

# Initiation à la programmation avec le langage de programmation Python

**Dr. Nourhène Ben Rabah**  
Contact : [nbenrabah@itescia.fr](mailto:nbenrabah@itescia.fr)

# Plan de la séance

1. Initiation à l'algorithmique (instruction, variable, programme, langage machine, interprétation, compilation)
2. Initiation à Python (interpréteur de commandes erreurs, variables, opérateurs et expressions, expressions booléennes, affectations, Entrées/sorties)



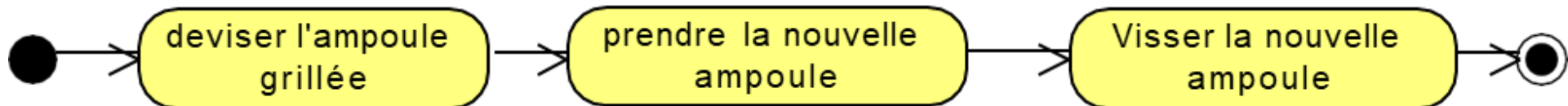


# Initiation à l'algorithmique

# Algorithme

---

- Un algorithme décrit, de façon non ambiguë l'ordonnancement des actions à effectuer pour traiter une fonctionnalité
- On implémente un algorithme à l'aide d'un langage de programmation
- Un algorithme est donc un processus



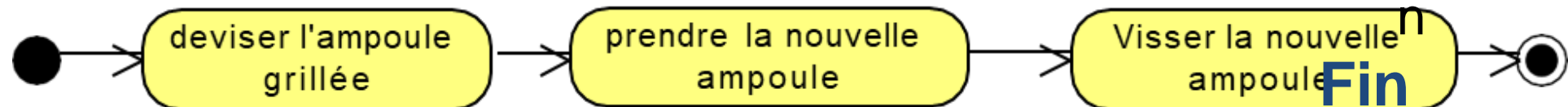
# Algorithme

---

- Un algorithme décrit, de façon non ambiguë l'ordonnancement des actions à effectuer (**instructions**) pour traiter une fonctionnalité
- On implémente un algorithme à l'aide d'un langage de programmation
- Un algorithme est donc un processus

## Début

Instruction  
1  
Instruction  
2  
Instruction  
3  
Instruction



# Algorithme

---

- Qu'est ce que veut dire « écrire un algorithme » ?
  1. Analyser et comprendre le problème : étude des données fournies et des résultats attendus.
  2. Résoudre le problème : C'est trouver les structures de données adaptées ainsi que l'enchaînement des actions (instructions) à réaliser pour passer des données aux résultats.

# Algorithme

---

- **Exemple :**

Problème : calculer le prix des baguettes

Entrée :

PU : prix d'une seule baguette

Sortie : Afficher le prix des baguettes achetées

## Algorithme



# Algorithme

---

- **Exemple :**

Problème : calculer le prix des baguettes

Entrée :

