Le language Python

Ambre Le Berre

2025/2026

Agenda

1. Rappels du TP
2. Les listes
3. Modules et graphes

Rappels du TP

Les variables

On peut stocker des données sous un nom : c'est une variable.

```
je_suis_une_variable = 3
s = "bonjour"
```

Les variables

On peut stocker des données sous un nom : c'est une variable.

```
je_suis_une_variable = 3
s = "bonjour"
```

On écrit toujours les noms de variables en `snake_case` :

- tout en minuscule (sauf les acronymes)
- un underscore _ entre les mots
- soit tout en français sans accent, soit tout en anglais, pas de mélange

Opérateurs de calculs

On peut faire des calculs entre des valeurs ou des variables.

```
a = 3 + je_suis_une_variable
b = 13039 % 12
```

Opérateurs de calculs

On peut faire des calculs entre des valeurs ou des variables.

```
a = 3 + je_suis_une_variable
b = 13039 % 12
```

On met toujours des espaces avant et après un opérateurs binaire. Pour un opérateurs unaire on ne met pas d'espace.

```
calcul = 3 * 5
nombre_negatif = -7
```

Les commentaires

Les commentaires n’ont aucun impact sur le programme : il sert simplement à comprendre le code plus simplement

```
# Les commentaires commencent par "#"
a = 3 # on peut les mettre à la suite d'une ligne
```

Les commentaires

Les commentaires n'ont aucun impact sur le programme : il sert simplement à comprendre le code plus simplement

```
# Les commentaires commencent par "#"
a = 3 # on peut les mettre à la suite d'une ligne
```

Il est important de commenter votre code dès qu'il n'est pas évident !

- commentaire inutile

```
a = 3 # assigne 3 à la variable a
```

- commentaire utile :

```
cpt = 0 # la variable cpt compte le nombre de bouteille dans la mer
```

Affichage

Pour afficher une valeur, on utilise la fonction `print`

```
print("bonjour")

a = 4
print("la valeur de a est : ", a)
```

Affichage

Pour afficher une valeur, on utilise la fonction `print`

```
print("bonjour")
```

```
a = 4
print("la valeur de a est : ", a)
```

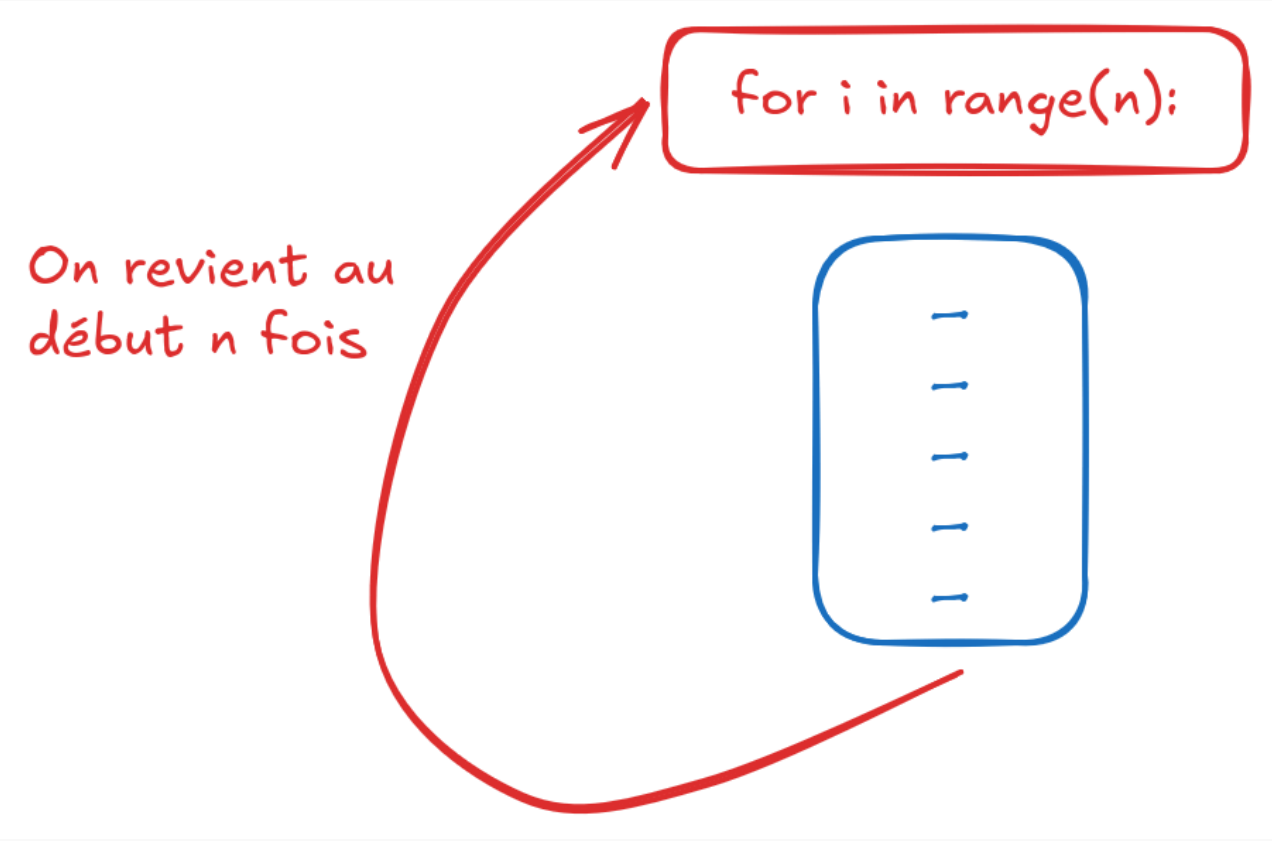
Un `print` affiche autant de valeur que l'on veut (séparées par des virgules) sur la même ligne, puis reviens à la ligne.

