Les nombres

Table des matières

Contexte

Durée : 2h

Environnement de travail : Repl.it

Pré-requis : Aucun

[cf. 2h83Lpb4]

Un type de variable qui peut sembler assez basique est celui des nombres. Mais en réalité, les nombres ne se cantonnent pas uniquement aux opérations mathématiques classiques comme la somme ou la multiplication. La plupart des langages proposent en effet des fonctions permettant de réaliser des opérations plus complexes comme des arrondis ou des puissances. Il est même possible d'utiliser les nombres comme booléens dans nos conditions. Enfin, nous allons nous intéresser à la manière de stocker en mémoire des nombres à virgule, car ce n'est pas aussi trivial que l'on peut le penser.

Les types de nombres

[cf. aH9ixIqn]

Objectif

- Connaître les différents types de nombres en programmation.

Mise en situation

En mathématique, on différencie les nombres entiers, les nombres décimaux et les nombres réels. En informatique cette distinction est également possible et se fait via les types de nombres.

Type de nombre  ***Définition***

Un type de nombre est un type de donnée (comme les chaînes et les booléens) avec plusieurs catégories : les nombres entiers, les nombres décimaux, les nombres réels. La différenciation de ces types dépend du langage de programmation utilisé.

Les nombres ayant une partie décimale sont écrits avec un point à la place de la virgule : `2.2` au lieu de `2,2`.

Les types de nombres en Python

En Python, il y a trois types de nombres : les nombres entiers `int`, les nombres décimaux (ou flottants) `float` et les nombres complexes `complex`.

Le nombres flottants servent à représenter les nombres décimaux et à approximer les nombres réels.

 ***Exemple***

Il n'est pas possible de stocker la valeur exacte de Pi (qui est un nombre réel avec une infinité de décimales), quel que soit le type utilisé, mais il est possible de l'approximer à l'aide d'un nombre flottant (environ 3.14159).

Les types de nombres en JavaScript

En JavaScript il n'y a qu'un type de nombre : `number`, qui stocke les nombres entiers et les nombres flottants sur 8 octets (64 bits).

 ***Exemple***

```
1 /** JavaScript */
2 console.log(typeof (3))
3 console.log(typeof (3.1))
```

```
1 "number"
2 "number"
```

```python
"""Python."""
print(type(3))
print(type(3.1))

```

```
<class 'int'>
<class 'float'>
```

L'encodage des types de nombres

Complément

Les types de nombres peuvent influencer le comportement de certains opérateurs et déterminent la place en mémoire nécessaire pour stocker un nombre.

En Python, un entier `int` est stocké sur 4 octets (32 bits) tandis qu'un nombre flottant `float` est stocké sur 8 octets (64 bits).

Les nombres entiers stockés sur 32 bits sont compris entre -2 147 483 648 et 2 147 483 647, c'est-à-dire entre -2^{31} et $2^{31}-1$. Si un programme contient un nombre plus grand, l'interpréteur Python le convertit en `long` c'est à dire en entier long, qui est un autre type de `int`.

À retenir

Il existe différent types de nombres en informatique pour représenter les types de nombres qui existent en mathématiques. Ces types varient en fonction du langage utilisé et déterminent quelle place est nécessaire pour stocker un nombre en mémoire.

[cf. HaiU05AD]

Exercice : Appliquer la notion

Question 1

Écrire un programme qui affecte la valeur `4,1` à une variable `note` et affiche la variable.

Question 2

Afficher les types des nombres 123456, -4 et 3.14159.

Les opérations de base

[cf. B6IA1kMo]

Objectif

- Connaître les opérations mathématiques de base pour la programmation.

Mise en situation

En mathématiques, il existe des règles précises pour écrire des opérations, en informatique aussi.

Les opérations de base *Définition*

On considère quatre opérations de base : l'addition **+**, la soustraction **-**, la multiplication ***** et la division **/**.

En informatique, le symbole **=** ne signifie pas « est égal à » mais « prend la valeur de ».

