

# PARADIGMES DE PROGRAMMATION

DIPLOME DE FORMATION D'INGÉNIEURS EN INFORMATIQUE (FI-A1)

## *Module 2*

*Mondher Bouden*



**2024-2025**

# **Programmation procédurale**

# Caractéristiques

---

- La programmation procédurale est basée sur le concept d'appel procédural.
- Une procédure contient une série d'étapes à réaliser.
- Elle peut être appelée à n'importe quelle étape de l'exécution du programme:
  - À l'intérieur d'autres procédures.
  - Dans la procédure elle-même (récursivité).

# Caractéristiques

---

- Par opposition à la programmation séquentielle, la programmation procédurale permet de réutiliser le même code à différents endroits dans le programme
  - Réduction de la taille du code source.
  - Amélioration de la maintenabilité.
- Elle permet de se passer d'instructions telles que « *goto* », évitant ainsi bien souvent de se retrouver avec un programme compliqué qui part dans toutes les directions.
  - Appelé souvent « programmation spaghetti.»

# Caractéristiques

---

- La programmation procédurale permet les « effets de bord »,
  - La possibilité pour une procédure qui prend des arguments de modifier des variables extérieures à la procédure auxquelles elle a accès.
  - Variables de contexte plus global que la procédure.
  - La programmation impérative permet l'emploi des effets de bord dans le fonctionnement de ses programmes contrairement à la programmation fonctionnelle pure.

# Examples

---

- Algol
- C
- COBOL
- Fortran
- Pascal
- Perl
- PL/SQL
- VBScript
- Python
- Etc.

# Le langage Python

---

- Langage interprété (facilite le prototypage) ;
- Typage dynamique (simplifie la programmation) ;
- Interactif ;
- Librairie standard très complète, grande quantité de librairies third-party ;
- Syntaxe concise, plus simple pour commencer

C++ :

```
# include <iostream >
using namespace std;
int main()
{
    cout << "Hello world" << endl;
    return 0;
}
```

Python :

```
print ("Hello world")
```

# D'accord, mais est-ce utilisé dans l'industrie ?

---

- Utilisé par :
  - Google
  - Youtube
  - BitTorrent
  - Intel, Cisco, HP, IBM
  - Pixar
  - NASA
  - Etc.



# Et on peut faire quoi avec ?

---

- Analyser des données (*pandas, matplotlib*)
- Faire du calcul scientifique (*numpy, scipy*)
- Faire de l'apprentissage automatique (*scikit-learn*)
- Construire des sites Web (*flask, cherrypy*)
- Maintenir des serveurs
- Programmer des jeux vidéos (*pygame*)
- Faire de la robotique avec un *Raspberry Pi* !
- Et beaucoup plus !

# Manuel et logiciels du cours

---

- "Apprendre à programmer avec Python 3" de Gérard Swinnen.
  - La version numérique de cet ouvrage est disponible en téléchargement libre via le lien suivant:
  - [http://inforef.be/swi/download/apprendre\\_python3\\_5.pdf](http://inforef.be/swi/download/apprendre_python3_5.pdf)
  - Version arabe:
  - [http://inforef.be/swi/download/apprendre\\_python3\\_arab.pdf](http://inforef.be/swi/download/apprendre_python3_arab.pdf)
- Pour réaliser les laboratoires du cours, vous auriez, essentiellement, besoin des logiciels suivant:
  - Python 3 (<https://www.python.org/>)
  - PyCharm Community Edition (conseillé, mais pas obligatoire):
  - <http://www.jetbrains.com/pycharm/download/>
    - Il faut installer (si ce n'est pas déjà fait) Java:
    - [www.java.com/fr](http://www.java.com/fr)

# Installation

- <https://www.python.org/downloads/release/python-3115/>

Version	Operating System	Description	MD5 Sum	File Size	GPG	Sigstore
<a href="#">Gzipped source tarball</a>	Source release		b628f21aae5e2c3006a12380905bb640	26571003	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">XZ compressed source tarball</a>	Source release		393856f1b7713aa8bba4b642ab9985d3	20053580	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">macOS 64-bit universal2 installer</a>	macOS	for macOS 10.9 and later	7a24f8b4eeca34899b7d75caaec3bc73	44239554	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">Windows embeddable package (32-bit)</a>	Windows		add17856887d34c04a9cfd6c051c4bea	10053367	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">Windows embeddable package (64-bit)</a>	Windows		c5e83dc45630df2236720a18170bf941	11170359	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">Windows embeddable package (ARM64)</a>	Windows		8fc7d74daf27882f2a32a1b10c3a3a2c	10428395	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">Windows installer (32-bit)</a>	Windows		ac8e48a759a6222ce9332691568fe67a	24662424	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">Windows installer (64-bit)</a>	Windows	Recommended	3afd5b0ba1549f5b9a90c1e3aa8f041e	25932664	<a href="#">SIG</a>	<a href="#">.sigstore</a>
<a href="#">Windows installer (ARM64)</a>	Windows	Experimental	cd2bfd6bb39a6c84dbf9d1615b9f53b5	25197192	<a href="#">SIG</a>	<a href="#">.sigstore</a>

