Chap 7. Python 3 : Les variables

Livre p 3 – Chap 1 Arithmétique, variables, instructions

1. Types de données

a. Introduction

En python, contrairement à la plupart des autres langages, il n'est pas nécessaire de définir un type de variable avant de l'utiliser, mais ce type existe et a son importance !
Remarque : Pour connaitre le type d'une expression ou d'une variable, écrire : *type*(expression)

b. Noms de variables

On utilise habituellement les lettres minuscules (non accentuées) mais les majuscules sont autorisées.
Les chiffres sont autorisés, sauf pour le premier caractère. Exemple : point1
On peut également utiliser le symbole « tiret bas du 8 » (« underscore » en anglais). Exemple : x_max
Attention : Il existe une trentaine de mots réservés que l'on ne peut pas utiliser comme nom de variable.

c. Les différents types

- *int* Nombre entier de taille quelconque (contrairement aux autres langages !)
- *float* Nombre à virgule flottante dans $\{-\text{inf}\}\cup[-10^{308} ; -10^{-323}]\cup\{0.0\}\cup[10^{-323} ; 10^{308}]\cup\{\text{inf}\}$
- *complex* Nombre complexe (attention, *i* est noté *j* et un coefficient doit être présent devant *j*)
- *bool* Booléen, uniquement deux valeurs possibles : *True* ou *False* (attention à la majuscule)
- *str* Chaîne de caractères Unicode entre guillemets ″…″ ou ′…′
- *tuple* Liste non modifiable de données de mêmes types ou variés, entre parenthèses (…)
- *list* Liste modifiable de données de mêmes types ou variés, entre crochets […]
- *set* Ensemble non ordonné de données de mêmes types ou variés entre accolades {…}
- *dict* Dictionnaire, entre accolades {…} Exemple : {′maths′ : 7, ′phys′ : 6}

d. Conversions

Une expression peut être converti d'un type de données vers un autre. Exemple : *int*(″12″) ou *str*(5).
Remarque : Pour passer d'une base quelconque vers le système décimal, écrire : *int*(chaine, *base* = entier)
Autres commandes de conversion :

- *bin*(entier) Pour convertir un entier du système décimal vers le système binaire.
- *oct*(entier) Pour convertir un entier du système décimal vers le système octal.
- *hex*(entier) Pour convertir un entier du système décimal vers le système hexadécimal.
- *ord*(caractère) Pour obtenir le codage Unicode d'un caractère placé entre guillemets.
- *chr*(entier) Pour obtenir le caractère codé par le codage Unicode donné.
- *eval*(expression) Permet d'évaluer une expression numérique placée entre guillemets.
- chaine.*split*() Permet de convertir une chaîne de caractères en une liste en coupant par défaut à chaque espace, mais on peut préciser un autre sécateur…
- expression.*join*(liste) Permet de convertir une liste en un chaîne de caractère en intercalant une expression entre chaque valeur de la liste (attention, il faut une liste de *str*)

2. Opérateurs

a. Opérateurs booléens

- a *or* b Si a est égal à *False*, renvoie b, sinon renvoie a.
- a *and* b Si a est égal à *False* renvoie a, sinon renvoie b
- not *a* Renvoie *False* si a est égal à *True*, sinon revoie *True*.

Remarque : 0, 0.0, ″″, (), [], {} valent tous *False*

b. Opérateurs logiques sur les bits

- & L'opérateur « et » bit à bit. Exemple : 25 & 41 renvoie 9.
- | L'opérateur « ou » bit à bit. Exemple : 25 | 41 renvoie 57.
- ^ L'opérateur « ou exclusif » bit à bit. Exemple : 25 ^ 41 renvoie 48.
- ~ L'opérateur « non » bit à bit. Exemple : ~25 renvoie –26. (complément à 2)
- << Un décalage de n bits vers la gauche. Exemple : 25<<2 renvoie 100.
- >> Un décalage de n bits vers la droite. Exemple : 25>>2 renvoie 6.

($25 = 11001_{b2}$, $41 = 101001_{b2}$, $9 = 1001_{b2}$, $57 = 111001_{b2}$, $48 = 110000_{b2}$, $100 = 1100100_{b2}$, $100 = 110_{b2}$)

c. Opérateurs de comparaison

Les opérateurs de comparaison agissent sur tous les types de données, et renvoient un booléen.

- < inférieur strictement Exemple : ″chat″ < ″chien″ renvoie *True* (ordre alphabétique)
- > supérieur strictement Exemple : 12 > 105 renvoie *False*
- <= inférieur ou égal Exemple : ″12″ <= ″105″ renvoie *False* (ordre alphabétique !)
- >= supérieur ou égal Exemple : 1083 >= 105 renvoie *True*
- == test d'égalité Exemple : 1 + 1 == 2 renvoie *True*
- != différent de Exemple : ″12″ != 12 renvoie *True*
- in est dans Exemple : 6 in [2, 4, 6, 8] renvoie *True*

d. Opérateurs mathématiques

- + addition pour les nombres, mais concaténation pour les chaînes de caractères et les listes.
  Exemple : 2.3 + 4 renvoie 6.3 mais ″2.3″ + ″4″ renvoie ″2.34″
- – soustraction pour les nombres uniquement.
- * multiplication pour les nombres, mais répétition pour les chaînes de caractères et les listes.
  Exemple : 2.3 * 4 renvoie 9.2 mais [2, 3] * 4 renvoie [2, 3, 2, 3, 2, 3, 2, 3]
- / division décimale pour les nombres uniquement, renvoie un *float*.
- ** exposant pour les nombres uniquement Remarque : 2**0.5 permet d'obtenir $\sqrt{2}$
- // division entière pour les entiers uniquement, renvoie un *int*.
  Exemple : 20 / 3 renvoie 6.666666666666667 mais 20 // 3 renvoie 6
- % reste de la division euclidienne pour les entiers uniquement, renvoie un *int*. (modulo)
  Exemple : 20 % 3 renvoie 2 car $20 = 3 \times 6 + 2$

e. Affectation

- = prend la valeur

Remarque : On peut aussi faire des affectations multiples : a = b = 12 ou parallèles : a, b = 12, 15.
Astuce : On peut combiner une affectation et une opération. Exemple : a += 2 signifie a = a + 2

3. Entrée / Sortie

a. Entrée au clavier

input() Permet la saisie d'une chaîne de caractères au clavier. Exemple : a = *input*(″Votre nom ?″)
Attention : Si l'on souhaite saisir des nombres, il faut changer le format de la saisie
Exemples : a = *int*(*input*(″Entrer un entier″)) ou : a = *float*(*input*(″Entrer un décimal″))

b. Sortie dans la console

print() Affiche une donnée ou plusieurs arguments séparés par une virgule.
Exemple : *print*(″Contenu de la variable x :″, x)
Remarque : La virgule sera traduite par un espace à l'affichage, mais on peut le modifier avec *sep* = ″…″.
Attention : Le saut de ligne est automatique à la fin, mais on peut aussi le changer avec *end* = ″…″.