PU : prix d'une seule baguette

Sortie : Afficher le prix des baguettes achetées

## Variables

Nb : Entier

Prx, Mtt : Réel

## Début

0. Nb=0, Prx=1.8, Mtt=0

1. Afficher "Nombre de baguettes achetées svp ?"

2. Saisir Nb

3.  $Mtt = Prx * Nb$

4. Afficher "Montant :", Mtt

## Fin



# Structure d'un algorithme


---



[S1COURS02] Structure d'un algorithme



Visible pour les étudiants

Vidéo  4:10 © Tahar Gherbi

# Algorithme

---

- **Exemple :**

## Variables

Nb : Entier

Prx, Mtt : Réel

## Début

0. Nb=0, Prx=1.8, Mtt=0

1. Afficher "Nombre de baguettes achetées svp ?"

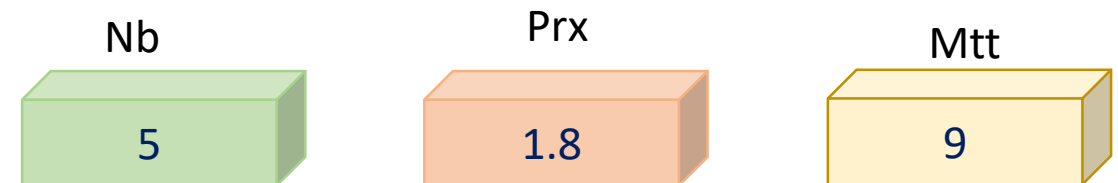
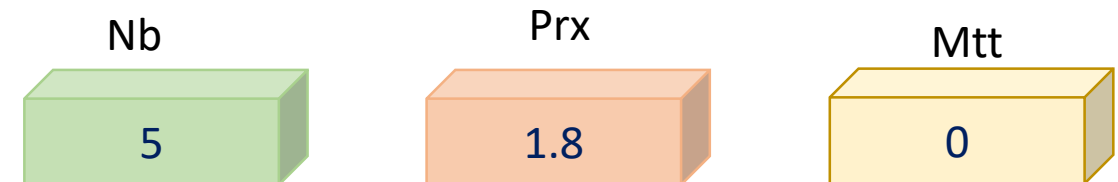
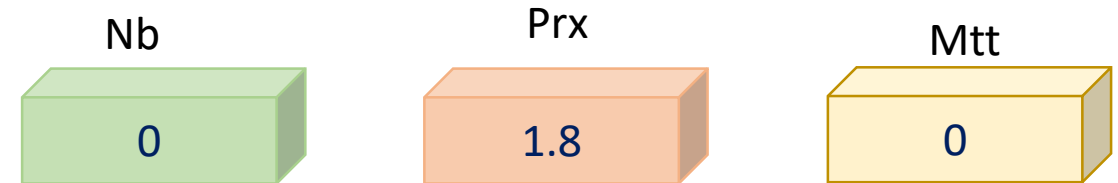
2. Saisir Nb

3.  $Mtt = Prx * Nb$

4. Afficher "Montant :", Mtt

## Fin

## Exécution manuelle



# Algorithme

---

- Comment exécuter un algorithme sur un ordinateur ?

1. Il faut traduire cet algorithme à l'aide d'un **langage de programmation** connu par l'ordinateur (Programme)

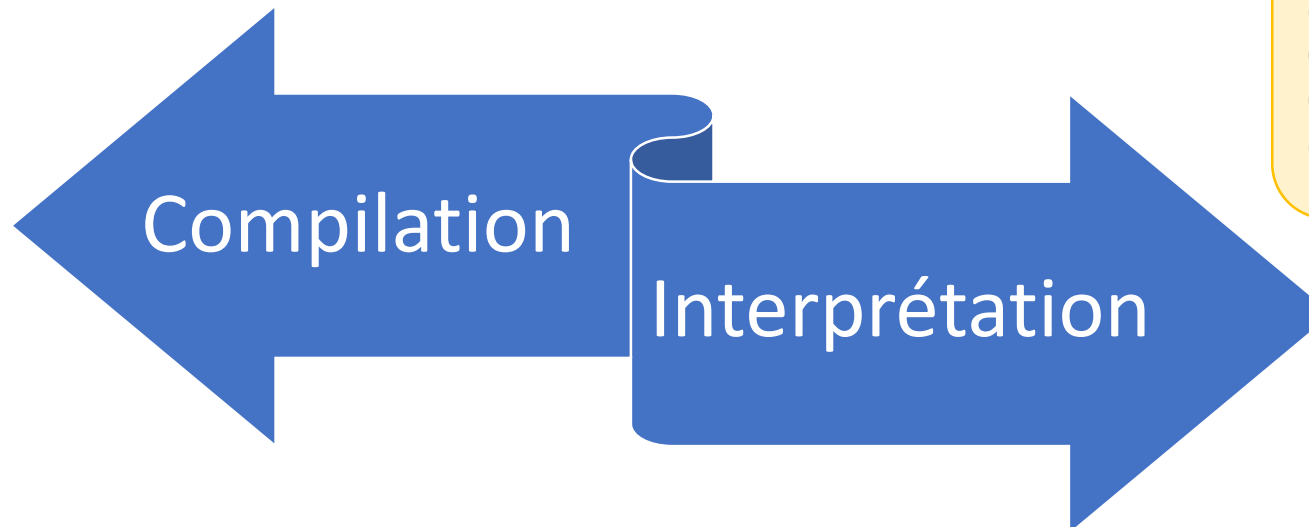


# Algorithme

---

- Comment exécuter un algorithme sur un ordinateur ?

