Chapitre 1 : programmation débutant

Audras

Lycée Saint-Just

année 2018-2019 Isn

Définition

Une variable est une étiquette qui permet de stocker et de retrouver une ou des données dans la mémoire de l'ordinateur.
La variable peut être de type nombre entier relatif (int), nombre décimal (float), chaîne de caractères (str) ou autre ...

```
>>> monText = "bonjour"
```
est de type str,
```
>>> i=1
```
est de type int,
```
>>> x=2.456
```
est de type float.

On peut affecter directement une valeur à une variable ou indirectement via d'autres variables.

Propriété

Programmer en Python 3 : on travaille dans une console. On appuie sur la flêche haut pour rappeller les instructions écrites auparavant.

- Affecter une valeur à une variable : `>>> monText = "bonjour"` ou `>>> i=1` ou `>>> x=2.456`
- on écrit la variable pour connaître son contenu :

```
>>> monText
'bonjour'
```

```
>>> i
1
```

```
>>> x
2.456
```

- Convertir un texte contenant un nombre, en un nombre décimal.

```
>>> float("3.14")
3.14
```

et réciproquement

```
>>> str(3.14)
'3.14'
```

- Convertir un nombre décimal en un nombre entier, il est coupé après la virgule.

```
>>> int(3.14)
3
```

Définition

'Si (condition) Alors {Actions} ' permet de ne réaliser une action ou un groupe d'actions que si une condition est vraie. Si la condition est fausse les lignes suivantes du code sont lues et exécutées.
Pour exécuter une action si la condition est vraie et une autre si elle est fausse on ajoute le terme 'Sinon' :
'Si (condition) Alors {Actions} Sinon {Actions}'
Par exemple : Si $(x \geqslant 0)$ Alors $\{y = \sqrt{x}\}$ permet d'éviter une erreur.
Si $(x \geqslant 0)$ Alors $\{y = \sqrt{x}\}$ Sinon {texte = 'x doit être positif !'}
Attention pour tester l'égalité on utilise $==$
On peut utiliser and et or pour plusieurs conditions. Par exemple :
if $(x \geqslant 0$ and $y \geqslant 0)$: print("(x, y) en haut à droite")

Propriété

- Pour utiliser les fonctions mathématiques on utilise le module math :

```
>>> import math
```

-
```
>>> if (x >= 0) :
...     y=math.sqrt(x)
```

-
```
>>> if (x == 0) :
...     text = "on ne peut pas diviser par 0"
... else :
...     y=1/x
```

-
```
>>> if (x>0): print("nombre positif")
... elif (x==0): print("nombre nul")
... else: print("nombre négatif")
```

math.sqrt(x) permet de calculer la racine carrée de x car 'sqrt' est l'abréviation de 'square root', 'root' signifiant racine et 'square' carré.
Remarque : les blocs débutent avec : et sont délimités par leur indentation. Pour ajouter une indentation on fait Tab et pour l'enlever Maj+Tab. Echap pour revenir à une ligne vide.

Définition

Répéter n fois un bloc d'actions :

for i in range(n) : actions

'Pour (la variable i allant de 1 à n) faire { les actions ... }' :

for i in range(1,n+1) : actions

ou 'Pour (la variable i allant de 1 à n avec un pas) faire { les actions ... }' :

for i in range(1,n+1,pas) : actions

Propriété

for i in range(0,n) : actions

i est un nom de variable, on peut le remplacer par n'importe quel nom de variable, c'est le compteur de la boucle, i prend les valeurs de 0 à n−1.

in range(0,n) permet de fixer le nombre de répétition à n

par exemple, pour écrire la table de 7 :

```
>>> tableDe7 = ""
>>> for i in range(1,10) :
...     tableDe7 = tableDe7 + "7 x " + str(i) + " = " + str(7*i) + "\n"
```

```
>>> print(tableDe7)
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
```

+ permet d'accoler des textes.

str(i) convertit un nombre en texte.

\n permet de provoquer un saut de ligne.

print(tableDe7) donne un meilleur affichage de la variable.

