Cours Programmation Python

Issam ben othmen

2020-2021

Partie 1: Programmer en Python

- Chapitre 1 : La calculatrice Python
- Chapitre 2 : Contrôle de flux d'instructions

Issam ben othmen

2020-2021

Chapitre 2 : La calculatrice Python

Issam ben othmen

2020-2021

- ✓ Une instruction est un ensemble de caractères permettant au développeur de déterminer une action à mener par son algorithme. Cette action peut être l'affectation d'une valeur à une variable, l'exécution d'une fonction, la déclaration d'une classe, l'écriture d'une condition.

✓ Une ligne de code =une instruction

- ✓ En python, une ligne de code permet d'écrire une instruction. Elle commence à gauche de l'écran et se termine tout simplement par un passage à la ligne

```
In [1]: print("Hello World!")
Hello World!
```

Mots-clés

- Python contient peu de mots-clés, 33 pour être précis.
- Ses mots-clés sont des éléments permettant de structurer les algorithmes.
- Ils ont chacun une signification particulière qui ne peut absolument pas être modifiée par le développeur.
- Très peu de mots clés ➔ Liberté aux développeurs

and	as	assert	break	class	continue	def	elif	else	None
except	finally	for	from	global	if	import	is	lambda	True
nonlocal	not	or	pass	raise	return	try	with	yield	False

- **None** : singleton qui représente l'élément **vide.**

- **True** et **False** sont les deux uniques ***instances*** booléennes qui représentent respectivement **vrai** et **faux**.

Les types de données entiers

- Le type d'un objet Python détermine de quelle sorte d'objet il s'agit:
- La fonction **type()** fournit le type d'une valeur.
- Python offre deux types entiers standards**: int et bool**

Le type int

- Les entiers littéraux sont représentés en décimal par défaut, mais on peut aussi utiliser les bases suivantes:

```
In [43]: 2013
Out[43]: 2013
```

Décimal (base 10) par défaut

```
In [44]: 0b11111011101
Out[44]: 2013
```

Binaire (base 2) avec le préfixe 0b

```
In [45]: 0o3735
Out[45]: 2013
```

Octal (base 8) avec le préfixe 0o

```
In [46]: 0x7dd
Out[46]: 2013
```

Hexadécimal (base 16) avec le préfixe 0x

Représentation binaire, octale et hexadécimale de l'entier 2013

```
In [47]: bin(2013), oct(2013), hex(2013)
Out[47]: ('0b11111011101', '0o3735', '0x7dd')
```

Le type int : opérations arithmétiques

```
In [48]:    20+3
Out[48]: 23

In [49]: 20-3
Out[49]: 17

In [50]: 20*3
Out[50]: 60

In [51]: 20**3
Out[51]: 8000

In [52]: 20/3
Out[52]: 6.666666666666667

In [53]: 20//3
Out[53]: 6

In [54]: 20%3
Out[54]: 2

In [55]: divmod(20,3)
Out[55]: (6, 2)

In [56]: abs(3-20)
Out[56]: 17
```

Division entière

Modulo

Division entière et modulo

Valeur absolue

Quelle est la différence entre / et // ?

Le type bool

- En Python les représentations littérales ses valeurs booléennes sont notées False et True. Les opérations de base sont noté respectivement not, and et or.

a	not(a)
False	*True*
True	*False*

Opérateur unaire not

a	b	a or b	a and b
False	*False*	*False*	*False*
False	*True*	*True*	*False*
True	*False*	*True*	*False*
True	*True*	*True*	*True*

Opérateurs binaires or et and

```
In [59]: (3==3) or (9 > 24)
Out[59]: True

In [60]: (9 >24) and (3==3)
Out[60]: False
```

Transtypage

```
In [62]: 0 or 56
Out[62]: 56

In [63]: b=0

In [64]: b and 3>2
Out[64]: 0
```

Exercice

Les opérateurs de comparaison renvoient « True » ou « False ».

```
a = 5
b = 6

print (a == b)  #Est ce que a est égal à b
print (a != b)  #Est ce que a est différent de b
print (a >= b)  #Est ce que a est supérieur ou égal à b
print (a <= b)  #Est ce que a est inférieur ou égal à b
print (a > b)   #Est ce que a est supérieur à b
print (a < b)   #Est ce que a est inférieur à b
print ("")
print (True and True)   #Opérateur "and"
print (True and False)
print (False and False)
print ("")
print (True or True)    #Opérateur "or"
print (True or False)
print (False or False)
print ("")
print not True          #Opérateur "not"
print not False
```