32 bits



64 bits



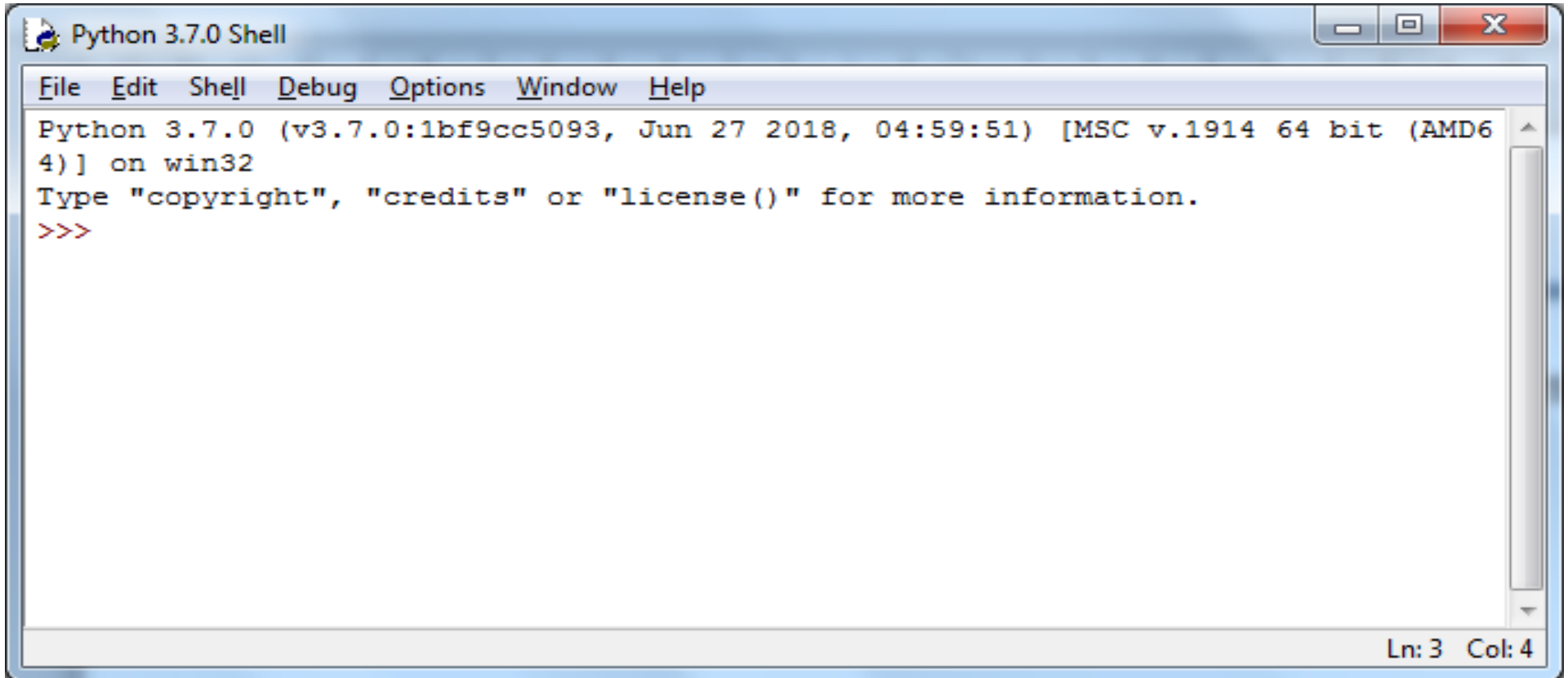
# Premiers pas avec Python

---

- Vous allez commencer à utiliser le mode interactif de Python.
  - Dialoguer avec l'interpréteur directement depuis le clavier.
  - Découvrir très vite un grand nombre de fonctionnalités du langage.
- Par la suite, vous allez apprendre comment créer vos premiers programmes (scripts) et les sauvegarder sur disque.

