Types et portée des variables

Table des matières

Contexte

Durée : 2h

Environnement de travail : Repl.it

Pré-requis : Aucun

[cf. 7jMkUSQk]

Les variables permettent de manipuler les données au sein d'un programme. Chaque variable est ainsi un emplacement mémoire utilisé pour stocker une valeur qui pourra subir différentes opérations. Au delà des opérations de base que vous connaissez, il est possible de réaliser des opérations plus complexes, comme le changement de type d'une variable. De plus une variable n'est pas forcément utilisable pendant la totalité de l'exécution du programme. Selon la manière d'écrire le code source, il sera possible de n'utiliser une variable que durant une partie du programme, et de libérer ensuite l'espace mémoire utilisé. Les variables nous réservent donc encore de nombreux secrets.

Types de données

[cf. EDcOg9QU]

Objectif

- Découvrir les différents types de données.

Mise en situation

Pour manipuler et traiter des données, on associe un type à chacune d'entre elles. Un ordinateur ne traite que des informations binaires, toutes les données sont donc converties en suite de 0 et de 1. Comment dans ce cas différencier une lettre d'un nombre ou un booléen d'une suite de caractères ? Grâce aux types de données. Celui-ci va permettre de définir de quelle manière doivent être réalisées les opérations sur les données. En effet ajouter 2 nombres ne se fait pas de la même manière que la concaténation de 2 chaînes de caractères.

Type de donnée *Définition*

Une donnée peut être un nombre, un caractère, une suite de caractères, etc.

Le type d'une donnée, c'est donc la catégorie que le programme va associer à celle-ci. Le type détermine la place occupée en mémoire par la donnée, les traitements possibles sur celle-ci ou le comportement d'un opérateur.

Les types principaux *Définition*

En JavaScript les types principaux sont :

- les nombres (entiers ou flottants) `number`,
- les chaînes de caractères `string`
- et les booléens `bool`.

En Python les types principaux sont :

- les nombres entiers `int`,
- les nombres flottants (c'est à dire les nombres à virgule) `float`,
- les chaînes de caractères `str`
- et les booléens `bool`.

Connaître le type *Méthode*

En JavaScript la fonction `typeof()` permet de connaître le type d'une variable.

En Python il s'agit de la fonction `type()`.

Exemple

```javascript
/** Javascript */
let word = 'Hello'
let value = 42
console.log(typeof(word))
console.log(typeof(value))
```

Ce programme affiche :

```
string
number

```

Comportement de l'opérateur + selon le type de données

Exemple

L'opérateur + peut être utilisé avec des nombres ou avec des chaînes de caractères, son comportement dépend justement du type de données manipulées.

```javascript
/** JavaScript */
const number1 = 1
const number2 = 4
const letter1 = '1'
const letter2 = '4'
console.log(number1 + number2)
console.log(letter1 + letter2)
```

```python
"""Python."""
number1 = 1
number2 = 4
letter1 = '1'
letter2 = '4'
print(number1 + number2)
print(letter1 + letter2)
```

Ces deux programmes retournent :

```
5
14
```

Les variables `number1` et `number2` ont été additionnées avec l'opérateur + alors que les variables `letter1` et `letter2` ont été concaténées avec le même opérateur, parce que les données `number1` et `number2` ne sont pas de même type que les données `letter1` et `letter2` (type `number` et type `string` en JavaScript).

Avec des données de type booléen, l'opérateur + correspond à l'opération logique ***ou***.

typeof

Complément

Le mot-clé `typeof` peut être utilisé comme une fonction (comme nous l'avons vu ici), ou bien comme opérateur unaire, ainsi les deux instructions d'affichage suivantes sont équivalentes :

```javascript
/** Javascript */
let word = 'Hello'
console.log(typeof(word))
console.log(typeof word)
```

D'autres types de données *Complément*

Il existe d'autres type de données comme les tableaux ou les objets.

Un tableau est une collection de variables type quelconque, rangées les unes à la suite des autres dans la mémoire.

Les tableaux sont en général utilisés pour stocker et traiter des données de même nature.