Les types de données flottants : Le type float

- Un **float** est noté avec un point décimal (jamais avec un virgule), ou en notation exponentielle, avec une e ou un E symbolisant le « puissance 10 » suivi des chiffres de l'exposant

```
In [75]: 2.718, .02, 3E5, -1.6e-13, 6.023E23
Out[75]: (2.718, 0.02, 300000.0, -1.6e-13, 6.023e+23)

In [76]: 3E13
Out[76]: 30000000000000.0

In [77]: 2e-1
Out[77]: 0.2
```

- L'import du module standard ***math*** autorise toutes les opérations mathématiques usuelles.

```
In [70]: import math

In [71]: math.sin(math.pi/4)
Out[71]: 0.7071067811865476

In [72]: math.degrees(math.pi)
Out[72]: 180.0

In [73]: math.factorial(9)
Out[73]: 362880

In [74]: math.log(1024, 2)
Out[74]: 10.0
```

Les types de données flottants : Le type Complex

```
In [78]: 1j
Out[78]: 1j

In [79]: (2+3j)*(4-7j)
Out[79]: (29-2j)

In [80]: (9+5j).real
Out[80]: 9.0

In [81]: (9+5j).imag
    ...: (abs(3+4j))
Out[81]: 5.0
```

Un module mathématique spécifique ***cmath*** leur est réservé

```
In [82]: import cmath

In [83]: cmath.phase(-1+0j)
Out[83]: 3.141592653589793

In [84]: cmath.polar(3+4j)
Out[84]: (5.0, 0.9272952180016122)

In [85]: cmath.sqrt(3+4j)
Out[85]: (2+1j)
```

- Une variable est un identificateur associé à une valeur.
  ➔ *C'est une référence d'objet*

Affectation ou assignation :

- On affecte une valeur à une variable en utilisant le signe d'égalité (=).

➔ *L'affectation n'a rien à voir avec l'égalité en math!*

```
In [1]: 2*7-8
Out[1]: 6

In [2]: _+4
Out[2]: 10
```

'_', utilisable en mode interactif, contient le *le résultat de la dernière opération*

```
In [3]: v=5

In [4]: v +=2

In [5]: v
Out[5]: 7

In [6]: c=d=12

In [7]: c,d
Out[7]: (12, 12)

In [8]: e, f=4.8, 7.5

In [9]: e,f
Out[9]: (4.8, 7.5)


In [10]: a=-4

In [11]: a, b = a+2, a*2

In [12]: a,b
Out[12]: (-2, -8)
```

Affectation simple

Affectation augmentée. v=v+2 si v est déjà référencé

d reçoit 12, puis c reçoit d. Ils référencent la mm donné

Un tuple

Affectation //

Toute les expressions sont évaluées avant la premier affectation.

Les chaînes de caractères:

le type de donnée non modifiable ***Str*** représente une séquence de caractère unicode.

- Les **chaines de caractères** sont des valeurs textuelles (espaces, symboles, alphanumériques,…) entourée par des *guillemets* **simples** ou **doubles**, ou par une série de **trois guillemets simples ou doubles**.

Les séquences d'échappement

A l'intérieur d'une chaîne, le caractère antislash (\) permet de donner une signification spéciale à certaines séquences de caractères.

Séquence	Signification
\	saut de ligne ignoré (placé en fin de ligne)
\\	antislash
\'	**apostrophe**
\''	guillemet
\a	Sonnerie (bip)
\b	Retour arrière
\n	**Saut de la page**
\f	Saut de la ligne
\r	Retour en début de ligne
\t	**Tabulation horizontal**
\v	Tabulation verticale
\N{nom}	**Caractère sous forme de code unicode nommé**

Instruction sur un ou plusieurs lignes

Séquence	Signification
\uhhhh	Caractère sous forme de *code Unicode 16 bits* sur 4 chiffres Hexa
\Uhhhhhhhh	Caractère sous forme de *code Unicode 32 bits* sur 8 chiffres Hexa
\ooo	Caractère sous forme de *code octal* sur 3 chiffres octaux
\xhh	Caractère sous forme de *code Hexa* sur 2 chiffres

```
In [29]: "\N{pound sign}  \u00A3     \U000000A3"
Out[29]: '£  £     £'
```

```
In [30]:  "d \144 \x64"
Out[30]:  'd d d'

In [31]: r"d \144 \x64"
Out[31]: 'd \\144 \\x64'
```