# Premiers pas avec Python

- Il est conseillé de commencer par utiliser l'environnement de développement fourni par Python (IDLE):

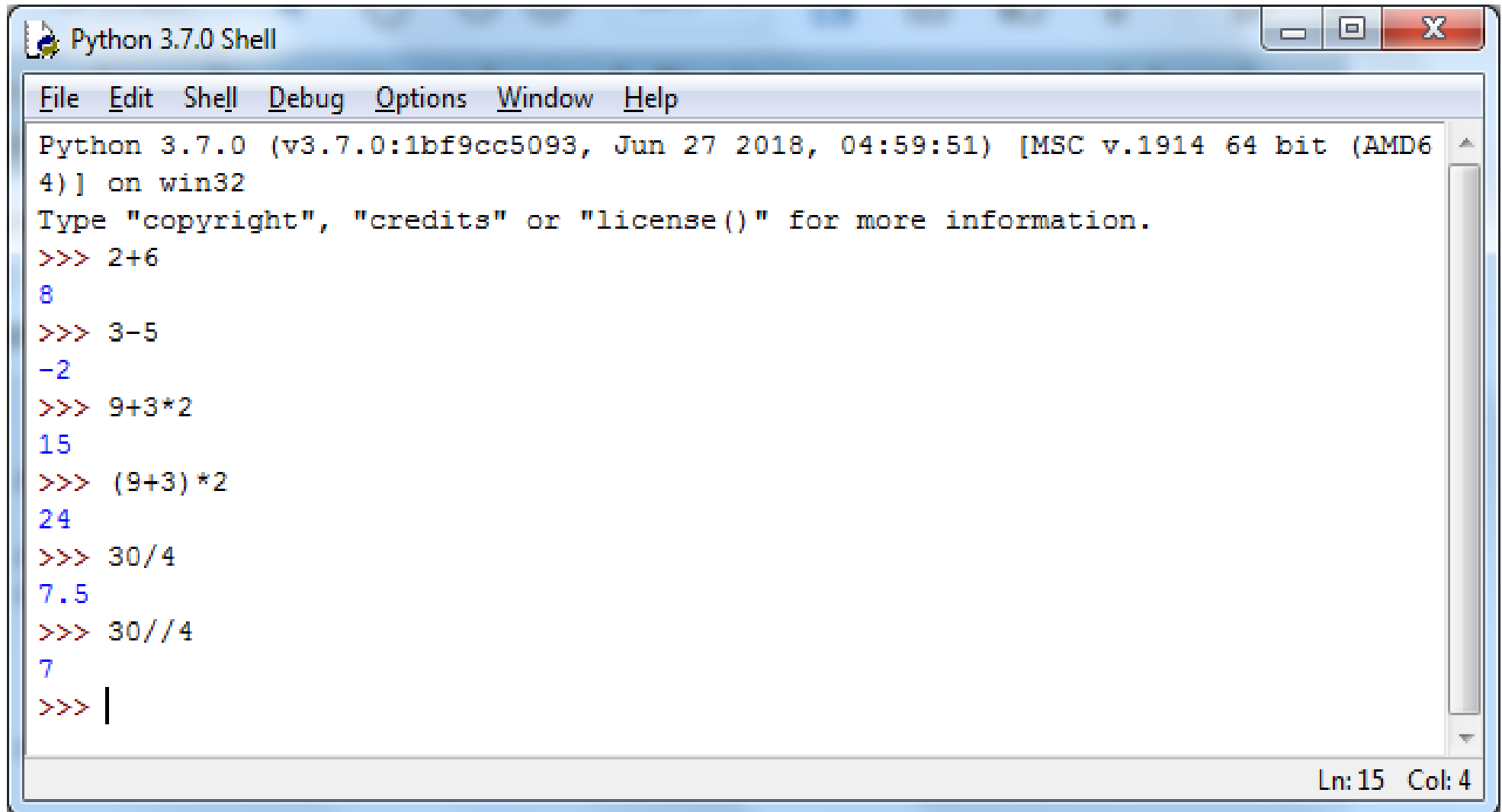
A screenshot of the Python 3.7.0 Shell window. The window has a title bar that says "Python 3.7.0 Shell" and standard Windows window controls (minimize, maximize, close). Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Window, and Help. The main text area displays the following text: "Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32", followed by "Type 'copyright', 'credits' or 'license()' for more information.", and then the prompt ">>>". The status bar at the bottom right shows "Ln: 3 Col: 4".

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

- Les trois caractères >>> constituent le signal d'invité, ou *prompt principal*, lequel vous indique que Python est prêt à exécuter une commande.