Conditions

On peut faire certaines opérations seulement si une condition est respectée :

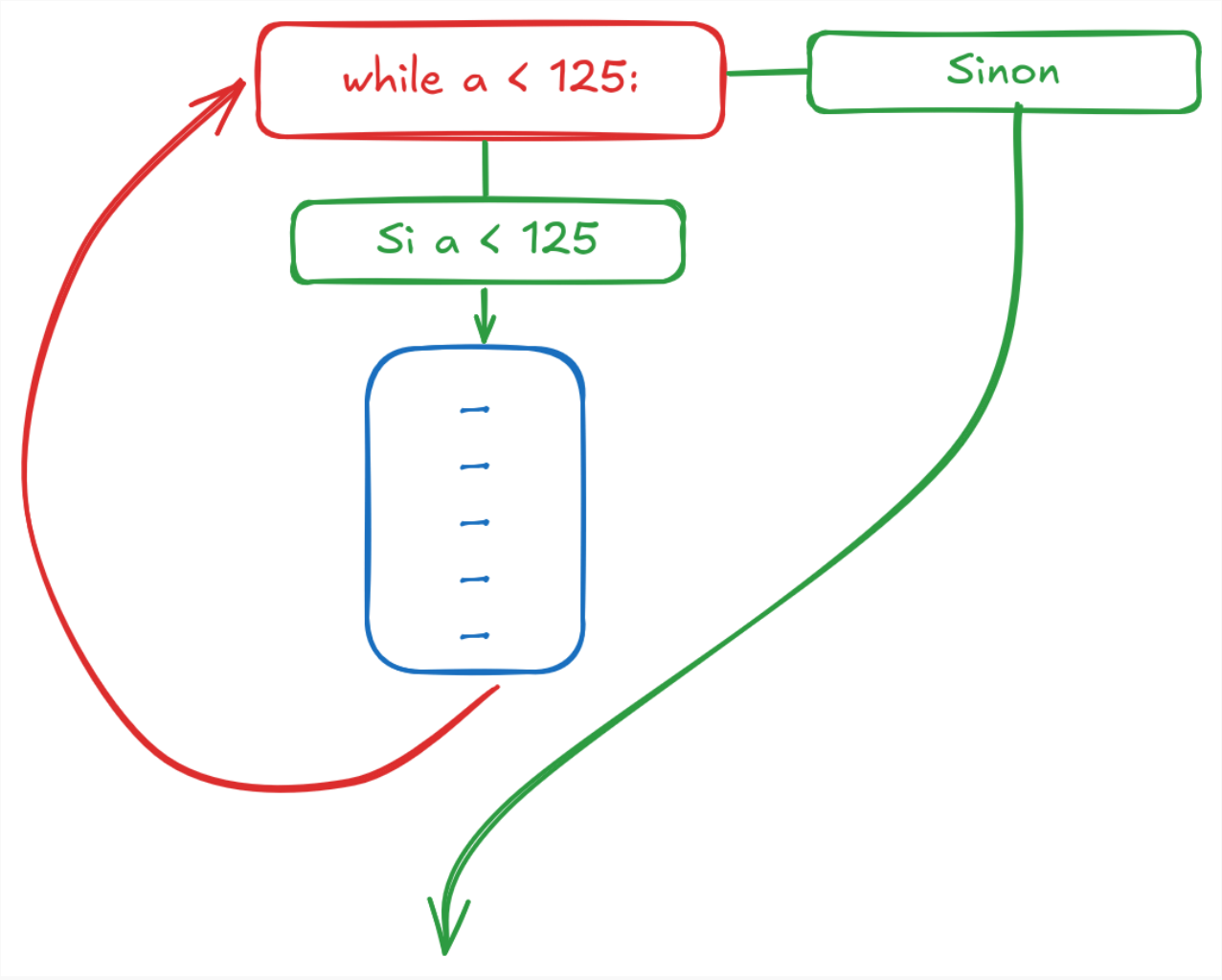
```
if a == 4:
  print("'a' a la bonne valeur")
elif a > 4:
  print("'a' est trop grand !")
else:
  print("'a' est trop petit !")
```

