

# Premiers pas

Introduction à la programmation en Python



# *Sommaire*

1. Autour de la notion d'algorithme.
2. Développer en Python.
3. Les variables en Python.



1. Autour de la notion d'algorithme.

# 1. Autour de la notion d'algorithme.

## **Notion d'algorithme : définition**

- Suite d'instructions permettant la résolution d'un problème en un nombre fini d'étapes.
- Transforme des données (entrées de l'algorithme) en un résultat (la sortie).
- Les instructions doivent être indépendantes des données.



# 1. Autour de la notion d'algorithme.

## **Notion d'algorithme : exemple d'une recette de cuisine**

- Problème : comment faire une pâte à crêpe ?
- Résolution :
  1. Casser trois œufs dans 250 grammes de farine.
  2. Mélanger.
  3. Diluer avec un demi-litre de lait.
  4. Ajouter une cuillère à soupe d'huile.



# 1. Autour de la notion d'algorithme.

## **Notion d'algorithme : un peu d'histoire**

- Origine du mot algorithme : nom du mathématicien perse Abu Abdullah Muhammad ibn Musa al-Khwarizmi (9 ème siècle après J.-C.).
- Nom latinisé au Moyen Âge en “algoritmi”.
- Le concept est cependant plus ancien, les premiers exemples remontant à l'antiquité.



# 1. Autour de la notion d'algorithme.

## Notion d'algorithme : exemple de calculs d'intérêts (1800 avant J.C.)

- Problème : combien d'années faut-il pour doubler un montant soumis à un taux  $t$  annuel ?
- Résolution :
  1. Initialiser  $t$ , une variable  $a = 0$  (le nombre d'année) et  $b = 1$  (le montant).
  2. Tant que  $b < 2$ , remplacer  $b$  par  $b \times (1 + t)$  et ajouter 1 à  $a$ .
  3. La valeur finale de  $a$  est le nombre d'années cherché.



# 1. Autour de la notion d'algorithme.

## Déroulé de l'algorithme précédent sur un exemple

- Exemple avec  $t = 0,2$

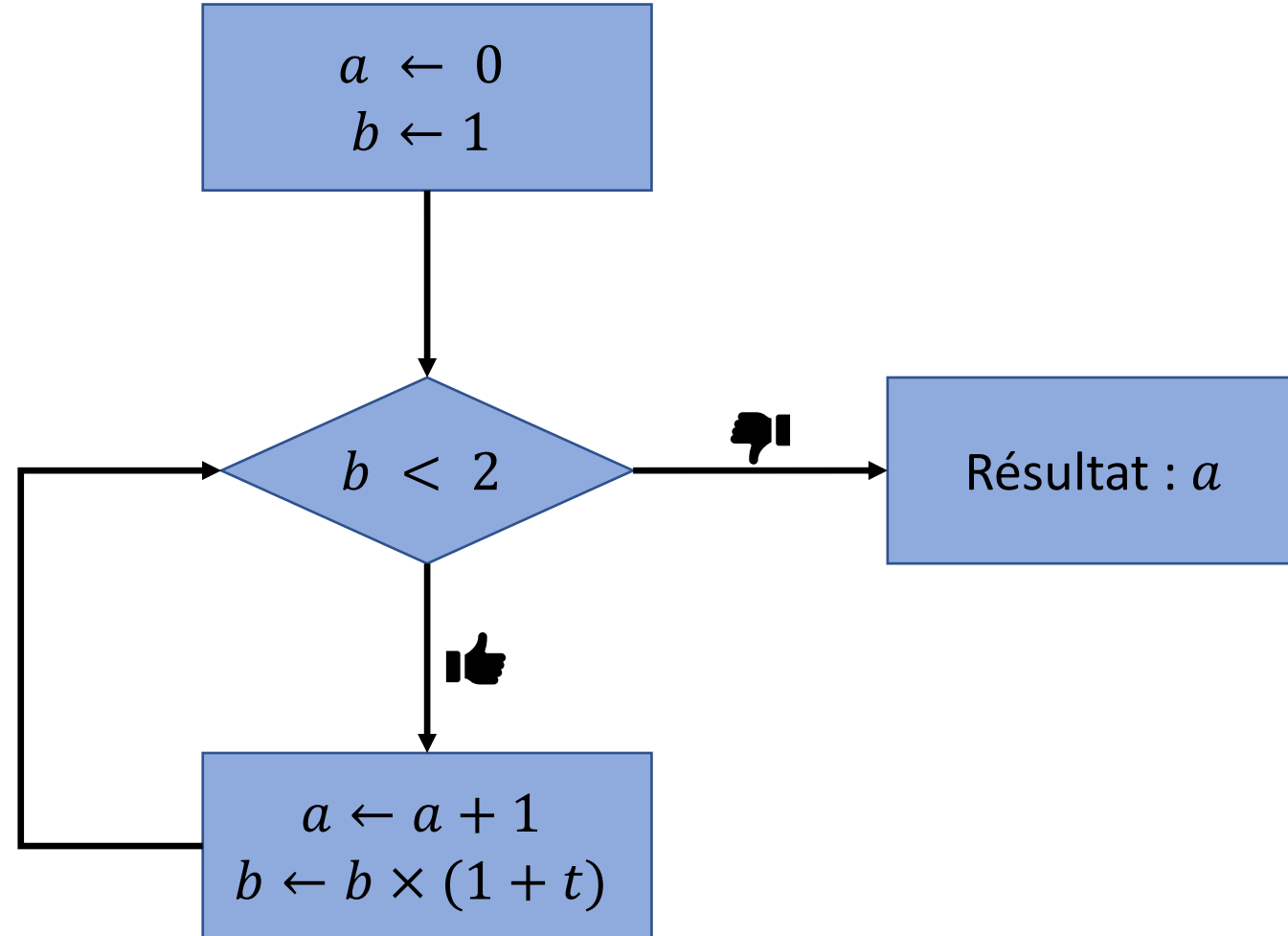
| <i>a</i> | <i>b</i> |
|----------|----------|
| 0        | 1        |
| 1        | 1,2      |
| 2        | 1,44     |
| 3        | 1,728    |
| 4        | 2,0736   |