# Premiers pas avec Python

- Commençons par faire des opérations simples:

A screenshot of the Python 3.7.0 Shell window. The window has a title bar with the text 'Python 3.7.0 Shell' and standard Windows window controls (minimize, maximize, close). Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options, Window, and Help. The main text area contains the following text:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2+6
8
>>> 3-5
-2
>>> 9+3*2
15
>>> (9+3)*2
24
>>> 30/4
7.5
>>> 30//4
7
>>> |
```

The text is displayed in a monospaced font. The prompt characters '>>>' are in red, and the output results are in blue. A vertical scrollbar is visible on the right side of the text area. At the bottom right of the window, the status bar shows 'Ln: 15 Col: 4'.

# Les types de données de base

---

- Entier : `int`, sans partie décimale.
- Réel : `float`, avec partie décimale.
- La représentation interne varie, l'interprétation (l'utilisation) aussi.
- Opérateurs arithmétiques : `+`, `-`, `*`, `/`, `//`, `%`, `**`
- Fonctions mathématiques : `abs(arg)`, `int(arg)`, `float(arg)`, etc.

Nom	Adresse	Contenu	Type
age_retraite	1002	65	int
age_embauche	1003	60.0	float
...	1004	...	...

# Les opérateurs arithmétiques

---

Opérateur	Exemple	x=9 et y=4
+	x + y	13
-	x - y	5
*	x * y	36
/	x / y	2.25
//	x // y	2
%	x % y	1
**	x ** y	6561



# Priorité d'évaluation

---

- Les parenthèses d'abord : `()`
- Les fonctions : `abs(arg)`, etc.
- L'exposant : `**`
- Les opérateurs unaires : `-`, `+`
- Les opérateurs binaires : `*`, `/`, `//`, `%`
- Les opérateurs binaires : `+`, `-`
- Si deux opérateurs ont la même priorité, l'évaluation est effectuée de gauche à droite.

# Le type Booléen : True ou False

---

- Opérateurs de comparaison : <, >, <=, >=, ==, !=
- Moins prioritaires que les opérateurs arithmétiques
- False correspond à la valeur 0
- True correspond à toute valeur différente de 0, généralement 1
- `erreur_emb = age_embauche > age_retraite`

Nom	Adresse	Contenu	Type
age_retraite	1002	65	int
age_embauche	1003	60.0	float
...	1004	...	...

# Le type Booléen : True ou False

---

- Opérateurs de comparaison : <, >, <=, >=, ==, !=
- Moins prioritaires que les opérateurs arithmétiques
- False correspond à la valeur 0
- True correspond à toute valeur différente de 0, généralement 1
- `erreur_emb = age_embauche > age_retraite`

Nom	Adresse	Contenu	Type