2. Il faut traduire ce programme (code source) en langage machine (code binaire)



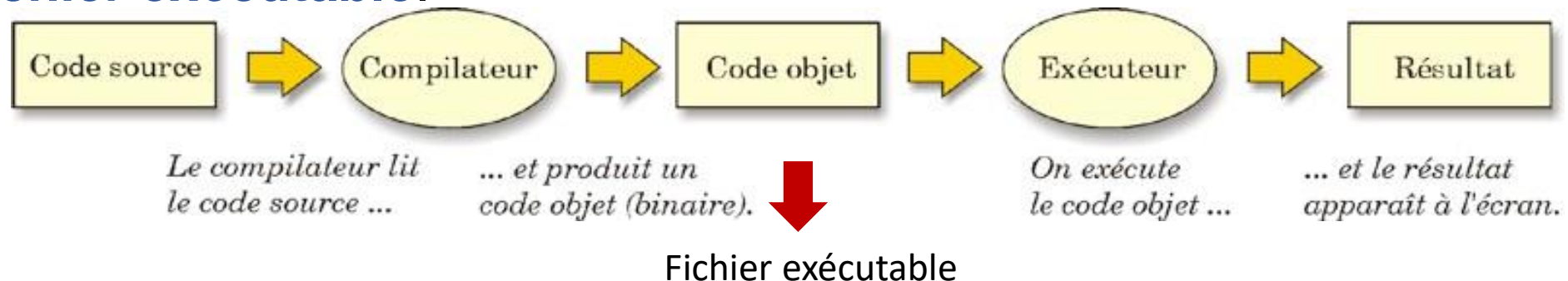
Le **langage binaire** est uniquement constitué de 0 et de 1. « 01000010011011110110111001101010 011011110111010101110010 », par exemple, signifie « Bonjour »

# Algorithme

---

## Compilation

- La **compilation** consiste à traduire la totalité du texte source en langage machine.
- Le logiciel compilateur lit toutes les lignes du programme source et produit une nouvelle suite de codes que l'on appelle **programme objet** (ou **code objet**).
- Celui-ci peut désormais être exécuté indépendamment du compilateur et être conservé tel quel dans un fichier, c'est **un fichier exécutable**.



# Algorithme

---

## Interprétation

- Dans cette technique en effet, **chaque ligne du programme source** analysé est traduite au fur et à mesure en quelques instructions du langage machine, qui sont ensuite directement exécutées.

Aucun programme objet n'est généré.