# 1. Autour de la notion d'algorithme.

## Notion d'algorithme : formalisation

Un schéma peut tout à fait servir pour concevoir et exprimer un algorithme.



# 1. Autour de la notion d'algorithme.

## De l'algorithme au programme

- La conception d'un algorithme peut (doit) se faire de façon non formelle en version papier.
- Ses instructions seront ensuite traduites dans un langage de programmation.



- L'algorithme contient toute la réflexion, l'abstraction du programme.

# 1. Autour de la notion d'algorithme.

## De l'algorithme au programme

- La phase de traduction de l'algorithme en Python sera souvent relativement simple.

```
def years(t):  
    a, b = 0, 1  
    while b<2:  
        b = b*(1+t)  
        a = a+1  
    return a
```

# 1. Autour de la notion d'algorithme.



## 2. Développer en Python.

## 2. Développer en Python.

### **Avantages du Python comme langage d'apprentissage**

- Syntaxe simple.
- Langage interprété.
- Possède un mode console.
- Langage de haut niveau.
- Structures de données existantes.
- Open source.
- Langage populaire.
- Bibliothèques graphiques faciles à installer.



## 2. Développer en Python.

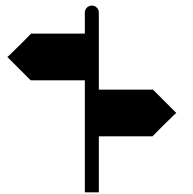
### Téléchargement de l'interpréteur Python

- Disponible sur [python.org](https://python.org)
- Choisir la version la plus récente de la branche 3.x
- Il est téléchargé simultanément un environnement de développement basique : IDLE.

## 2. Développer en Python.

### **Choix d'un IDE (Integrated Development Environment)**

- Se contenter par défaut de IDLE
- Utiliser une solution plus élaborée :
  - Pycharm
  - Komodo
  - Spyder
  - ...