age_retraite	1002	65	int
age_embauche	1003	60.0	float
erreur_emb	1004	...	...

# Le type Booléen : True ou False

---

- Opérateurs de comparaison : <, >, <=, >=, ==, !=
- Moins prioritaires que les opérateurs arithmétiques
- False correspond à la valeur 0
- True correspond à toute valeur différente de 0, généralement 1
- `erreur_emb = age_embauche > age_retraite`

Nom	Adresse	Contenu	Type
<code>age_retraite</code>	1002	65	int
<code>age_embauche</code>	1003	60.0	float
<code>erreur_emb</code>	1004	False	...

# Le type Booléen : True ou False

---

- Opérateurs de comparaison : <, >, <=, >=, ==, !=
- Moins prioritaires que les opérateurs arithmétiques
- False correspond à la valeur 0
- True correspond à toute valeur différente de 0, généralement 1
- `erreur_emb = age_embauche > age_retraite`

Nom	Adresse	Contenu	Type
age_retraite	1002	65	int
age_embauche	1003	60.0	float
erreur_emb	1004	False	int

# Les opérateurs de comparaison

---

Opérateur	Exemple	x=9 et y=4
==	x == y	False
!=	x != y	True
>	x > y	True
>=	x >= y	True
<	x < y	False
<=	x <= y	False

# Les opérateurs logiques

---

Opérateur	Exemple	cond1 = True cond2 = False
not	not cond1	False
not	not cond2	True
and	cond1 and cond2	False
or	cond1 or cond2	True

## Table de vérité : not

---

<u>Opérateur</u>	<u>Valeur</u>	<u>Résultat</u>
not	True	False
not	False	True



## Table de vérité : and

---

valeur1	valeur2	Résultat
True	True	True
True	False	False
False	True	False
False	False	False

## Table de vérité : or

---

<u>valeur1</u>	<u>valeur2</u>	<u>Résultat</u>
True	True	True
True	False	True
False	True	True
False	False	False

# Priorité d'évaluation

---

- Les parenthèses : `()`
- Les fonctions : `abs(arg)`, etc.
- L'exposant : `**`
- Les opérateurs unaires : `-`, `+`
- Les opérateurs binaires : `*`, `/`, `//`, `%`
- Les opérateurs binaires : `+`, `-`
- Les opérateurs de comparaison : `<`, `<=`, `>`, `>=`
- Les opérateurs de comparaison : `==`, `!=`
- L'opérateur logique : `not`
- L'opérateur logique : `and`
- L'opérateur logique : `or`

# Opérateurs de gestion d'information

---

- Les variables : pour mémoriser l'information.
- L'affectation : pour déplacer de l'information.
- L'input : pour acquérir des données provenant de l'extérieur.
- L'output : pour communiquer des données vers l'extérieur.

# Une variable en Python

---

- Représente une donnée.