La **notation r"..."** Désactive la signification spécial du caractère (\ )

Instruction sur un ou plusieurs lignes

```
print("Hello LSI1")
Hello LSI1

print('Hello LSI1')
Hello LSI1

print('Hello',\
'LSI1')
Hello LSI1

print(Hello LSI1)
File "<ipython-input-34-
c1537a502534>", line 1
    print(Hello LSI1)
                ^
SyntaxError: invalid syntax
```

```
print('premier',\
'seance',\
'cours',\
'python')
premier seance cours python
```

```
print('Hello\
LSI1')
HelloLSI1
```

espace

Instruction sur un ou plusieurs lignes

```
In [3]: print("Un \nDeux")
Un
Deux

In [4]: print("Un \n Deux")
Un
 Deux

In [5]: print("""A\tB\tC\tD\tE""")
A       B       C       D       E
```

Chaîne mutiligne

tabulation

Les Chaines de caractères : opérations

```
In [39]: len("LSI A B C D E et F")
Out[39]: 18
```

Longueur

```
In [40]: "ABCDE" + "FG"
Out[40]: 'ABCDEFG'
```

Concaténation

```
In [41]: "HELLO!" * 5
Out[41]: 'HELLO!HELLO!HELLO!HELLO!HELLO!'
```

Répitition

```
In [42]: "H" in "Hello"
Out[42]: True
```

Test d'appartenance

Les Chaines de caractères : opérations

On peut agir sur une chaîne en ***utilisant des fonctions (notion procédurale)*** communes à tous les types ***séquences ou conteneurs***, ou bien des ***méthodes (notion d'objet) spécifiques aux chaines***:

```
In [43]: len('LES ETUDIENTS')
Out[43]: 13
```

Syntaxe d'une fonction

```
In [44]: "abracadabra".upper()
Out[44]: 'ABRACADABRA'
```

Syntaxe d'une méthode

Méthodes de test de l'état d'une chaîne

```
In [45]: "Le petit GARCON".isupper()
Out[45]: False

In [46]: "Le petit GARCON".istitle()
Out[46]: False

In [47]: "Le Petit GARCON".istitle()
Out[47]: False

In [48]: "Le Petit Garçon".istitle()
Out[48]: True

In [49]: "Garçon".isalpha()
Out[49]: True

In [50]: "Neuf".isdigit()
Out[50]: False

In [51]: "676".isdigit()
Out[51]: True
```

Tout est en majuscule

Chaque mot commence par une Majuscule

Ne contient que des caractères alphabétiques

Ne contient que des caractères numériques

Méthodes de test de l'état d'une chaîne (suite)

```
In [52]: "Le Petit Garçon".startswith('Le')
Out[52]: True
```

Commence par .....

```
In [53]: "Le Petit Garçon".endswith('Garçon')
Out[53]: True

In [54]: "Le Petit Garçon".endswith('garçon')
Out[54]: False
```

..... Finit par

Méthodes retournant une nouvelle chaîne

```
In [55]: "Le Petit Garçon".lower()
Out[55]: 'le petit garçon'

In [56]: "Le Petit Garçon".upper()
Out[56]: 'LE PETIT GARÇON'

In [57]: "Le Petit GARCON".swapcase()
Out[57]: 'lE pETIT garcon'

In [58]: "Le Petit Garçon".center(10,'~')
Out[58]: 'Le Petit Garçon'

In [59]: "Le Petit Garçon".center(31,'~')
Out[59]: '~~~~~~~~~Le Petit Garçon~~~~~~~~~'

In [60]: "Le Petit Garçon".center(31,'*')
Out[60]: '********Le Petit Garçon********'

In [61]: "Le Petit Garçon".center(31,'b')
Out[61]: 'bbbbbbbbLe Petit Garçonbbbbbbbbb'

In [62]: "Le Petit Garçon".rjust(31,'^')
Out[62]: '^^^^^^^^^^^^^^^^^Le Petit Garçon'

In [63]: "Le Petit Garçon".ljust(31,'^')
Out[63]: 'Le Petit Garçon^^^^^^^^^^^^^^^^^'
```