*L'interpréteur lit  
le code source ...*

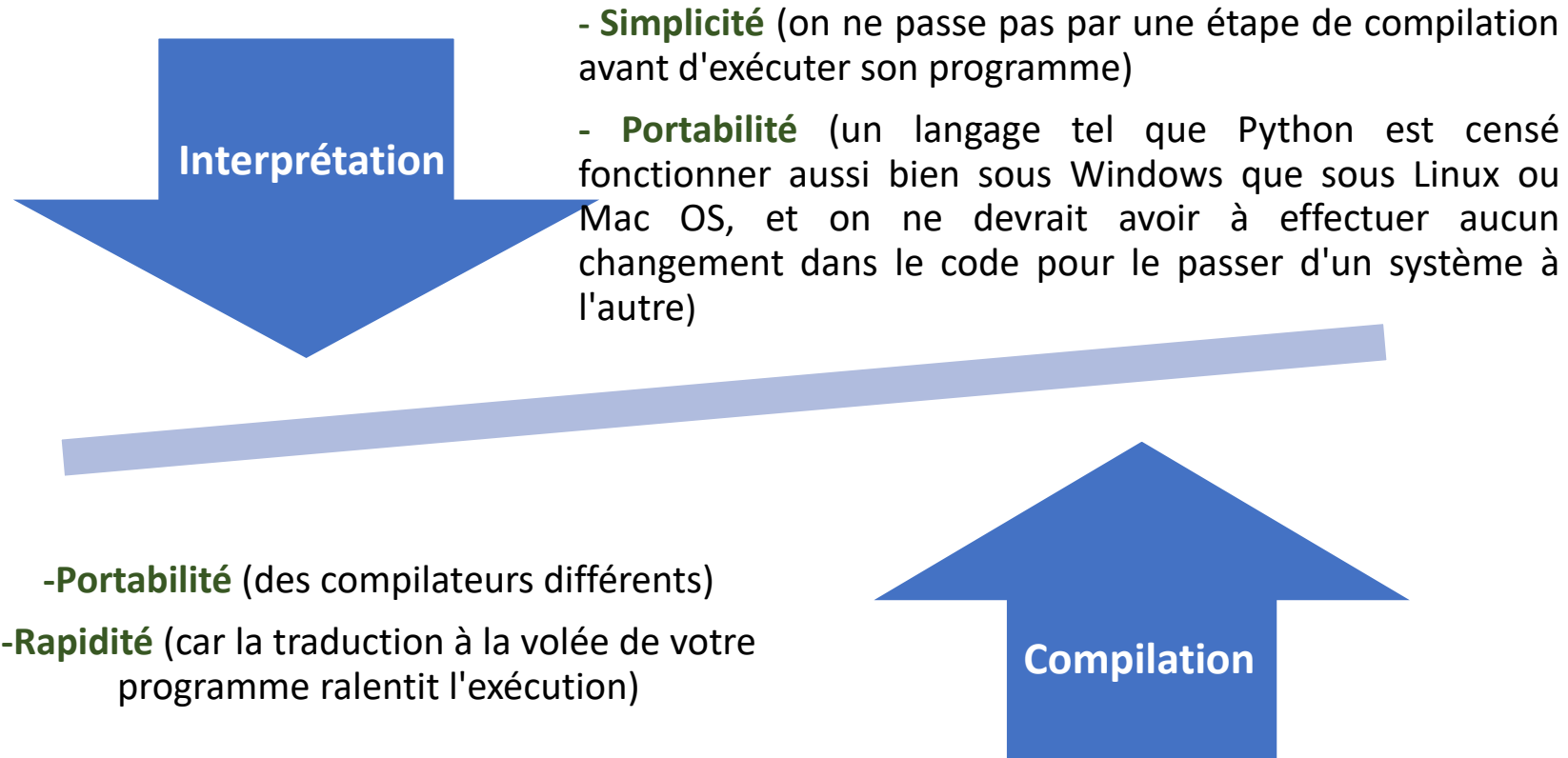
*... et le résultat  
apparaît sur l'écran.*

# Algorithme

---

## Compilation et Interprétation

Chacune de ces deux techniques a ses avantages et ses inconvénients :



# Algorithme

---

## Enoncé d'un problème

Analyse, compréhension

Résolution

## Algorithme

Codification

## Programme

Interprétation ou  
compilation

## Exécution par l'ordinateur

Pseudo code

Langage de  
programmation  
(code)

Langage machine



2

# Initiation à Python



# Python

---

- Python est un langage interprété
- Plusieurs versions Python 2.x (2.3, 2.5, 2.6...)
- Depuis le 13 février 2009, la version 3.0.1 est disponible
- Installez la dernière version de Python  
<https://www.python.org/downloads/>
- Sélectionnez le lien qui correspond à Système d'exploitation et votre processeur. Si vous avez un doute, téléchargez une version « x86 ».