- Porte un nom, idéalement significatif (qui désigne le contenu d'une case mémoire à une adresse donnée) ;
- Est associée à une case de la mémoire (statique) ;
- A donc une adresse mémoire (statique) ;
- Est du type de la dernière donnée qu'on lui a affectée (dynamique) ;
- Ce type est nécessaire pour **interpréter** le contenu de la variable.

# Un nom de variable en Python

---

- Séquence de lettres (a-z, A-Z) et de chiffres (0-9), qui doit toujours débuter par une lettre ;
- Seul le caractère spécial \_ (souligné) est permis ;
- Nous éviterons les noms débutant ou se terminant par un \_ (ceux-ci ont une signification particulière en Python) ;
- Les majuscules et minuscules représentent différentes lettres ;
- Il faut éviter les 33 mots réservés qui ont une signification prédéfinie ;
- Par convention, les variables en Python utilisent le format suivant :

`ma_variable.`

# Les mots réservés en Python

---

- Les mots suivants sont réservés par le langage, et ne peuvent donc pas être utilisés comme nom de variable.

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

# Typage des variables

---

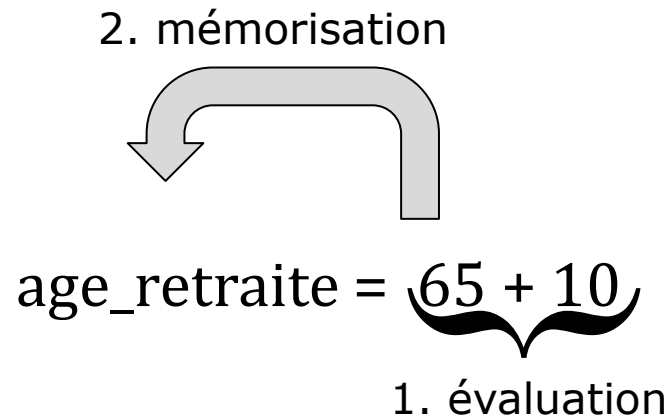
- Le typage des variables sous Python est dynamique.
  - Par opposition au typage statique (ex. C++ ou Java) où il faut d'abord déclarer le nom et le type des variables, et ensuite seulement leur assigner un contenu.
- Sous Python, il suffit d'assigner une valeur à un nom de variable pour que celle-ci soit automatiquement créée avec le type qui correspond au mieux à la valeur fournie.
- Le typage statique est préférable dans le cas des langages compilés
  - Permet d'optimiser l'opération de compilation.
- Le typage dynamique permet d'écrire plus aisément des constructions logiques de niveau élevé.



# L'opération d'affectation

---

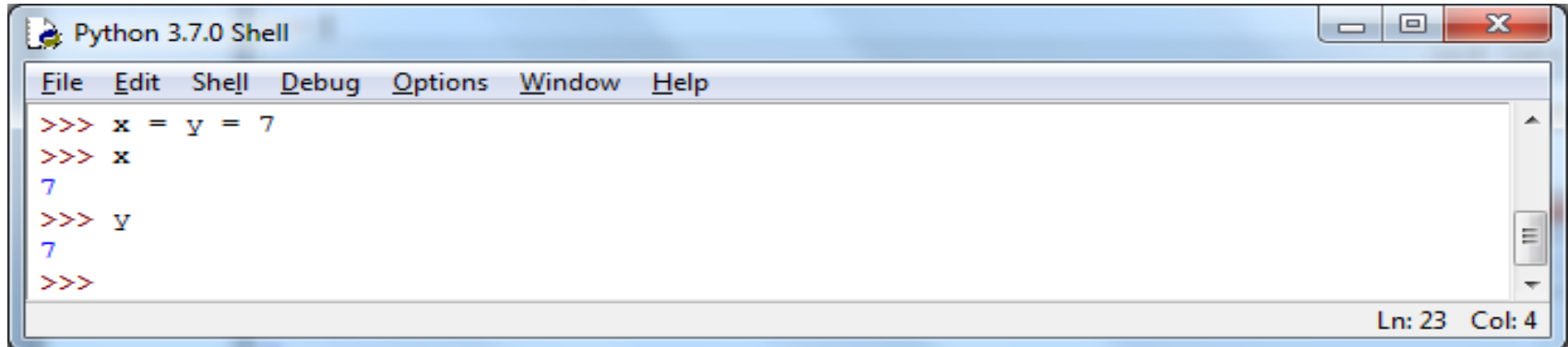
- Évalue l'expression à droite du symbole =.
- Mémoire le résultat de l'évaluation de cette expression dans la variable identifiée à gauche du symbole =.



- Ce que vous placez à gauche du signe égal doit toujours être une variable.
  - L'instruction  $m + 1 = b$  est illégale.

# L'opération d'affectation

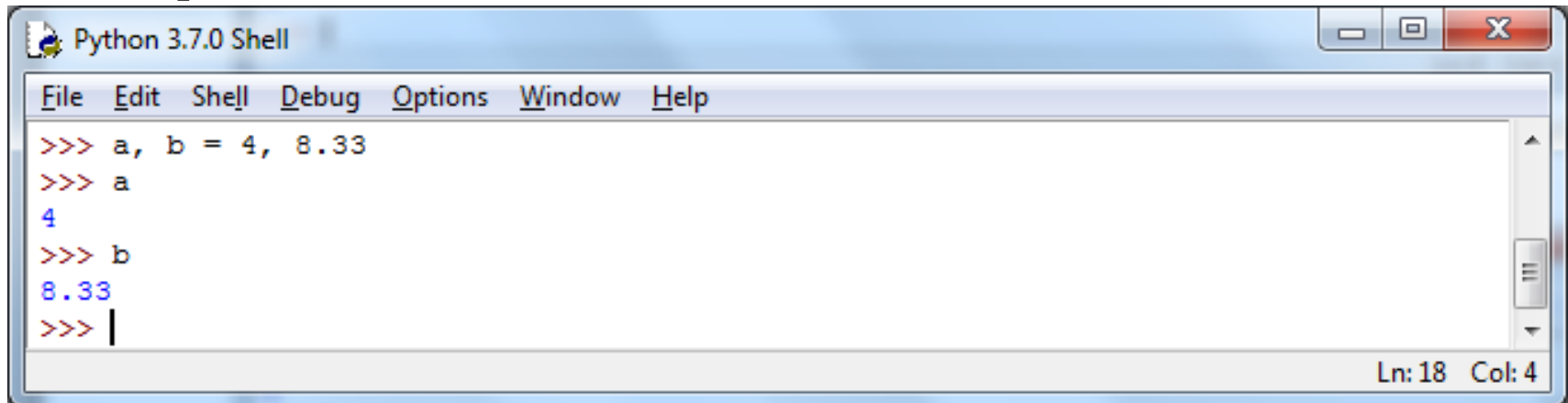
- Sous Python, on peut assigner une valeur à plusieurs variables simultanément. Exemple :



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
>>> x = y = 7
>>> x
7
>>> y
7
>>>
```