Toute en miniscule

Toute en majuscule

Inverser la casse

Chaîne centrée

Chaîne justifier à droite

Chaîne justifier à gauche

Méthodes retournant une nouvelle chaîne (suite)

```
In [65]: "Le Petit Garçon".split()
Out[65]: ['Le', 'Petit', 'Garçon']

In [66]: "Le Petit Garçon".split('-')
Out[66]: ['Le Petit Garçon']

In [67]: "Le-Petit-Garçon-AHMED".split('-')
Out[67]: ['Le', 'Petit', 'Garçon', 'AHMED']

In [68]: "Le,Petit,Garçon,AHMED".split(',')
Out[68]: ['Le', 'Petit', 'Garçon', 'AHMED']

In [69]: "5566-4477-787-777-000".split('-')
Out[69]: ['5566', '4477', '787', '777', '000']
```

Découpe la chaîne suivant le séparateur (séquence d'espace par défaut ou – ou , …)

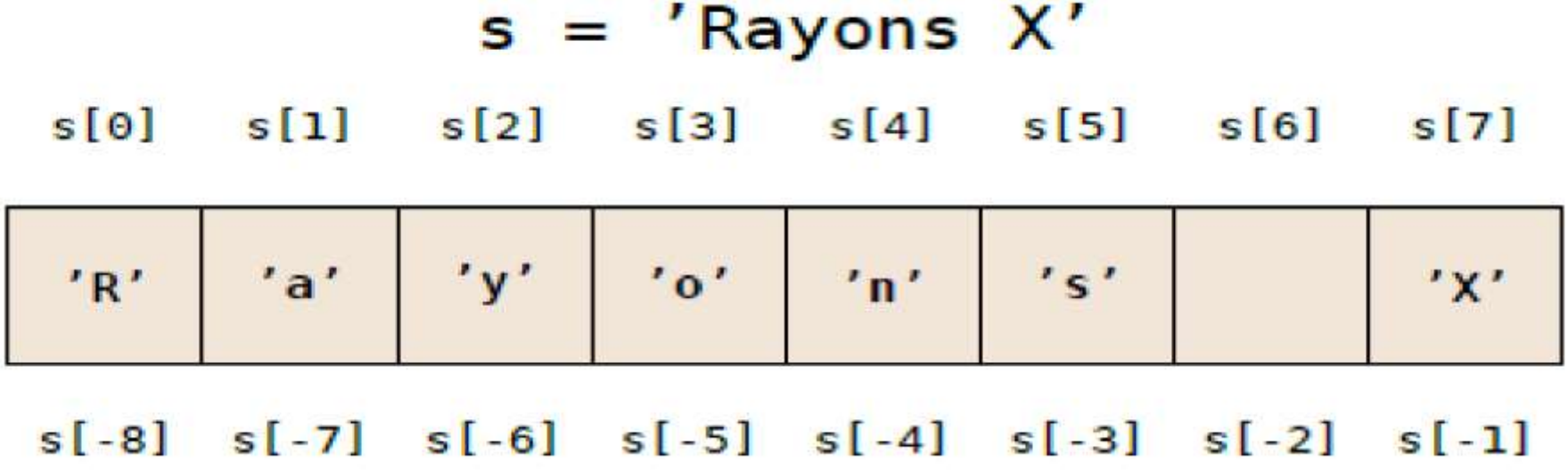
Méthodes retournant un indice

- find(sub[,start[, stop]]) : renvoi l'index de la chaîne sub dans la sous-chaîne start à stop, sinon renvoi -1.
- Rfind(): même travail en commençant par la fin.
- Index() et rindex() font de mm mais produisent une erreur si la chaine sub n'est pas trouvée.

```
In [73]: "Le Petit Garçon".index('P')
Out[73]: 3
```

Indexation simple

```
In [75]: s="Rayons X"

In [76]: s[0]
Out[76]: 'R'

In [77]: s[2]
Out[77]: 'y'

In [78]: s[-1]
Out[78]: 'X'

In [79]: s[-4]
Out[79]: 'n'
```