Boucles `for`



La fonction range peut aussi s'écrire `range(start, end, step)`

Boucles `while`



Laquelle utiliser ?

On utilise :

- Une boucle `for` quand on sait à l'avance combien de tour de boucles on va faire
- Une boucle `while` sinon

Les fonctions

Les fonctions sont des blocs de code réutilisable :

```
def tape(personne):
  #code pour taper une personne
  return point_de_degats

vie = tape("Mon voisin de table")
vie = tape("La prof")
vie = tape("La table")
```

Les fonctions

Les fonctions sont des blocs de code réutilisable :

```
def tape(personne):
  #code pour taper une personne
  return point_de_degats

vie = tape("Mon voisin de table")
vie = tape("La prof")
vie = tape("La table")
```

- `personne` est un argument de `tape`. On peut en mettre autant qu'on veut.
- La fonction retourne une valeur (avec `return`), à laquelle on peut ensuite accéder.

Documenter ses fonctions

On peut (et on doit) ajouter une chaine de documentation à ses fonctions.

```
def tape(personne):
  """
  Prend en argument une personne et la tape.
  Renvoie la quantité de dégats effectuée.
  """
  # code pour taper une personne
```

Documenter ses fonctions

On peut (et on doit) ajouter une chaine de documentation à ses fonctions.

```
def tape(personne):
  """
  Prend en argument une personne et la tape.
  Renvoie la quantité de dégats effectuée.
  """
  # code pour taper une personne
```

Ensuite, on peut accéder à cette chaine avec

```
help(tape)
```

Les listes

À quoi ça sert ?

En informatique, on est souvent amené à représenter des listes de valeurs. Par exemple, la liste des élèves d'une classe.