Ln: 23 Col: 4

- On peut aussi effectuer des affectations parallèles à l'aide d'un seul opérateur :



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
>>> a, b = 4, 8.33
>>> a
4
>>> b
8.33
>>> |
```

Ln: 18 Col: 4

# Les variables

- *variable* = ... : mémoriser une information, `age_retraite = 65`

1. Trouver ou créer la variable nommée à gauche de =.
2. Évaluer la valeur de l'expression à droite de =.
3. Mettre cette valeur dans la variable nommée à gauche de =.
4. Mémoriser le type de cette valeur comme type de la variable.

Nom	Adresse	Contenu	Type
...	1000	...	...
...	1001	...	...
	1002		
	1003		
	1004		

# Les variables

- *variable* = ... : mémoriser une information, age\_retraite = 65

1. Trouver ou créer la variable nommée à gauche de =.
2. Évaluer la valeur de l'expression à droite de =.
3. Mettre cette valeur dans la variable nommée à gauche de =.
4. Mémoriser le type de cette valeur comme type de la variable.

Nom	Adresse	Contenu	Type
...	1000	...	...
...	1001	...	...
age_retraite	1002		
	1003		
	1004		

# Les variables

- *variable* = ... : mémoriser une information, age\_retraite = 65

1. Trouver ou créer la variable nommée à gauche de =.
2. Évaluer la valeur de l'expression à droite de =.
3. Mettre cette valeur dans la variable nommée à gauche de =.
4. Mémoriser le type de cette valeur comme type de la variable.

Nom	Adresse	Contenu	Type
...	1000	...	...
...	1001	...	...
age_retraite	1002	65	
	1003		
	1004		

# Les variables

- *variable* = ... : mémoriser une information, age\_retraite = 65

1. Trouver ou créer la variable nommée à gauche de =.
2. Évaluer la valeur de l'expression à droite de =.
3. Mettre cette valeur dans la variable nommée à gauche de =.
4. Mémoriser le type de cette valeur comme type de la variable.

Nom	Adresse	Contenu	Type
...	1000	...	...
...	1001	...	...
age_retraite	1002	65	entier
	1003		
	1004		

# Les variables

- *variable* = ... : mémoriser une information, age\_embauche = 60.0

1. Trouver ou créer la variable nommée à gauche de =.
2. Évaluer la valeur de l'expression à droite de =.
3. Mettre cette valeur dans la variable nommée à gauche de =.
4. Mémoriser le type de cette valeur comme type de la variable.

Nom	Adresse	Contenu	Type
...	1000	...	...
...	1001	...	...
age_retraite	1002	65	entier
	1003		
	1004		

# Les variables

- *variable* = ... : mémoriser une information, age\_embauche = 60.0

1. Trouver ou créer la variable nommée à gauche de =.
2. Évaluer la valeur de l'expression à droite de =.
3. Mettre cette valeur dans la variable nommée à gauche de =.
4. Mémoriser le type de cette valeur comme type de la variable.

Nom	Adresse	Contenu	Type
...	1000	...	...
...	1001	...	...
age_retraite	1002	65	entier
age_embauche	1003		
	1004		



# Les variables

- *variable* = ... : mémoriser une information, age\_embauche = 60.0

1. Trouver ou créer la variable nommée à gauche de =.
2. Évaluer la valeur de l'expression à droite de =.