# Lancer Python

---

- Démarrer>Tous les programmes>Python 3.9>Python (Command Line)

Ou

Démarrer > Exécuter >py

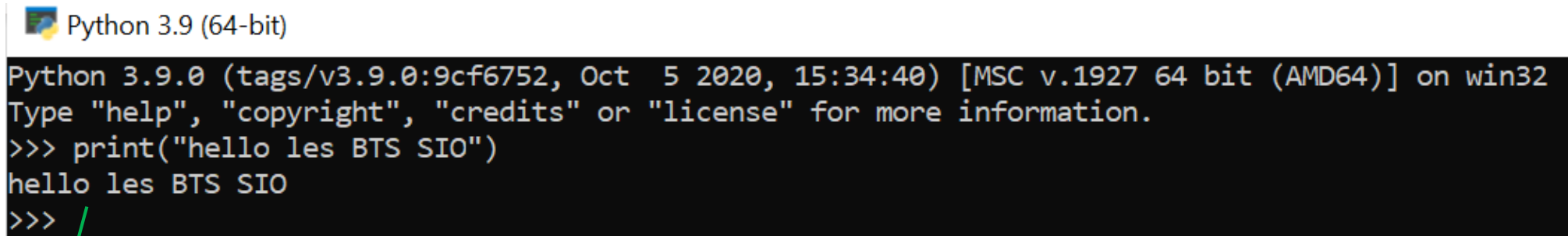
 C:\WINDOWS\py.exe

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```

# Ecrire votre premier programme sur l'interpréteur de commandes Python

---

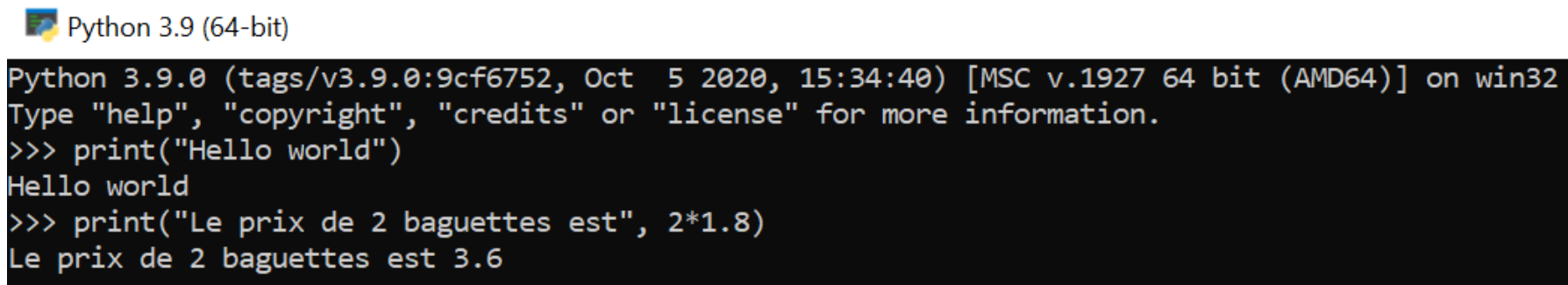
- Afficher le message « Hello les BTS SIO ! »



```
Python 3.9 (64-bit)
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello les BTS SIO")
hello les BTS SIO
>>>
```

Fonction d'affichage

- Afficher le prix de deux baguettes



```
Python 3.9 (64-bit)
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world")
Hello world
>>> print("Le prix de 2 baguettes est", 2*1.8)
Le prix de 2 baguettes est 3.6
>>>
```

# Les erreurs

---

## 1. L'erreur de syntaxe

- Python ne peut exécuter un programme que si sa **syntaxe** est parfaitement correcte.
- Dans le cas contraire, le processus s'arrête et vous obtenez un **message d'erreur**.
- Le terme syntaxe se réfère **aux règles** que les auteurs du langage ont établies pour la structure du programme.

## Exemples :

```
>>> aaa
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'aaa' is not defined
>>> "aaa"
'aaa'
```

```
>>> a="12
  File "<stdin>", line 1
    a="12
        ^
SyntaxError: EOL while scanning string literal
```