Extraction de tranches

```
In [84]: s="Rayons X"

In [85]: len(s)
Out[85]: 8

In [86]: s[1:4]
Out[86]: 'ayo'

In [87]: s[-3:]
Out[87]: 's X'

In [88]: s[:3]
Out[88]: 'Ray'

In [89]: s[3:]
Out[89]: 'ons X'

In [90]: s[::2]
Out[90]: 'Ryn '

In [91]: s[::-1]
Out[91]: 'X snoyaR'
```

de l'index 1 compris à 4 non compris

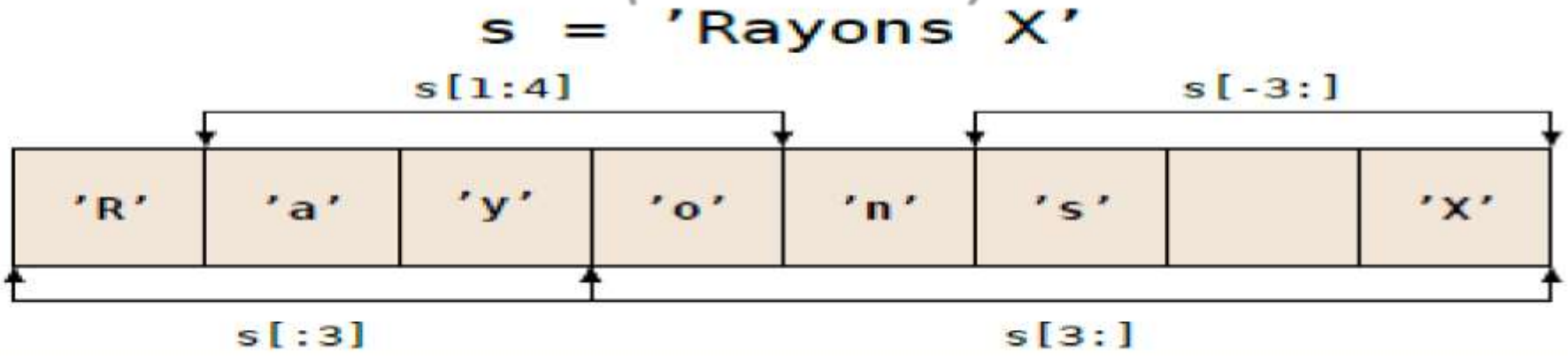
de l'index -3 compris à la fin

du début à l'index 3 non compris

de l'index 3 compris à la fin

du début à la fin, de 2 en 2

du début à la fin en pas inverse (retournement)



Principe

Utilisation du mot-clé désignant le type

> nouveau_type (objet)

Conversion en numérique

```
a = « 12 » # a est de type chaîne caractère
b = float(a) #b est de type float
```

N.B. Si la conversion n'est pas possible ex. float(« toto »), Python renvoie une erreur

Conversion en logique

```
a = bool(« TRUE ») # a est de type bool est contient la valeur True
a = bool(1) # renvoie True également
```

Conversion en chaîne de caractères

```
a = str(15) # a est de type chaîne et contient « 15 »
```

Les types binaires

Python propose deux types de données binaires: bytes (immutable) et bytearray (mutable)

La fonction bytes () peut convertir des objets en objets bytes ou créer l'objet bytes vide de la taille spécifiée.

La principale différence entre bytes () et byteArray () est que bytes () renvoie un objet qui ne peut pas être modifié, ce qui signifie qu'il renvoie un objet immuable.

En revanche, la fonction byteArray () renvoie l'objet qui peut être changé ou altéré.

https://docs.python.org/3.4/library/functions.html#bytearray

(a) Entrée

Lecture au clavier

(b) Sortie

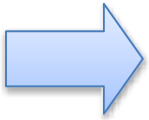
Ecriture sur l'écran

- Saisir une valeur
- L'instruction est **input()**, dans les parentheses, on écrit le message qui s'affiche.

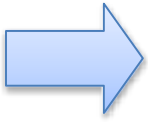
```
In [6]: a=input("Saisir a : "), print ("a vaut ",a)

Saisir a : 2
a vaut  2
```

- La variable saisie n'est pas considérée comme un nombre par Python mais comme une chaine de caractères

» a=input("Saisir a : ")
» a=a+1
» print ("a vaut ",a)



```
a=int(input("Saisir a : "))
a=a+1
print ("a vaut ",a)
```



```
a=float(input("Saisir a : "))
a=a+1
print ("a vaut ",a)
```

En mode calculatrice Python lit-évalue-affiche, mais la fonction print() reste indispensable aux affichages dans les scripts. Elle se charge d'afficher la représentation textuelle des informations qui lui sont données en paramètres.

```
In [16]: print('Hello World!')
Hello World!

In [17]: print()
```