## 2. Développer en Python.

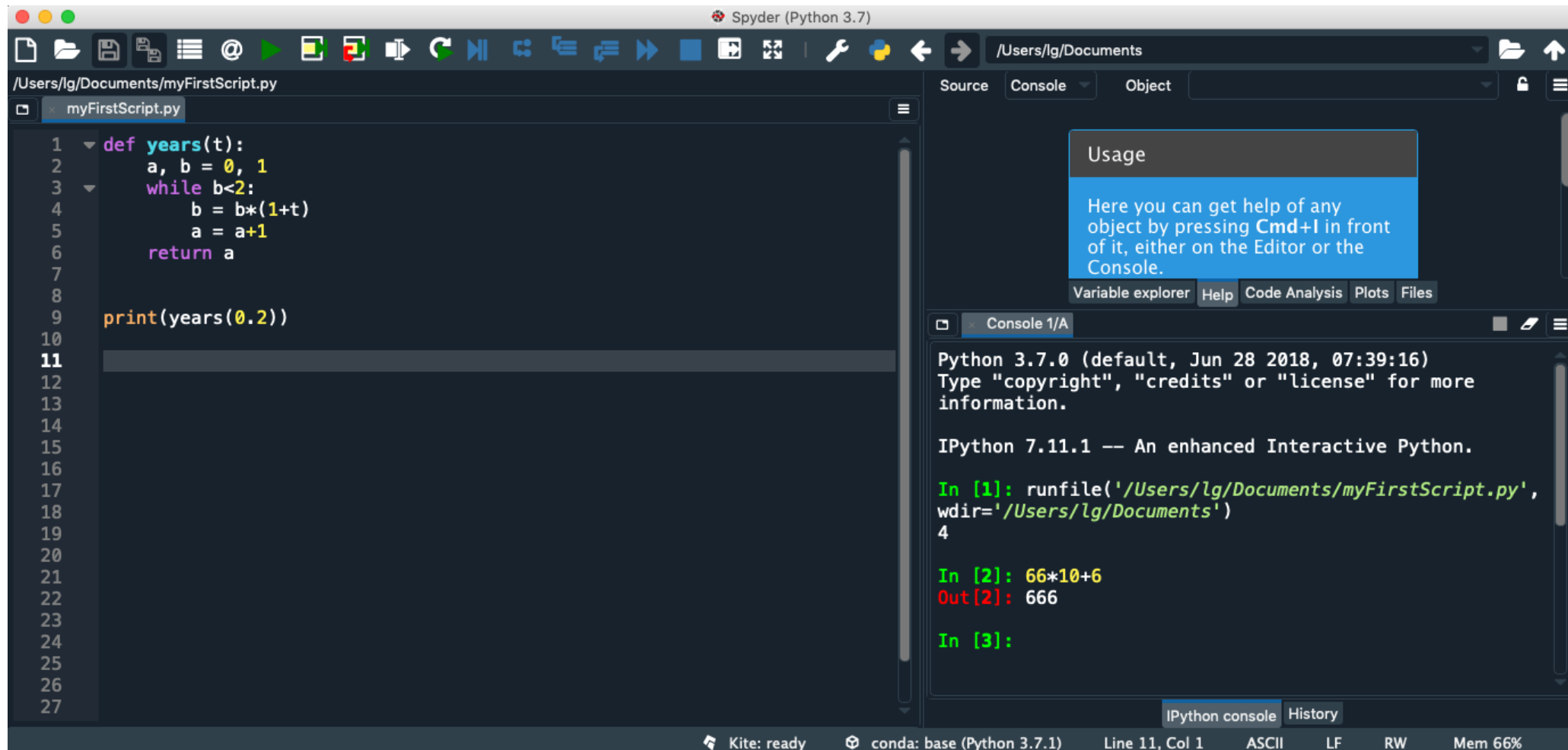
### Les deux modes d'utilisation

- Console :
  - Mode interactif (style calculatrice).
  - Permet de tester du code à la volée.
- Éditeur :
  - Pour écrire et sauvegarder des scripts.



## 2. Développer en Python.

### Les deux modes d'utilisation



## 2. Développer en Python.

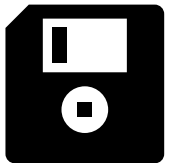


# 3. Les variables en Python.

### 3. Les variables en Python.

#### **Notion de variable : une nécessité**

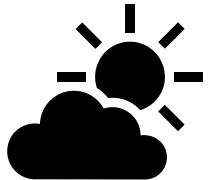
- Un algorithme ou un programme manipulent des données. Certaines sont connues dès le départ, d'autres sont calculées lors de son exécution.
- Pour pouvoir manipuler ces données il faut garder leurs valeurs en mémoire.
- C'est le rôle des variables.



### 3. Les variables en Python.

#### **Notion de variable : définition**

- Une variable est un nom désignant une donnée (nombre, texte, etc.) susceptible de changer de valeur lors de l'exécution d'un programme.
- Une variable possède un type, ce qui permet à l'ordinateur de savoir quelles valeurs elle peut prendre et quelles opérations on peut effectuer avec.



### 3. Les variables en Python.

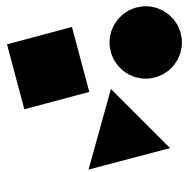
#### Les types de variables en Python

| <i>Types</i> | <i>Valeurs</i>           |
|--------------|--------------------------|
| int          | Nombres entiers naturels |
| float        | Nombres décimaux         |
| complex      | Nombres complexes        |
| bool         | Booléens (True ou False) |
| str          | Chaînes de caractères    |

### 3. Les variables en Python.

#### **Particularités du typage en Python**

- Typage dynamique : pas besoin de déclarer explicitement le type des variables, l'interpréteur le fait lors de l'initialisation.
- Typage fort : pas de conversions de type implicites pour forcer à réaliser certaines opérations (exemple : pas d'addition entre 5 et '6').





