Informatique & Programmation

#3 Programmation python

par David Albert

GEMA
GRANDES ÉCOLES DES MÉTIERS D'AVENIR

2024

Table des matières

01

Entrées / Sorties

Entrées et Sorties

input

Pour lire des entrées saisies par un utilisateur dans la console, on utilisera la fonction `input`.

Exemple

```
name = input()
country = input("Quel est votre pays d'origine ?")
age = int(input("Quel est votre age ?"))
```

output

Pour afficher n'importe quel objet Python dans la console, on utilisera la fonction `print`.

Exemple

```
age = 35
print("Age")
print(age)
```

i

Il existe d'autres types d'entrées/sorties (fichier, réseau, base de données, ...). Nous les verrons plus tard.

02

Type de données

Types primitifs

Les booléens

Nom : `bool`

Valeurs : `True`, `False`

Opérateurs booléens

`and`, `or`, `not`

Les entiers

Nom : `int`

Valeurs : ..., -3, -2, -1, 0, 1, 2, 3, 4, ...

Opérateurs de comparaison

- égalité : `==`
- inégalité : `!=`
- infériorité : `<`, `<=`
- supériorité : `>`, `>=`

i

Les opérateurs de comparaisons retournent un booléen (`True` ou `False`).

Types primitifs

Les flottants

Nom : `float`

Valeurs : Permet de s'approcher d'une représentation des **nombres réels** (bien qu'incomplète).

Opérateurs de comparaison

- égalité : `==`
- inégalité : `!=`
- infériorité : `<`, `<=`
- supériorité : `>`, `>=`

i

Les opérateurs de comparaisons retournent un booléen (`True` ou `False`).

Types primitifs

Les chaînes de caractères

Nom : `string`

Définition : Une liste de caractères.

Exemples :
`"Brice"`, `"la vie est belle !"`, `"#0$£ù%&-"`

Caractères spéciaux

- Saut de ligne : `\n`
- Tabulation : `\t`

Formatage de texte

En Python, on peut simplement formater du texte en faisant précédé la chaîne de caractère par le caractère `f`

```
age_min = 12
warning_msg = f"interdit aux moins de {age_min} ans"
```

```
score = 0.33333333
print(f"score à 2 décimal={age_min:.2}")
```

Types primitifs

Les chaînes de caractères

Opérations classiques

Opérations	Exemples
assigner une variable	`my_str = "bidule"`
accès au 3ème caractère	`my_str[2]`
sous-chaîne	`my_str[:2]`, `my_str[1:]`, `my_str[1:2]`, `my_str[::2]`
longueur de la chaîne	`len(my_str)`
changer la casse	`my_str.capitalize()`, `my_str.upper()`, `my_str.lower()`
vérifier la casse	`my_str.islower()`, `my_str.isupper()`
concaténation	`"Hello " + "World"` (donnera `"Hello World"`)
répétition	`"la" * 5` (donnera `"lalalalala"`)

Types composites

Les listes

Nom : `list`

Définition : Une liste ordonnée de données.

Exemples :
`[0, 21, 13, 7, 100]`, `[]`, `[True, True, False]`, ...

Opérations classiques

- assigner une variable : `l = []`
- accès au 3ème caractère : `l[2]`
- accès au dernier caractère : `l[-1]`
- sous-listes : `l[:2]`, `l[1:]`, `l[1:2]`, `l[::2]`
- ajout d'un élément à la fin : `l.append(3)`
- supprimer du 3ème élément : `l.remove(2)`
- tri d'une liste : `sorted(l)`

Types composites

Les dictionnaires

Nom : `dict`

Définition : Un dictionnaire est une structure de données qui assimile des clés à des valeurs.

Exemples :
`{"nom" : "Fred", "age" : 20 }`, `{}`, `{True : "eat", False : [0, 1, 2]}`, ...

Opérateurs de bases

- assigner une variable : `d = {}`
- accès à une valeur (depuis sa clé) : `d[key]`
- ajout d'un élément à la fin : `d[key] = value`
- supprimer du 3ème élément : `del d[key]`
- accès aux clés : `d.keys()`
- accès aux valeurs : `d.values()`
- accès aux couples (clé, valeurs) : `d.items()`

03

Variables

Qu'est-ce qu'une variable ?

Définition - Variable

En informatique, les variables sont des symboles qui associent un nom (**l'identifiant**) à une **valeur**. Dans la plupart des langages, **les variables peuvent changer de valeur au cours du temps**.

De plus, **les variables ont un type** de valeur.

En python, la déclaration d'une variable se fait avec l'opérateur d'allocation `=`

Exemples :

```python
prenom = "Jonathan"          # variable de type chaîne de caractères (str)

age = 23                     # variable de type entier (int)

moyenne = 10.8               # variable de type flottant (float)

notes = [16, 12, 13, 9]      # variable de type liste d'entiers (list[int])
```

Convention de nommage

Nommage des variables

Par convention en python, un nom de variable commence par une lettre minuscule puis les différents mots sont séparés par un tiret bas (tiret du 8).

Exemples : `distance`, `distance_max`, `consigne_courante`, `etat_bouton_gauche_souris`

Nommage des constantes

Par convention en python, un nom de constante est en majuscule.

Exemple : `MAX_PLAYERS`, `HEIGHT`, `WIDTH`, ...

Mots réservés

Les mots réservés sont les mots prédéfinis du langage python.
Ils ne peuvent pas être réutilisés pour des identifiants.

Exemples : `for`, `while`, `if`, `return`, `None`, ...

Portée des variables

La portée (scope) d'un identifiant (variables, fonctions, ...) est l'étendue au sein de laquelle cet identifiant est lié.

En python, la portée peut être globale (en dehors de tout bloc d'indentation) ou locale (au bloc courant).

Portée des variables (globale / locale)

```
var1 = 10 # var1 est globale

def foo(var2): # var2 est locale
    var3 = 30 # var3 est locale
    if (var3 > 0):
        var4 = 40
        print(var1, var2, var3, var4) # 10, 20, 30, 40

    print(var1, var2, var3, var4) # 10, 20, 30, Erreur

foo(20)
print(var1, var2, var3, var4) # 10, Erreur, Erreur, Erreur
```

04

Commentaires

Commenter son code

Pourquoi ?
Les commentaires permettent d'expliquer succinctement certains portions de notre code.

Comment ?
Pour commenter une ligne ou une fin de ligne en python, on utilise le symbole **#**

Exemple

```
# Cette ligne est un commentaire
nom = "Franc" # ici aussi c'est un commentaire
```

i

Les commentaires doivent aider à comprendre le code. On ne répète pas simplement une ligne.

✅ **Bon commentaire**

```
x = x + 4 # increase the border width
```

⬆️ Ce commentaire aide à la compréhension du code.

❌ **Mauvais commentaire**

```
x = x + 4 # increase x by 4
```

⬆️ Ce commentaire est inutile.