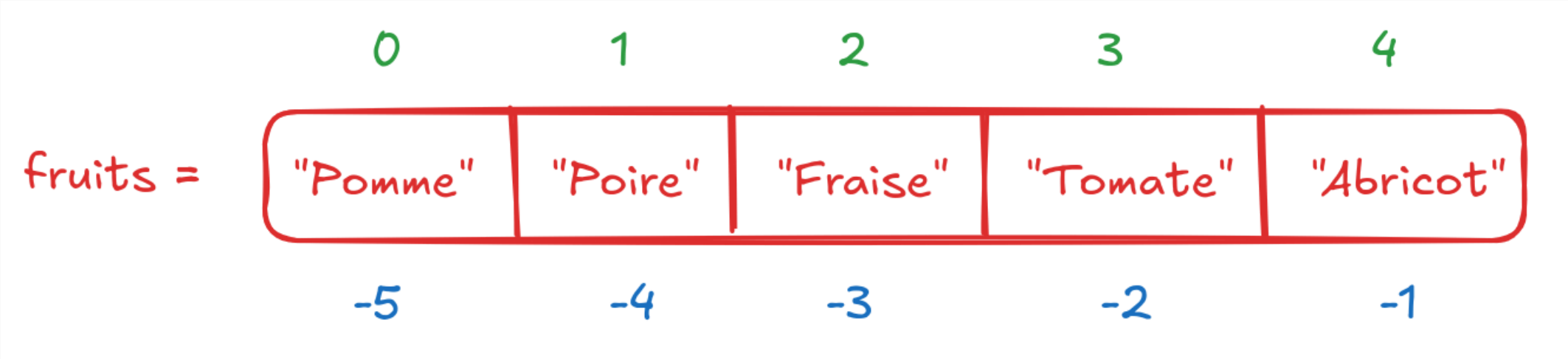
Le type `list` de `Python` sert justement à représenter de tels objets.

```
fruits = ["Pomme", "Poire", "Fraise", "Tomate"]
```

Comment on s'en sert ? L'indexation

Pour récupérer l'élément n° `i` d'une liste :

```
fruits[i]
fruits[0]
fruits[-1]
```



Comment on s'en sert ? Les `slice`

On peut aussi directement récupérer une sous-liste, avec ce qu'on appelle le slicing.

```
trois_premiers = fruits[:3]
trois_derniers = fruits[-3:]
milieu = fruits[2:4]
un_sur_deux = fruits[::2]
```

En fait, le slicing s'écrit `liste[start:end:step]`, avec exactment les mêmes arguments que la fonction `range` !

Comment on s'en sert ?

Pour connaitre la longueur d'une liste :

```
nombre_fruits = len(fruits)
```

Pour itérer sur les éléments d'une liste

```
for i in range(len(fruits)):
  print(fruits[i])
```

```
for i in range(len(fruits)):
  print(fruits[i])
```

Comment on s'en sert ?

Pour ajouter un élément à la fin d'une liste, on utilise la méthode `append`.

```
fruits.append("Pêche")
```

Une méthode est une fonction d'un objet, qui s'utilise avec `objet.methode(arguments ... )`.

Comment on s'en sert ?

Pour tester si un élément est présent dans une liste, on utilise le mot-clé `in` :

```
if "Tomate" in fruits:
  print("La tomate est bien un fruit !")
```

Modules et graphes

Le module numpy : import

Pour importer un module, on utilise

```
import numpy as np
```

au début du programme.

Le module numpy : tableau

Le module numpy permet de gérer des tableaux (array en anglais). Ce sont des listes de taille fixée.

Pour convertir une liste en tableau :

```
tableau = np.array(liste)
```

Le module numpy : matrices

Les tableau peuvent avoir plusieurs dimensions (pensez aux matrices). On peut alors accéder à l'élément (i, j) avec

```
element_ij = tableau[i, j]
```

On peut récupérer la taille du tableau avec :

```
tableau.shape
```

Le module matplotlib : import

On importe le module matplotlib avec :

```
import matplotlib.pyplot as pyplot
```

Le module matplotlib : une courbe de base

Pour afficher une courbe simple, on utilise les lignes suivantes :

```
plt.plot(X, Y, "b-")
plt.show()
```

Le “b-” est optionnel. Il est constitué

- de l’initial de la couleur souhaitée en anglais (b, r, g, etc)
- du type de point (+, *, ., -, etc)

Le module matplotlib : une courbe de base

Pour afficher une courbe simple, on utilise les lignes suivantes :

```
plt.plot(X, Y, "b-")
plt.show()
```

Le “b-” est optionnel. Il est constitué

- de l’initial de la couleur souhaitée en anglais (b, r, g, etc)
- du type de point (+, *, ., -, etc)

Si vous avez des probèmes de courbes qui se superposent, utilisez au début `plt.clf()`.