### 3. Les variables en Python.

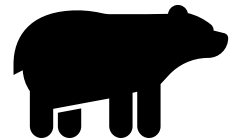
#### Affectations

- Affectation simple :

```
beast = 666
```

- Affectation multiple :

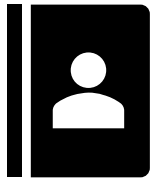
```
pi, e = 3.14, 2.71
```



### 3. Les variables en Python.

#### **Deux fonctions spécifiques**

- `type(...)` : retourne le type (élémentaire ou complexe) d'une variable.
- `id(...)` : retourne l'identifiant (moralement l'adresse mémoire) d'une variable.



### 3. Les variables en Python.

#### Conversions explicites de type

| <i>Types</i>  | <i>Valeurs</i>  |
|---------------|---|
| int(x)        | <b>x</b> est un décimal (qui sera tronqué) ou une chaîne  |
| int(x, base)  | <b>x</b> est une chaîne et base un entier   |
| float(x)      | <b>x</b> est un entier ou une chaîne  |
| complex(x, y) | <b>x</b> et <b>y</b> sont des nombres   |
| bool(x)       | <b>x</b> est un nombre ou une chaîne. Le résultat vaut <b>False</b> uniquement si <b>x</b> vaut 0 ou '' |
| str(x)        | <b>x</b> est un nombre ou un booléen  |
| eval(x)       | <b>x</b> est une chaîne qui est évaluée comme une expression Python                                     |

### 3. Les variables en Python.

#### Affichage et saisie

- Affichage :

```
print(expr1, expr2, ..., exprN, sep=' ', end='\n')
```

- Saisie :

```
var = input(expression)
```



### 3. Les variables en Python.

#### Saisie

- La fonction “input” retourne une chaîne de caractères.
- Pour conserver le côté numérique d’une saisie, il faudra effectuer une conversion avec la fonction “eval” ou avec une conversion explicite de type (“int” ou “float”).



### 3. Les variables en Python.

#### Opérateurs arithmétiques

| <i>Opération</i> | <i>Résultat</i>                                   |
|------------------|---|
| $x + y$          | Addition de <b>x</b> et <b>y</b>                  |
| $x - y$          | Soustraction de <b>y</b> à <b>x</b>               |
| $x * y$          | Multiplication de <b>x</b> et <b>y</b>            |
| $x ** y$         | Élévation de <b>x</b> à la puissance <b>y</b>     |
| $x / y$          | Quotient réel de <b>x</b> par <b>y</b>            |
| $x // y$         | Quotient entier de <b>x</b> par <b>y</b>          |
| $x \% y$         | Reste du quotient entier de <b>x</b> par <b>y</b> |

### 3. Les variables en Python.

#### Opérateurs arithmétiques (raccourcis)

| <i>Opération</i> | <i>Résultat</i> |
|------------------|-----------------|
| $x += y$         | $x = x + y$     |
| $x -= y$         | $x = x - y$     |
| $x *= y$         | $x = x * y$     |
| $x **= y$        | $x = x ** y$    |
| $x /= y$         | $x = x / y$     |
| $x //= y$        | $x = x // y$    |
| $x \% = y$       | $x = x \% y$    |

### 3. Les variables en Python.

#### Opérateurs de comparaison

| <i>Opération</i> | <i>Résultat</i>       |
|------------------|-----------------------|
| <                | Strictement inférieur |
| <=               | Inférieur ou égal     |
| >                | Strictement supérieur |
| >=               | Supérieur ou égal     |
| ==               | Égal                  |
| !=               | Différent             |



### 3. Les variables en Python.

#### Opérateurs logiques

| <i>Opération</i> | <i>Résultat</i> |
|------------------|-----------------|
| and              | ET logique      |
| or               | OU logique      |
| not              | NON logique     |

- Les deux premiers sont des opérateurs binaires, le dernier est unaire.



### 3. Les variables en Python.

#### Tables de vérité des opérateurs logiques

| <i><b>A</b></i> | <i><b>B</b></i> | <i><b>A or B</b></i> | <i><b>A and B</b></i> |
|-----------------|-----------------|----------------------|-----------------------|
| False           | False           | False                | False                 |
| False           | True            | True                 | False                 |
| True            | False           | True                 | False                 |
| True            | True            | True                 | True                  |

| <i><b>A</b></i> | <i><b>not A</b></i> |
|-----------------|---------------------|
| False           | False               |
| True            | True                |



### 3. Les variables en Python.