Exemple

```python
"""Python."""
x = 3 * 2.5
```

```javascript
/** JavaScript */
const x = 3 * 2.5
```

Les exemples ci-dessus **affectent** le résultat de l'opération `3 * 2.5` à la variable `x`.

Les priorités de calcul

Le priorités de calcul sont les mêmes en mathématiques et en informatique : la multiplication et la division sont plus **prioritaires** que l'addition et la soustraction.

Si deux opérations ont le même niveau de priorité, l'opération située **la plus à gauche** sera effectuée en premier.

Un opération écrite entre **parenthèses** est prioritaire sur toutes les autres opérations.

La **puissance** est prioritaire sur la multiplication et la division.

Ces règles sont les mêmes quel que soit le langage utilisé.

Exemple

```python
"""Python."""
x = 3 * (4 + 1) / 5 - 3
print(x)
```

```javascript
/** JavaScript */
const x = 3 * (4 + 1) / 5 - 3
console.log(x)
```

```
0
```

Les deux exemples ci-dessus renvoient 0 car ils effectuent d'abord l'addition de 4 et de 1, puis la multiplication par 3, puis la division par 5 et enfin la soustraction de 3.

Espaces entres les opérateurs en JavaScript *Attention*

Une bonne pratique en JavaScript consiste à séparer chaque opérateur de ses **opérandes** à l'aide d'un espace.

Au lieu d'écrire `x=3*2.5`, on écrit `x = 3 * 2.5`, ce qui est plus lisible.

À retenir

Les règles de calcul et de priorité des opérations en informatique sont identiques ou très proches de celles en mathématiques.

Il est important de séparer chaque opérateur avec un espace pour rendre le code plus lisible.

[cf. uCZHrET6]

Exercice : Appliquer la notion

Question 1

Écrire un programme qui calcule le nombre de repas mangés en une semaine à raison de 3 repas par jour.

Indice :

Il est préférable d'utiliser une multiplication, en utilisant le symbole *****.

Question 2

Écrire un programme qui affiche le nombre de chocolats de Noël achetés par 3 familles avec 1, 3 et 4 enfants, à raison de 7 chocolats par enfants, en utilisant le moins d'opérateurs possibles.

Indice :

Il pourrait être tentant de multiplier le nombre d'enfants de la première famille avec le nombre de chocolats, et de faire de même pour les autres familles.

L'utilisation de **parenthèses** permet de **factoriser** ces opérations, en calculant d'abord le nombre total d'enfants.

Les opérations avancées

[cf. npVSDcZV]

Objectif

- Apprendre à utiliser des opérations plus avancées que les opérations de base.

Mise en situation

Nous avons vu comment faire les quatre opérations de base, mais il existe beaucoup d'autres opérations mathématiques plus complexes qui peuvent être facilement utilisées.

Les opérations avancées — *Définition*

On considère comme opérations avancées les calculs de puissance, de racine, de valeur absolue, de modulo, d'arrondis, de division entière et de tirage aléatoire.

Il en existe bien plus, comme les opérations trigonométriques.

Les opérations avancées en JavaScript — *Syntaxe*

- Pour calculer la puissance d'un nombre, par exemple 2^3 :

```
x = Math.pow(2, 3)
```

- Pour calculer une division entière, par exemple, la division entière de 5 par 3 qui vaut 1 :

```
x = Math.trunc(5 / 3)
```

- Pour calculer le reste d'une division entière (le modulo), par exemple le modulo de 5 par 3 qui vaut 2 :

```
x = 5 % 3
```

- Pour calculer l'arrondi au plus proche d'un nombre, par exemple de 2.6 qui vaut 3 :

```
x = Math.round(2.6)
```

- Pour calculer la racine carrée d'un nombre, par exemple la racine de 25 :

```
x = Math.sqrt(25)
```

- Pour calculer l'arrondi à l'entier inférieur d'un nombre, par exemple de 2.6 qui vaut 2 :

```
x = Math.floor(2.6)
```

