Nom :	Prénom :	Classe :

NSI 1re — Conversion de types + Commande input()

Élément du programme : Langages et programmation → Constructions élémentaires.

Pourquoi changer de type ?

La conversion de types (ou « casting ») en programmation, et plus particulièrement en Python, est une opération qui vise à transformer une valeur d'un type de données à un autre, selon les besoins du programme.

La conversion de types existe parce que les ordinateurs gèrent des valeurs de diverses natures (entiers, chaînes, flottants, etc.), et il est fréquent de devoir passer de l'un à l'autre (par exemple, convertir une *chaîne de caractères* entrée par un utilisateur en *nombre* pour effectuer des calculs).

Par exemple, sans conversion de type, on ne peut pas additionner un entier `10` et une chaîne de caractères `"20"` sans provoquer une erreur :

```
1  >>> a = 10
2  >>> b = '20'
3  >>> c = a + b
4          ~~^~~
5  TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Explication de code :

- Ligne 1 : affectation à une variable `a` de **l'entier** `10`.
- Ligne 2 : affectation à une variable `b` de la **chaîne** `'20'`.
- Ligne 3 : affectation à une variable `c` du résultat de l'addition des valeurs contenues dans `a` et `b`.
- Ligne 4 et 5 : Python affiche une erreur signifiant qu'il ne peut pas additionner un entier (int) avec une chaîne (str).

Les développeuses et développeurs que nous sommes utilisant des langages typés (Python, C, Java, etc.) ont toujours recours à la conversion de types, car nous devons jongler avec des données issues de sources variées.

Comment changer de type ?

Il est possible de changer un type de donnée en un autre, grâce aux commandes suivantes :

Conversion en ...	Commande	Conversion en ...	Commande
Entier	`int()`	Chaîne de caractères	`str()`
Flottant	`float()`	Booléen	`bool()`

Afficher le type d'une variable (rappel)

Bref rappel sur **l'affichage du type** d'une variable :

```
1   >>> a = '123'       # Assignation d'une chaîne à une variable « a »
2   >>> print(a)        # Affichage de la valeur stockée dans « a »
3   123                 # À première vue, c'est un entier ?!

4   >>> print(type(a))  # Affichage du type de « a »
5   <class 'str'>       # Ah ! C'est bien une chaîne
```

Dans le code commenté ci-dessus, nous assignons une chaîne à une variable (ligne 1).

Mais lorsque nous demandons à Python d'en afficher son contenu avec `print()` (ligne2)
rien ne nous permet de savoir si c'est une chaîne, ou un entier...

À moins d'utiliser la — désormais célèbre — commande `print( type( nom_de_la_variable ) )`

Changer le type d'une variable

Voici maintenant un exemple de *changement de type* dans lequel on assigne une **chaîne**, qui est ensuite convertie en **entier**.

```
1   >>> a = '123'       # Assignation d'une chaîne à une variable « a »

2   >>> b = int(a)      # Assignation à une variable « b » de la conversion de « a » en entier
3   >>> print(b)        # Affichage de la valeur stockée dans « b »
4   123                 # Est-ce vraiment un entier ?

5   >>> print(type(b))  # Affichage du type de « b »
6   <class 'int'>       # Oui, c'est bel et bien un entier
```

Dans le code commenté ci-dessus, nous assignons une chaîne à une variable `a`.

Puis nous assignons à une variable `b` la conversion de `a` en entier.

Tableau d'exemples

Prenons différentes valeurs (0, 1, 3.14, '12' et True) et observons les changements qui opèrent à travers les commandes de changement de types :

	0	1	3.14	'12'	True
int()	0	1	3	12	1
float()	0.0	1.0	3.14	12.0	1.0
str()	'0'	'1'	'3.14'	'12'	'True'
bool()	False	True	True	True	True

➤ Exercice 1 (à vous de jouer)

Pour chacun des codes ci-dessous, complétez la colonne "changement de type", permettant d'obtenir la valeur souhaitée. *La première ligne est donnée en guise d'exemple.*

Code	Valeur souhaitée	Changement de type (à compléter)
`a = 3.0`	`3`	`a = int(a)`
`a = 3`	`3.0`	
`a = 45.12`	`'45.12'`	
`a = 10 / 2`	`5`	
`a = 60 // 6`	`10.0`	
`a = 13 % 3`	`'1'`	

Commande input()

La commande `input()` en Python est une commande qui permet à un programme de demander une saisie à l'utilisateur, puis de récupérer cette valeur comme une **chaîne de caractères**.

```
1  >>> a = input('Votre âge svp ? ')
2  Votre âge svp ? 23
3  >>> print(a)
4  23
5  >>> print(type(a))
6  <class 'str'>
```

Explication de code :

Ligne	Explication
1	On assigne à la variable `a` la saisie utilisateur, en affichant la phrase "Votre âge svp ? "
2	La phrase "Votre âge svp ? " est affichée, et l'utilisateur a ici saisi le nombre `23`.
3	On affiche le contenu de la variable `a`.
4	Python affiche la valeur `23` — Mais difficile d'en évaluer précisément le type 🤔
5	On affiche le type de la variable `a`.
6	Python nous précise qu'il s'agit d'un `str`, donc d'une chaîne de caractères !

La commande input() ne peut-elle envoyer qu'une chaîne de caractères ?

En un mot : oui !

Peu importe la phrase affichée avec `input()`, cette commande retournera toujours une chaîne :

```
>>> a = input('Nombre entre 1 et 6 svp ? ')
Nombre entre 1 et 6 svp ? 5
>>> print(type(a))
<class 'str'>

>>> b = input('Combien font 1 + 1 ? ')
Combien font 1 + 1 ? 2
>>> print(type(b))
<class 'str'>

>>> pseudo = input('Quel est votre pseudo ? ')
Quel est votre pseudo ? John83
>>> print(type(pseudo))
<class 'str'>
```

Dans l'exemple ci-dessus, les variables `a`, `b` et `pseudo` ont été définies par des commandes `input()`.

Dans chaque cas, la valeur retournée est un `str`.

Et si le message affiché par input() contient des guillemets ?

Les « messages » affichés par `input()` sont des **chaînes de caractère**. Donc les mêmes règles s'appliquent[1].

Il est possible d'échapper des guillemets simples :

```
r = input('J\'ai un petit "faible" pour Python, et toi ? ')
```

Il est également possible d'échapper des guillemets doubles :

```
r = input("J'ai un petit \"faible\" pour Python, et toi ? ")
```

1 Voir, à ce propos, le cours « Type de donnée - entier, flottant, booléen, chaîne de caractères ».