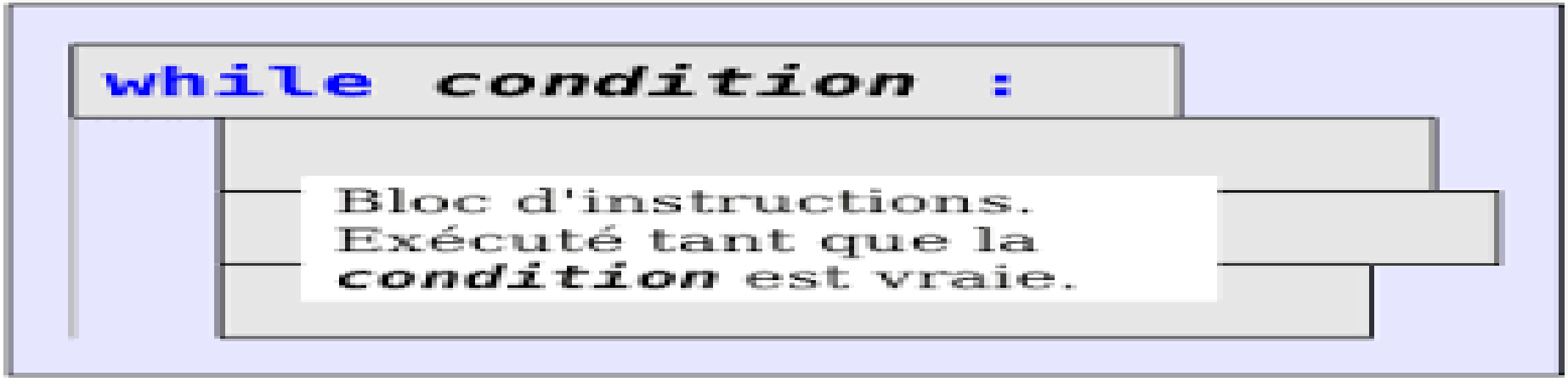
```
In [14]: a, b = 2, 5

In [15]: print('somme:', a + b, ';', a-b, 'est la différence ;', a*b, 'le produit et ', a / b, 'la division')
somme: 7 ; -3 est la différence ; 10 le produit et  0.4 la division
```

Une Instruction composée se compose :

- ***D'une ligne d'introduction terminée par le caractère « deux points »(:);***
- ***D'un bloc d'instructions indenté par rapport à la ligne d'introduction.***

Remarque :

Toutes les instructions au même niveau d'indentation appartiennent au même bloc d'instructions, jusqu'à ce que l'indentation redevienne inférieur à ce niveau.



Exemple Instructions composées

```
ph=float(input("entrer la valeur du PH ?"))
if ph < 7:
    print("Le potentiel hydrogène (ph) est inférieur à 7.")
    print("C'est un acide.")
if ph > 7:
    print("Le potentiel hydrogène (ph) est supérieur à 7.")
    print("C'est une base.")
if ph == 7:
    print("Le potentiel hydrogène (ph) est exactement 7.")
    print("La solution est neutre.")
```

Simple

Exemple Instructions composées

```
t = float(input("Quelle est la valeur du température (°C) ?"))
print("Température 't' en dgrés celsius")
if t <=0:
    print('Négative ou Nulle : risque de gel')
else:
    print('Positive')
    if t > 25:
        print('Plus de 25°C')
        print('Prévoir tee-shirt ou veste légére')
    elif t >18:
        print('Douce mais sans plus')
    else:
        print('Mais sortez couverts')
print('Evaluation terminée')
```

imbriquée

Indentation

- L'indentation est simplement un décalage vers la droite d'une ou de plusieurs lignes de code.
- C'est la présence des deux-points derrière une condition.
- Ce simple décalage va indiquer le fait que l'on rentre dans un bloc de code qui ne sera exécuté que si la condition est vraie.

```
if condition :
    instruction si vrai
nouvelle instruction non identité = fin du bloc conditionnel
```

- l'ensemble des lignes indentées constitue un bloc d'instructions.

Indentation

C	Python

Int a=0
For (int i=0; i<10;i++) {
a=a+i
}
	a=0
	for i in range(10):
	a +=i

- Le décalage avec des tabulations est primordial en Python.
- Pour python , pas d'accolades
- Les deux algorithmes suivants sont différents. Testez-les !

```
a=int(input("Saisir a: "))
if a==0:
    print("a=0")
print("C'est gagne!")
```

```
a=int(input("Saisir a: "))
if a==0:
    print("a=0")
    print("C'est gagne!")
```