La première méthode consiste en un dialogue avec l'interprète. C'est le mode interactif.
- La seconde consiste à écrire un programme dans un fichier source puis à le faire exécuter par l'interprète. C'est le mode programme.

Mode interactif

Le mode interactif de l'interprète Python se présente comme une calculatrice.
Les trois chevrons >>> constituent l'invite de commandes de Python, qui indique qu'il attend des instructions.

Par exemple si on tape 1+2 puis la touche « Entrée » , l'interprète Python calcule et affiche le résultat.

```
>>> 1+2
3
>>>
```

Les chevrons apparaissent de nouveau. L'interprète est prêt à recevoir de nouvelles instructions.

Arithmétique

On peut saisir des combinaisons arbitraires d'opérations arithmétiques, en utilisant notamment les quatre opérations les plus communes.

```
>>> 2 + 5 * (10 - 1 / 2)
49,5
```

Dans l'exemple ci-dessus, on a utilisé des espaces pour améliorer la lisibilité. Ces espaces sont purement décoratifs et ne modifient pas la façon dont l'expression est calculée.

Attention, les nombres « à virgule » s'écrivent à l'anglo-saxonne avec un point, et non avec une virgule qui a un autre sens en Python. En essayant d'appliquer des opérations arithmétiques à des nombres écrits avec des virgules on obtient toute une panoplie de comportements inattendus.

```
>>> 1,2 + 3,4
1, 5, 4

>>> 1,2 * 3
1, 6
```

Variables

Les résultats calculés peuvent être mémorisés par l'interprète, afin d'être réutilisés plus tard dans d'autres calculs.

```
>>> a = 1 + 1
```

La notation *a* = permet de donner un nom à la valeur à mémoriser. L'interprète calcule le résultat de 1+1 et le mémorise dans la variable *a*. Aucun résultat n'est affiché. On accède à la valeur mémorisée en utilisant le nom a.

```
>>> a
2
```

Plus généralement, la variable peut être réutilisée dans la suite des calculs.

```
>>> a * (1 + a)
6
```

On peut donner une nouvelle valeur à la variable *a* avec une nouvelle affectation. Cette valeur remplace la précédente.

```
>>> a = 3
>>> a * (1 + a)
12
```

Le calcul de la nouvelle valeur de *a* peut utiliser la valeur courante de *a.*

```
>>> a = a + 1
>>> a
4
```

Un nom de variable peut être formé de plusieurs caractères (lettres, chiffres et souligné). Il est recommandé de ne pas utiliser de caractères accentués et l'usage veut que l'on se limite aux caractères minuscules.

```
>>> cube = a * a * a
>>> ma_variable = 42
>>> ma_variable2 = 201
```

Mode programme

Le mode programme de Python consiste à écrire une suite d'instructions dans un fichier et à les faire exécuter par l'interprète Python. Cette suite d'instructions s'appelle un programme, ou encore un code source.

Affichage

En mode programme, les résultats des expressions calculées ne sont plus affichés à l'écran. Il faut utiliser pour ceci une instruction explicite d'affichage. En Python, elle s'appelle *print*.

Par exemple, écrire l'instruction suivante.

```
print(3)
```

On peut alors faire exécuter ce programme par l'interprète Python, ce qui affiche 3 à l'écran.

```
print(1+3)
```

calcule la valeur de l'expression 1+3 puis affiche 4 à l'écran .

Affichage de textes

L'instruction print n'est pas limitée à l'affichage de nombres. On peut lui donner un message à afficher, écrit entre guillemets.
L'instruction

```
print("Salut tout le monde !")
```

affiche le message Salut tout le monde ! à l'écran.

Il est important de comprendre que le contenu d'une chaîne est arbitraire et n'est pas interprété par Python.

```
print("1+3")
```

affiche simplement 1+3 à l'écran.

Séquence d'instructions

Un programme est généralement constitué de plusieurs instructions.
Chaque instruction est écrite sur une ligne. L'interprète Python les exécute l'une après l'autre, dans l'ordre du fichier.

```
a = 34
b = 21 + a
print(a)
print(b)
```

affiche deux entiers à l'écran, sur deux lignes successives.

```
34
55
```

Si on souhaite afficher les deux entiers sur une même ligne, on peut remplacer les deux instructions *print* par une seule, en séparant les deux éléments à afficher par une virgule.

```
print(a, b)
```

```
34 55
```

Plus généralement, on peut utiliser *print* avec un nombre arbitraire d'éléments, qui peuvent être des nombres ou des chaînes de caractères.

```
a = 34
b = 55
print("la somme de", a, "et de", b, "vaut", a+b)
```

L'exécution de ce programme affiche le message suivant.

```
la somme de 34 et de 55 vaut 89
```

Interagir avec l'utilisateur

Pour permettre l'interaction du programme avec l'utilisateur, il faut procéder en deux temps : d'abord utiliser l'instruction *input* pour récupérer des caractères tapés au clavier par l'utilisateur, puis utiliser l'instruction *int* pour convertir cette chaîne de caractères en un nombre entier.

```
s = input()
a = int(s)
print("le nombre suivant est ", a + 1)
```

L'instruction input interrompt l'exécution du programme et attend que l'utilisateur saisisse des caractères au clavier. La saisie se termine lorsqu'il appuie sur la touche *Entrée* .
Si on exécute ce programme et qu'on saisit successivement les caractères 2, 7 et la touche *Entrée* , le programme affiche la ligne suivante.

```
le nombre suivant est 28
```

Deux choses à signaler

print sépare l'affichage des objets avec une seule espace.
print termine l'affichage par un saut de ligne à la fin.

On peut changer ce comportement

`sep` : pour *separator*, permet de choisir le séparateur entre les éléments
par défaut : `sep=" "` un espace
`end` : pour la fin, permet de choisir la fin de l'affichage
par défaut : `end="\n"` un saut de ligne

```
print("mot", 12345, "colle", 7, sep="", end="")
print("fin")

mot12345colle7fin
```

Applications

- Deviner l'affichage produit par ce programme

```
print("abc", 123, "567", 8, sep="...", end="===")
print(42, 1337, sep=" T ")
```

- Modifier ce script suivant le plus simplement possible

```
print(0, 1, 1, 2, 3, 5, 8, 13, 21)
```

pour obtenir `0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21`

Ecrire un programme

```
# calcul de l'âge
saisie = input("Entrez votre année de naissance : ")
annee = int(saisie)
age = 2048 - annee   # on calcule l'âge par soustraction
print("Vous aurez", age, "ans en 2048."
```

un commentaire commence par un symbole # et se poursuit jusqu'à la fin de la ligne.

Analysez et testez le programme précédent.

Il est possible de remplacer les deux instructions :

```
s = input("Entrer un nombre :")
a = int(s)
```

en une seule de la manière suivante.

```
a = int(input("Entrer un nombre :"))
```