3. Mettre cette valeur dans la variable nommée à gauche de =.
4. Mémoriser le type de cette valeur comme type de la variable.

Nom	Adresse	Contenu	Type
...	1000	...	...
...	1001	...	...
age_retraite	1002	65	entier
age_embauche	1003	60.0	
	1004		

# Les variables

- *variable* = ... : mémoriser une information, age\_embauche = 60.0

1. Trouver ou créer la variable nommée à gauche de =.
2. Évaluer la valeur de l'expression à droite de =.
3. Mettre cette valeur dans la variable nommée à gauche de =.
4. Mémoriser le type de cette valeur comme type de la variable.

Nom	Adresse	Contenu	Type
...	1000	...	...
...	1001	...	...
age_retraite	1002	65	entier
age_embauche	1003	60.0	réel
	1004		

# Les variables

- *variable* = ... : mémoriser une information, `age_retraite = 65.0`

1. Trouver ou créer la variable nommée à gauche de =.
2. Évaluer la valeur de l'expression à droite de =.
3. Mettre cette valeur dans la variable nommée à gauche de =.
4. Mémoriser le type de cette valeur comme type de la variable.

Nom	Adresse	Contenu	Type
...	1000	...	...
...	1001	...	...
age_retraite	1002	65	entier
age_embauche	1003	60.0	réel
	1004		

# Les variables

- *variable* = ... : mémoriser une information, `age_retraite = 65.0`

1. Trouver ou créer la variable nommée à gauche de =.
2. Évaluer la valeur de l'expression à droite de =.
3. Mettre cette valeur dans la variable nommée à gauche de =.
4. Mémoriser le type de cette valeur comme type de la variable.

Nom	Adresse	Contenu	Type
...	1000	...	...
...	1001	...	...
age_retraite	1002	65.0	réel
age_embauche	1003	60.0	réel
	1004		

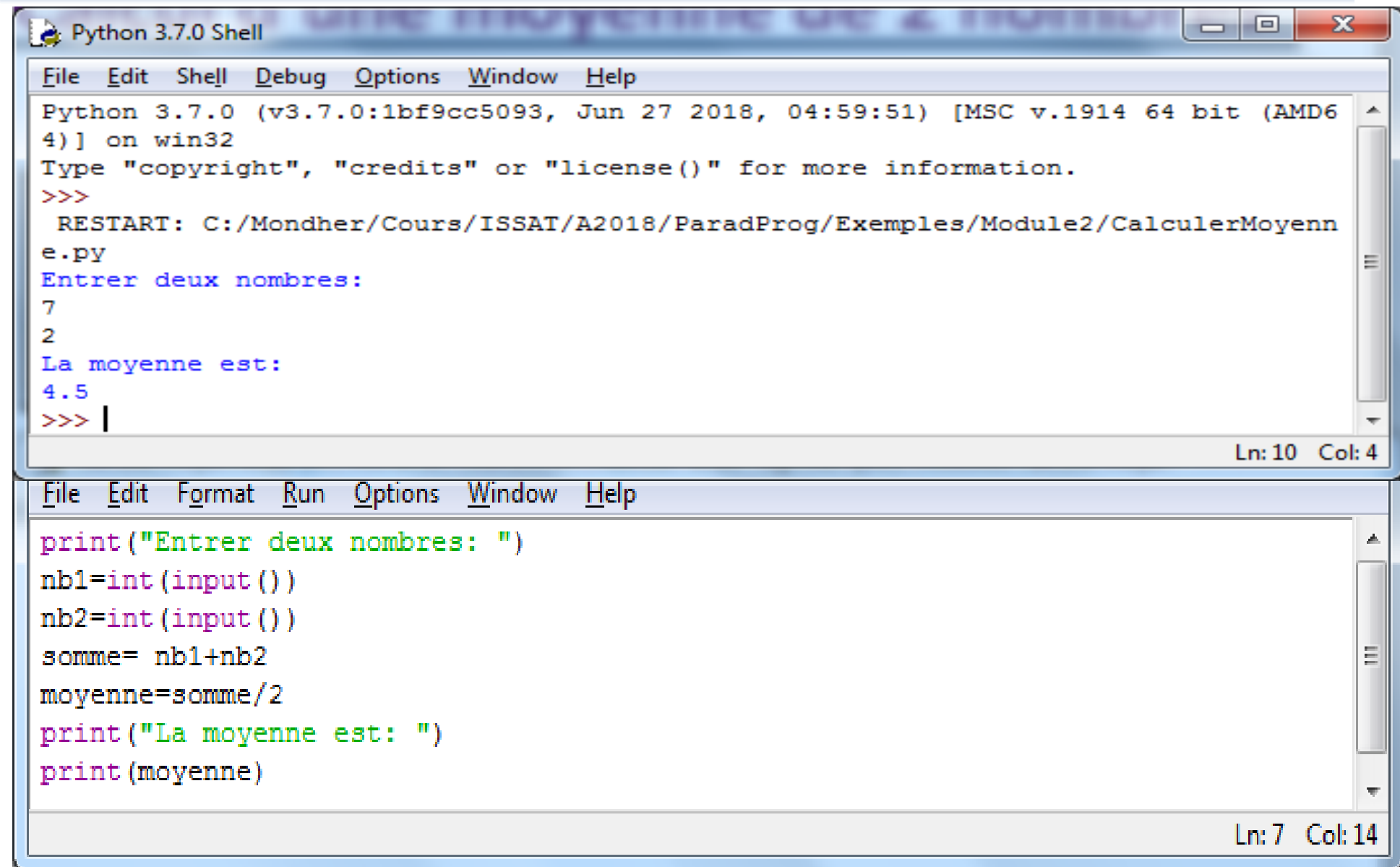
# Les entrées-sorties

---

- `input()`
  - Du clavier
  - Avec mémorisation temporaire
  - Convertit la chaîne de caractères selon le type requis : p.ex. les 2 caractères consécutifs « 6 » et « 5 » sont convertis en valeur numérique 65
- `output()` (La fonction `print()` permet d'afficher le contenu des variables).
  - À l'écran
  - Convertit la donnée en une chaîne de caractères correspondants ; p.ex. la valeur numérique 60.0 est convertie en 4 caractères affichés consécutivement : « 6 » « 0 » « . » « 0 »

# Exemple

## Calcul de la moyenne de 2 nombres



The image shows two windows from a Python 3.7.0 Shell. The top window displays the execution of a program named 'CalculerMoyenne.py'. The program prompts the user to enter two numbers, which are 7 and 2. It then calculates the average, which is 4.5. The bottom window shows the source code of the program.

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
  RESTART: C:/Mondher/Cours/ISSAT/A2018/ParadProg/Exemples/Module2/CalculerMoyenne.py
  Entrer deux nombres:
  7
  2
  La moyenne est:
  4.5
  >>> |
Ln: 10 Col: 4
```

```
File Edit Format Run Options Window Help
print("Entrer deux nombres: ")
nb1=int(input())
nb2=int(input())
somme= nb1+nb2
moyenne=somme/2
print("La moyenne est: ")
print(moyenne)
Ln: 7 Col: 14
```