Dans l'exemple suivant le tableau `temp` contient des températures, les températures sont séparées par des virgules.

```python
"""Python."""
temp = [21, 24, 18, 19]
```

```javascript
/** JavaScript */
const temp = [21, 24, 18, 19]
```

À retenir

Le type d'une donnée correspond à la catégorie de cette donnée (nombre, chaîne de caractères, etc.), ce type indique au programme comment il doit manipuler et traiter celle-ci.

[cf. GC2S0L9d]

Exercice : Appliquer la notion

Question 1

Afficher le résultat de l'opération + entre une variable `nbStrawberries` de type `number` (valeur *4*) et une variable `type` de type `string` (valeur *gariguettes*).

Question 2

Afficher le résultat de l'opération + entre une variable `boolean` de type `bool` (valeur *true*) une variable `nb` de type `number` (valeur *16*).

Question 3

Afficher le résultat de l'opération + entre une variable `boolean` de type `bool` (valeur *true*) une variable `word` de type `string` (valeur *magic*).

Conversion de type (cast)

[cf. YcD2nROg]

Objectif

- Apprendre à convertir des données d'un type vers un autre.

Mise en situation

Comment changer le type d'une donnée ?

Chaque donnée a un type qui détermine quels traitements sont possibles et comment un même opérateur agit sur sur cette donnée. Cependant on peut avoir besoin de changer le type d'une donnée, par exemple si l'on veut effectuer des traitements incompatibles avec le type initial. Imaginons que nous avons un nombre que l'on souhaite ajouter dans une chaîne de caractères, pour l'intégrer à une phrase. Il n'est pas possible d'additionner un nombre avec du texte, mais en convertissant ce nombre en une chaîne de caractères, il devient possible de concaténer nos 2 chaînes de caractères.

Conversion en chaîne de caractères *Définition*

En JavaScript la méthode `String()` permet de convertir des nombres ou des booléens en chaînes de caractères.

En Python il s'agit de la méthode `str()`.

Exemple

```python
"""Python."""
number = 25
number = str(number)
```

```javascript
/** JavaScript */
let number = 25
number = String(number)
```

Conversion en nombre *Définition*

En JavaScript la méthode `Number()` permet de convertir des chaînes de caractères ou des booléens en nombres.

En Python il s'agit de la méthode `int()` si on veut faire une conversion vers un nombre entier et `float()` si on veut faire une conversion vers un nombre flottant.

Exemple

```python
"""Python."""
word_1 = '3.26'
word_1 = float(word_1)
word_2 = '426'
word_2 = int(word_2)
```

```javascript
/** JavaScript */
let word = '3.26'
word = Number(word)
```

Remarque

Les méthodes de conversion de type fonctionnent de la même manière avec des booléens.

La conversion d'un booléen `true` vers un nombre renvoie 1, celle d'un booléen `false` vers un nombre renvoie 0.

[cf. Zwkbne8S]

Exercice : Appliquer la notion

Question 1

Déclarer une variable `nbBirds` initialisée à 325 et utiliser la conversion de type pour afficher le nombre des centaines de la variable (c'est à dire 3).

Indice :

Il faut convertir la variable en chaîne de caractères.

Indice :

Le numéro de centaine est le premier caractère de 325. On peut accéder à un caractère d'une chaîne avec la syntaxe `variable[indice]`, les indices commençant à 0.

Question 2

La programme suivant n'affiche pas la somme du nombre d'oranges et du nombre de bananes.

Modifiez-le afin que ce soit bien le cas, sans changer les variables `nbOranges` et `nbBananas`.

```
1 /** JavaScript */
2 const nbOranges = '47'
3 const nbBananas = '31'
4 const nbFruits = nbOranges + nbBananas
5 console.log(nbFruits)
```

Déclaration préalable des variables

[cf. rsCVKmjm]

Objectifs

- Savoir déclarer une variable avec ou sans affectation ;
- Savoir à quoi correspond la valeur `undefined`.