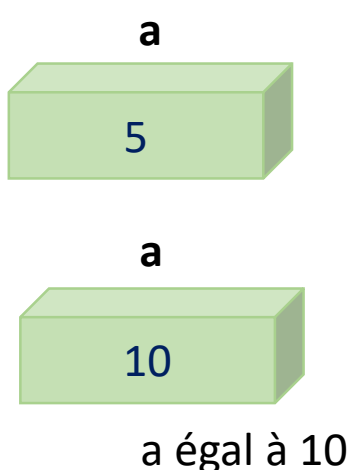
# Les erreurs

---

## 1. L'erreur sémantique ou logique

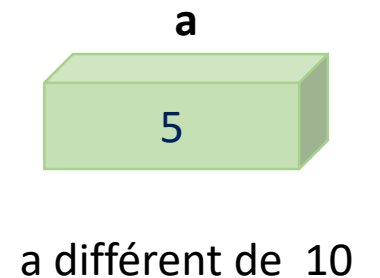
- Le programme s'exécute parfaitement, (pas de message d'erreur) **mais le résultat n'est pas celui que vous attendiez.**
- La sémantique (la logique) est incorrecte.
- Rechercher des fautes de logique peut être une tâche ardue.

**Exemples :** **signe d'affectation (erreur)** à la place du **signe de comparaison**



```
if(a = 10):  
    print "a égal à 10"  
else:  
    print "a différent de 10"
```

```
if(a == 10):  
    print "a égal à 10"  
else:  
    print "a différent de 10"
```



# Les erreurs

---

## 3. L'erreur à l'exécution (Run-time error)

- Elle apparaît seulement lorsque votre programme **fonctionne déjà**, mais que des **circonstances particulières** se présentent (par exemple, votre programme essaie de lire un fichier qui n'existe plus).
- Ces erreurs sont également appelées **des exceptions**, parce qu'elles indiquent généralement que quelque chose d'exceptionnel s'est produit (et qui n'avait pas été prévu).
- Vous rencontrerez davantage ce type d'erreur lorsque vous programmerez des projets de plus en plus volumineux

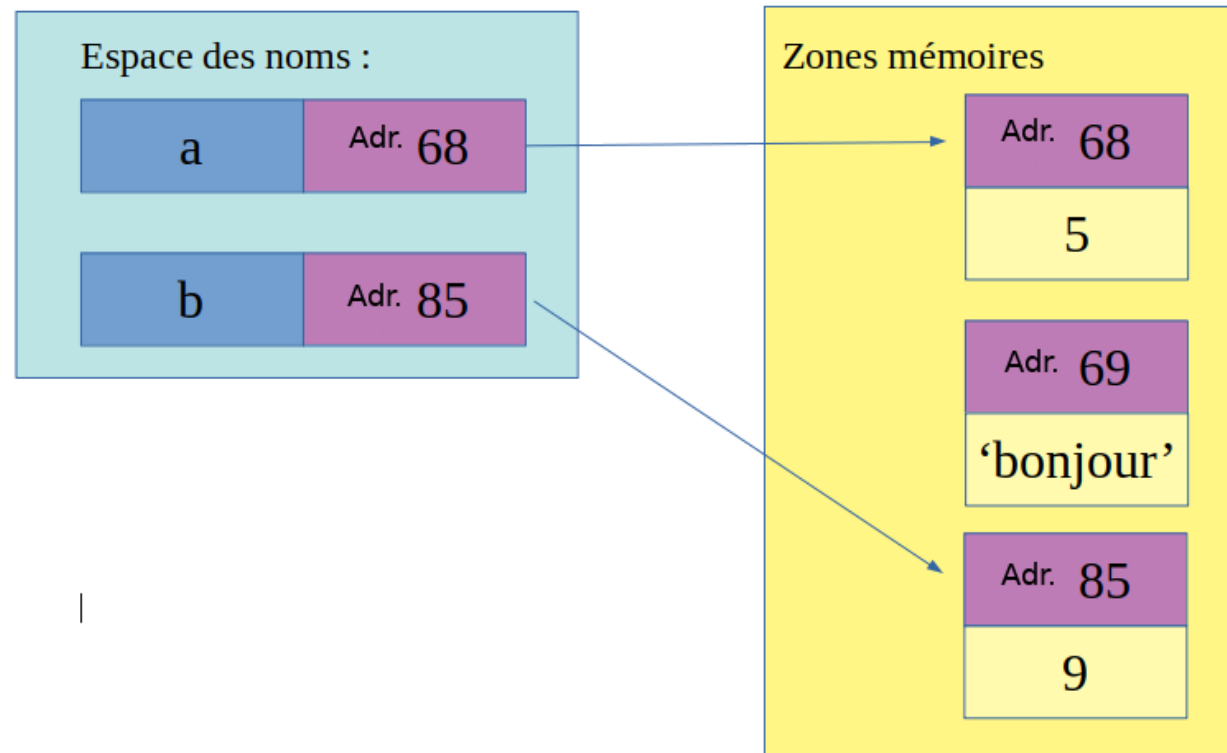
```
>>> print (L)
[1, 2]
>>> print(L[2])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