Définition

Répéter Tant qu'une condition est vraie :
'Tant que (condition) faire { les actions ... }'
Si dans le bloc d'actions rien ne vient rendre la condition fausse alors la boucle se répète à l'infini et bloque l'ordinateur. Il faut toujours vérifier que la condition sera fausse au bout d'un temps fini.

Propriété

while (condition) : actions

par exemple, déterminer la plus petite puissance de 2 qui dépasse 10000.

```
>>> indice = 1
>>> valeur = 2
>>> while (valeur < 10000) :
...     valeur = valeur*2
...     indice = indice + 1
...
>>> print(indice,valeur)
14 16384
```

Pour vérifier on tape 2**n pour obtenir 2^n.

```
>>> 2**13
8192
```

```
>>> 2**14
16384
```

Définition

Les fonctions mathématiques utilisent un ou plusieurs antécédents pour calculer une ou plusieurs images. Il y a par exemple les fonctions linéaires, affines, carrée, inverse. Il y a l'algorithme de la division euclidienne qui prend comme antécédents deux entiers et qui retourne comme images le quotient et le reste.
En informatique il y a des fonctions comme en mathématiques, on dit qu'elles sont pures, et d'autres fonctions qui éxécutent des actions mais qui peuvent ne pas retourner d'image.
'fonction maFonction () {actions}'
'fonction maFonction (arguments) {actions}'
'fonction maFonction (arguments) {actions
retourne image}'

Propriété

- sans argument ni retour, les variables sont extérieures à la fonction.

```
>>> def carreX() :
...     global x
...     print("le carré de x est : " + str(x*x))
```

Pour utiliser la fonction :

```
>>> x=4
```

```
>>> carreX()
le carré de x est : 16
```

- fonction pure :

```
>>> def carre(x) :
...     affiche = "le carré de " + str(x) + " est : " + str(x*x)
...     return affiche
```

```
>>> carre(5)
'le carré de 5 est : 25'
```

global indique que x est une variable globale à tous le programme.
affiche n'est visible et n'est utilisable qu'à l'intérieure de carre() :

```
>>> affiche
Traceback (most recent call last):
NameError: name 'affiche' is not defined
>>>
```

Exemple

Les fonctions récursives.
Quand une fonction est donnée par une relation de récurrence, on peut la programmer de façon récursive, en se retournant elle même avec un indice inférieur.
La fonction factorielle : $n! = n(n-1)(n-2)\dots 2 \times 1$ ou
$n! = n \times (n-1)!$ et $1! = 1$

```
1 def facto(n):
2     if n==1: return 1
3     return n*facto(n-1)
```

Les expressions sont stockées en mémoires avant d'être évaluées. Pour facto(3) , 3*facto(2) est gardée en mémoire, puis pour facto(2), 2*facto(1) puis facto(1) retourne 1 et ensuite les expressions gardées sont évaluées : facto(2)=2*facto(1) retourne 2*1=2 et facto(3)=3*facto(2) retourne 3*2=6.

Définition

En Python les fonctions mathématiques prédéfinies commence par math. car elles sont dans ce module. Il y a la racine carrée, le nombre pi, ...
La fonction pour arrondir un nombre au plus proche avec n décimales est dans la base du langage.
Pour obtenir un nombre au hasard on utilise le module random. On peut avoir un nombre décimal entre 0 et 1 exclu avec random() et un nombre entier entre 1 et n inclus avec randint(1,n).

Propriété

```
>>> import math
>>> math.sqrt(2)
1.4142135623730951
>>> math.pi
3.141592653589793
```

```
>>> round(2.53)
3
>>> round(2.53,1)
2.5
```

```
>>> import random
>>> random.random()
0.13175707783155832
>>> random.randint(1,6)
5
```

```
>>> 3**3
27
>>> 2**(-1)
0.5
```

Définition

Déboguer signifie corriger les erreurs d'un programme. Pour connaître le contenu des variables locales à une fonction ou à une boucle, il existe une instruction qui affiche dans la console du texte et des variables.
En Python 3 c'est print()

```
>>> for i in range(1,3) :
...     print("i contient ",i)
...
i contient  1
i contient  2
```