Mise en situation

Lors du développement d'un programme, le développeur est en charge de définir les variables à utiliser, les déclarer ainsi que leurs types, et leur affecter des valeurs. L'étape de déclaration d'une variable est primordiale : il n'est pas possible d'utiliser une variable si elle n'a pas été déclarée au préalable. En effet, puisque une variable permet de réserver de l'espace mémoire pour la valeur à stocker, l'ordinateur ne pourra rien stocker si la variable n'a pas été déclarée. De plus dans certains langages, c'est lors de la déclaration que le type de la variable est déterminée.

Déclarer une variable — *Définition*

Déclarer une variable c'est indiquer au compilateur ou à l'interpréteur qu'il doit **réserver un emplacement mémoire** pour que le programmeur puisse y stocker une donnée.

En Python il est obligatoire d'initialiser une variable lors de sa déclaration. Mais d'autres langages comme le JavaScript permettent de déclarer une variable sans l'initialiser.

Cependant il est préférable d'initialiser une variable lors de sa déclaration afin de s'assurer de la valeur que contient la variable.

Exemple

```python
"""Python."""
apples = 10
```

```javascript
/** JavaScript */
let apples
apples = 10
```

Attention

On ne peut pas utiliser une variable qui n'a pas été préalablement déclarée.

De plus, une constante doit être initialisée lors de sa déclaration.

Remarque

JavaScript associe un type à une variable en fonction de la valeur associée, en l'absence de valeur associée (ce qui est le cas lors d'une déclaration sans initialisation), la variable ne peut prendre que le type `undefined`. Ici, dans le programme en JavaScript, lorsqu'on déclare la variable `apples` sans l'initialiser elle prend automatiquement le type `undefined`.

À retenir

Pour pouvoir utiliser une variable, il faut la déclarer au préalable.

Lors de sa déclaration, il est parfois indispensable de l'initialiser (en Python par exemple) mais même si ce n'est pas obligatoire c'est fortement recommandé car beaucoup plus sûr.

[cf. kRj6LXgD]

Exercice : Appliquer la notion

Question 1

Déclarer une variable `potatoes` sans l'initialiser, afficher sa valeur et afficher son type.

Question 2

Déclarer une variable `potatoes` sans l'initialiser, afficher sa valeur et afficher son type, affecter la valeur 15 à `potatoes`, afficher sa valeur et son type.

Indice :

Il faut compléter le programme précédent.

Portée des variables

[cf. PtHkXr21]

Objectif

- Comprendre la notion de portée locale et de bloc de définition.

Mise en situation

Un programme est séparé en blocs afin d'être plus lisible. Par exemple si on veut écrire un programme qui affiche une suite de nombres puis qui calcule leur somme, on peut séparer le programme en deux parties : deux blocs, qui ont chacun leurs variables. En effet chaque variable ne sera utilisable que dans le bloc de code où elle a été définie. On appelle portée d'une variable la zone de code dans laquelle une variable sera définie et utilisable.

Portée d'une variable *Définition*

Lorsqu'on définit une variable elle est associée au bloc où elle se trouve et n'est visible que dans celui-ci. On parle de **portée locale**.

Cela signifie qu'il est **impossible** d'afficher ou d'utiliser cette variable dans un autre bloc : elle n'existe tout simplement pas.

Si une variable est déclarée dans le bloc principal (c'est à dire en dehors de toute boucle ou fonction) elle est visible dans toute la partie du programme située après sa déclaration.

Exemple

Dans le programme suivant la variable `numberApple` est visible dans tout le programme y compris dans le bloc `if`.

En revanche la variable `enoughApple` est déclarée dans le bloc `if`, donc visible uniquement dans celui-ci.

La dernière instruction génère donc une erreur.

```
1 /** JavaScript */
2 const numberApple = 30
3 console.log(numberApple)
4 if(numberApple === 30)
5 {
6   let enoughApple = true
7   console.log(enoughApple)
8 }
9 console.log(enoughApple)
10
```

Remarque

La portée des variables fonctionne de la même manière avec d'autres boucles ou dans des fonctions.

Une variable définie dans une fonction n'est visible que dans celle-ci.

À retenir

En fonction du bloc où sont déclarées les variables, elles n'ont pas la même portée et ne sont donc pas visibles et utilisables dans la même partie du programme.