```
>>> a=10
>>> b=12
>>> c=a/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

# Les variables (1)

---

- Une variable= conteneur d'information qui porte un nom
- = référence à une adresse mémoire
- Les noms des variables sont conventionnellement écrits en minuscule.
- Ils doivent être différents des mots réservés de Python.





# Les variables (2)

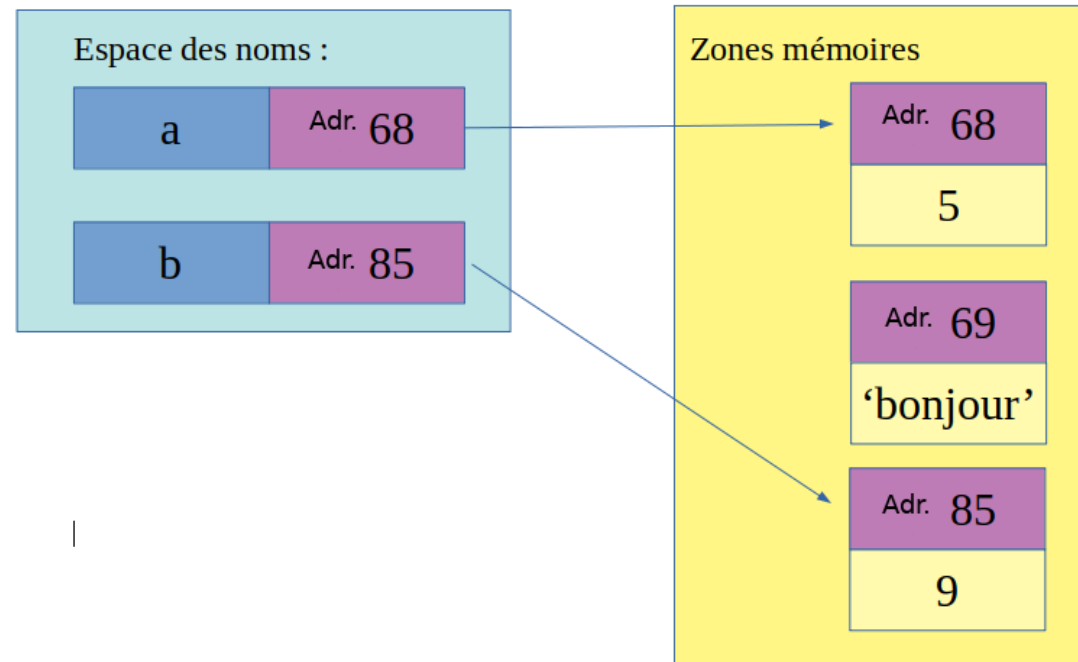
- ✓ Jamais de caractères accentués
- ✓ Jamais de blanc entre les mots

## Syntaxe proposée et appliquée

- Variable : nom**De**Variable
- Sous-programme :  
nom**Fonction**(..)

## Exemple

- Variable : valMin
- Sous-programme :  
ecrire**Chaine**(...)



# Les variables (3)

---

## Typage des variables (spécifique à Python)

- Il n'est pas nécessaire de **définir le type des variables** avant de pouvoir les utiliser.
- il suffit d'assigner une valeur à un nom de variable pour que celle-ci soit automatiquement créée avec le type qui correspond au mieux à la valeur fournie.
- Par exemple :

```
n = 10    msg = "Bonjour" euro = 6,55957
```

Python typerait automatiquement ces trois variables :

- n sera de type entier (integer)
- msg sera de type chaîne de caractères (string)
- euro sera de type réel (float)