[cf. SPaFTUOD]

Exercice : Appliquer la notion

Programme exemple

```
/** JavaScript */
const numberApple = 30
console.log(numberApple)
if(numberApple === 30)
{
  let enoughApple = true
  console.log(enoughApple)
}
console.log(enoughApple)

```

Question 1

Réécrire le programme exemple afin qu'il ne génère pas d'erreur.

Indice :

Il faut modifier la portée de la variable `enoughApple`.

Question 2

Voici une modification du programme précédent. Qu'affiche le programme ? Pourquoi ?

```
/** JavaScript */
const numberApple = 30
let enoughApple = false
console.log(numberApple)
if(numberApple === 30)
{
  let enoughApple = true
  console.log(enoughApple)
}
console.log(enoughApple)
```

Variables globales

[cf. 0ijqYNFR]

Objectifs

- Savoir ce que sont et à quoi servent les variables globales ;
- Adopter les bonnes pratiques.

Mise en situation

Les variables ont généralement une portée locale, c'est à dire qu'elles ne sont accessibles que dans le bloc de code où elles ont été définies. C'est la plupart du temps largement suffisant, mais il peut arriver d'avoir besoin d'une variable soit accessible dans la totalité du programme. Par exemple une variable qui contiendrait une information de configuration régulièrement utilisé à des endroits très différents du code. Pour cela, il est possible d'utiliser une variable de portée globale, qui ne sera donc pas limitée de la même manière qu'une variable locale.

Variable globale en JavaScript


Définition

En JavaScript, on définit habituellement une variable avec `let nameV = value`.

En remplaçant le mot-clé `let` par le mot-clé `var` on change la portée de la variable.

La variable aura une portée globale et sera donc visible dans toute la partie du programme située après sa définition.

En Python le mot-clé `global`, suivi du nom de la variable indique que la variable à utiliser a été préalablement définie dans un autre bloc du programme et que c'est cette variable qu'il faut modifier.


Exemple

```
1 /** JavaScript : programme erroné */
2 const numberApple = 30
3 console.log(numberApple)
4 if(numberApple === 30)
5 {
6   let enoughApple = true
7   console.log(enoughApple)
8 }
9 console.log(enoughApple)
10
```

L'exécution de ce programme renvoie une erreur car la dernière instruction fait référence à une variable qui n'existe pas : `enoughApple` est locale au bloc `if`.

```
1 /** JavaScript  : programme corrigé */
2 const numberApple = 30
3 console.log(numberApple)
4 if(numberApple === 30)
5 {
6   var enoughApple = true
```

```
  console.log(enoughApple)
}
console.log(enoughApple)

```

Si on remplace `let` par `var` dans la définition de `enoughApple`, il n'y a plus d'erreur car la variable est visible hors de la boucle `if`.

Ce code fonctionne, mais c'est une **mauvaise pratique** !

À consommer avec modération *Attention*

Les variables globales sont source de nombreux problèmes.

En effet, une variable déclarée globale pourra être modifiée dans n'importe quelle autre partie du programme, ce qui peut provoquer des erreurs difficiles à détecter, comme l'écrasement inattendu de la valeur d'origine (on appelle cela un effet de bord).

Il est recommandé de ne jamais utiliser de variables globales par défaut, et de les utiliser avec parcimonie uniquement dans des cas bien identifiés.

Complément

Si on veut inverser la valeur de deux variables avec une fonction, on peut utiliser des variables globales (mais ce n'est pas une bonne façon de faire).

```python
"""Python."""
fruit_1 = ' pear '
fruit_2 = ' pineapple '

def inversion():
    global fruit_1
    global fruit_2
    temp = fruit_2
    fruit_2 = fruit_1
    fruit_1 = temp

print(fruit_1 + fruit_2)
inversion()
print(fruit_1 + fruit_2)
```

Le programme affiche « pear pineapple » puis « pineapple pear ».

À retenir

Il est possible de créer des variables dont la portée s'étend au delà de leur bloc de définition, avec le mot-clé `var` en JavaScript et le mot-clé `global` en Python, il s'agit des variables globales. Cependant il est recommandé de les utiliser avec prudence car cela peut provoquer des résultats inattendus.

[cf. 1S0DOMJl]

Exercice : Appliquer la notion

Question 1

Que se passe-t-il si on définit dans une boucle `if` une variable **locale** portant le même nom qu'une variable **globale** déjà déclarée préalablement ?

Expliquez à partir de l'exécution du code suivant :

```
/** JavaScript */
var numberApple = 30
var enoughApple = false
console.log(numberApple)
if(numberApple === 30)
{
  let enoughApple = true
  console.log('A-t-on assez pommes ? (dans le "if")', enoughApple)
}
console.log('A-t-on assez pommes ? (en dehors du if)', enoughApple)
```

Question 2

Faut-il obligatoirement initialiser une variable globale (comme pour les constantes), ou est-il possible de les déclarer sans les initialiser (comme pour les déclarations utilisant `let`) ? Essayez.

Question 3

Qu'affiche le programme suivant ?

```
/** JavaScript */
console.log(flowers)
var flowers = 25
```

Essentiel

[cf. N9Fo7DPl]

Nous avons vu que la déclaration d'une variable détermine deux choses importantes : son type et sa portée. En effet lorsque l'on déclare puis affecte une valeur à une variable, celle-ci se retrouve avec un type, qui définit les opérations qu'il est possible de réaliser. Cependant il est possible de modifier le type d'une variable lorsque cela a du sens.

La portée de la variable est aussi définie à la déclaration. Par défaut une variable est locale est accessible uniquement dans le bloc de code où elle se trouve. Mais il est possible de définir une variable comme étant globale, pour qu'elle soit accessible de partout.

Quiz

Exercice 1 : Quiz - Culture

Exercice

Qu'affiche le programme suivant ?

```
/** JavaScript */
const number = 3.14
const conversion = String(number)
console.log(conversion[2])
```

- ○ undefined
- ○ .
- ○ 1

Exercice

Quelle est la portée d'une variable déclarée dans une boucle ?

- ○ locale
- ○ globale

Exercice

L'usage de variables globales... :

- ☐ permet d'avoir accès à la variable dans tous les blocs du programme
- ☐ peut provoquer des effets de bord
- ☐ est déconseillé en général

Exercice 5 : Quiz - Méthode

Exercice

Quelle est la meilleure façon de déclarer une variable `nbDogs` et de lui affecter la valeur 3 ?

- ○ var nbDogs = 3
- ○ var nbDogs
  nbDogs = 3
- ○ let nbDogs =3
- ○ let nbDogs
  nbDogs = 3

Exercice

Quelle instruction permet de déclarer une variable contenant la caractère 1 ?

- `let letter = 1`
- `let character = 1`
- `let num = '1'`

Exercice 8 : Quiz - Code

Exercice

Comment convertir `nbBananas`, une chaîne de caractères valant *'3415'*, en un nombre, en JavaScript ?

- `nbBananas = Number(nbBananas)`
- `nbBananas.Number()`
- `int(nbBananas)`
- `nbBananas.int()`

Exercice

Soit le programme :

```
1 /** JavaScript */
2 let f = false
3 console.log('Boolean : ', f)
4 console.log('String : ', String(f))
5 console.log('Number : ', Number(f))
```

Compléter le résultat de son exécution :

Boolean :

String : '

'

Number :

Exercice

Qu'affiche le programme suivant ?

```
1 /** JavaScript */
2 const numberApple = 30
3 let enoughApple = false
4 if(numberApple === 30)
5 {
6   let enoughApple = true
7 }
8 console.log(enoughApple)
```

- true
- false

Exercice

Qu'affiche le programme suivant ?

```
/** JavaScript */
const bool1 = false
const bool2 = false
const res = bool1 + bool2
console.log(res)
```

- falsefalse
- 0

Exercice

Qu'affiche le programme suivant ?

```
/** JavaScript */
const word = 'bonjour'
const number = 126458
console.log(word[2])
console.log(number[2])
```

- n
  6
- o
  2
- n
  undefined