L'instruction **type(variable)** permet de connaître le type d'une variable

# Les variables (4)

---

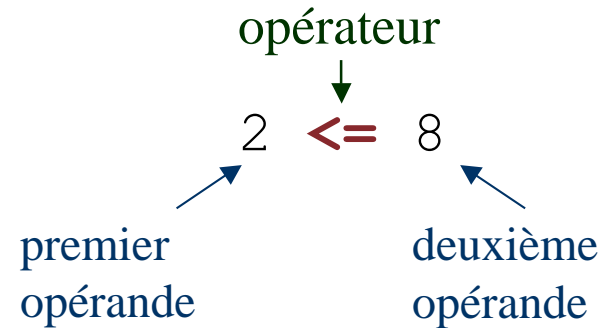
## Exemple :

```
>>> a=10
>>> print(type(a))
<class 'int'>
>>> a="abc"
>>> print(type(a))
<class 'str'>
>>> a=1.3
>>> print(type(a))
<class 'float'>
>>> a=[1,2,3]
>>> print(type(a))
<class 'list'>
```

# Opérateurs et expressions

---

## Expression



- Les opérateurs Python ne sont pas seulement les quatre opérateurs mathématiques de base.
- Il faut ajouter :
  - l'opérateur `**` pour l'exponentiation,
  - des opérateurs logiques, (`or`, `and`, `not`)
  - des opérateurs agissant sur les chaînes de caractères,
  - des opérateurs effectuant des tests d'identité ou d'appartenance,
  - Voir : [https://fr.wikibooks.org/wiki/Programmation\\_Python/Op%C3%A9rateurs](https://fr.wikibooks.org/wiki/Programmation_Python/Op%C3%A9rateurs)

# Expressions booléennes

---

**Deux valeurs possibles** : False, True.

Opérateurs de comparaison : ==, !=, >, >=, <, <=

2 > 8                      # False

2 <= 8 < 15            # True

Opérateurs logiques : not, or, and

(3 == 3) **or** (9 > 24)    # True (dès le premier membre)

(9 > 24) **and** (3 == 3)    # False (dès le premier membre)

# L'affectation

---

On affecte une valeur à une variable en utilisant le **signe "="**

Dans une affectation, la partie de gauche reçoit ou prend pour valeur la partie droite :

`a = 2`      # prononcez : a "reçoit" 2  
# ou a "prend pour  
valeur" 2

La valeur d'une variable peut évoluer au cours du temps (la valeur antérieure est perdue) :

`a = a + 1`      # 3  
(incréméntation)  
`a = a - 1`      # 2  
(décéméntation)

# L'affectation

---

## Affecter n'est pas comparer !

- l'affectation a un effet mais n'a pas de valeur :

`a = b`

# effet : a reçoit la valeur contenue dans b

# valeur de l'expression : aucune

- la comparaison a une valeur mais n'a pas d'effet :

`a == b`

# valeur de l'expression : True ou False

# effet : aucun

# L'affectation

---

Outre l'affectation simple, on peut aussi utiliser **les formes suivantes** :

```
a = 4          # forme de base
a += 2         # idem à : a = a + 2 si a existe déjà
c = d = 8      # cibles multiples
               # (affectation de droite à gauche)
e, f = 2.7, 5.1 # affectation de tuple (par position)
e, f = f, e     # échange les valeurs de e et f
g, h = ['G', 'H'] # affectation de liste (par position)
```



# Entrées

---

- L'instruction **input()** permet d'effectuer une saisie. Le résultat est toujours une chaîne de caractères

```
a1 = input("Entrez un flottant : ") # a1 contient une
chaine (ex : '10.52')
a = float(a1) # transtypage en flottant ou plus
brièvement :
b = float(input(" Entrez un autre flottant : "))
```

# Sorties

---

- L'instruction **print** permet d'afficher des sorties à l'écran :

```
# par exemple a = 2.45 et b = 32  
  
print (a)                                # 2.45  
print ("Somme : ", a + b)                # 34.45  
print ("Différence : ", a - b)           # -29.55  
print ("produit : ", a * b)              # 78.4
```

# **TD 1 : Prise en main de Python**



**Merci pour votre attention**