- Pour calculer l'arrondi à l'entier supérieur d'un nombre, par exemple de 2.6 qui vaut 3 :

```
x = Math.ceil(2.6)
```

Les opérations avancées en JavaScript

Exemple

```js
/** JavaScript */
console.log('2 puissance 10 vaut: ' + Math.pow(2, 10))
console.log('Avec 5 paquets de bonbons pour 3 personnes on a ' + Math.floor(5 / 3) + ' paquet(s) par personne')
console.log('Il restera ' + 5 % 3 + ' paquet(s) de bonbons')
console.log('Avec une note de 2.6/20 à mon examen j\'ai eu environ ' + Math.round(2.6))
console.log('la racine carree de 25 est: ' + Math.sqrt(25))
console.log('le plus petit entier superieur à 2.6 est :' + Math.ceil(2.6))
```

Autres opérations en JavaScript

Complément

- Pour calculer la valeur absolue d'un nombre, par exemple la valeur absolue de -3 qui vaut 3 :

  `x = Math.abs(-3)`

- Pour tirer aléatoirement un nombre entre 0 et 1 exclu :

  `x = Math.random()`

- Pour tirer aléatoirement un nombre entier situé entre deux bornes, par exemple entre 1 et 5 :

  `x = Math.ceil(Math.random() * 5)`

Les opérations avancées en Python

Syntaxe

- Pour calculer la puissance d'un nombre, par exemple 2^3 :

  `x = 2 ** 3`

- Pour calculer une division entière, par exemple, la division entière de 5 par 3 qui vaut 1 :

  `x =5 // 3`

- Pour calculer le reste d'une division entière (ou **modulo**), par exemple le reste de la division entière de 5 par 3 qui vaut 2 :

  `x = 5 % 3`

- Pour calculer l'arrondi au plus proche d'un nombre, par exemple de 2.6 qui vaut 3 :

  `x = round(2.6)`

Les opérations avancées en Python

Exemple

```python
"""Python."""
print('2 puissance 10 vaut: ',2**10)
print('Avec 5 paquets de bonbons pour 3 personnes on a ',5 // 3, ' paquet(s) par personne')
print('Il restera ',5 % 3,' paquet(s) de bonbons')
print('Avec une note de 2.6/20 à mon examen j\'ai eu environ', round(2.6))

```

Les opérations en Python nécessitant la bibliothèque math

Syntaxe

Pour les opérations suivantes il faut importer une bibliothèque d'opérations mathématiques avec l'instruction `from math import *`

- Pour calculer la racine carrée d'un nombre, par exemple la racine de 25 :

  `x = sqrt(25)`

- Pour calculer l'arrondi à l'entier inférieur d'un nombre, par exemple de 2.6 qui vaut 2 :

  `x = floor(2.6)`

- Pour calculer l'arrondi à l'entier supérieur d'un nombre, par exemple de 2.6 qui vaut 3 :

  `x = ceil(2.6)`

Les opérations en Python nécessitant la bibliothèque math *Exemple*

```
"""Python."""
from math import *
print('la racine carree de 25 est: ',sqrt(25))
print('le plus grand entier inférieur à 2.6 est :',floor(2.6))
print('le plus petit entier superieur à 2.6 est :',ceil(2.6))

```

Autres opérations en Python *Complément*

Il existe beaucoup d'autres fonctions mathématiques très utiles.

Pour calculer la valeur absolue d'un nombre, par exemple la valeur absolue de -3 qui vaut 3 : `x = abs(-3)`

Pour les opérations suivantes il faut importer une bibliothèque d'opérations mathématiques avec l'instruction : `from random import *`

Pour tirer aléatoirement un nombre entre 0 et 1 exclu : `x = random()`

Pour tirer aléatoirement un nombre entier situé entre deux bornes, par exemple entre 1 et 5 : `x = randint(, 5)`

À retenir

Il est possible de faire facilement des opérations avancées en Python et en JavaScript. Les fonctions utilisées portent en général le même nom mais sont appelées différemment.

Pour utiliser certaines fonctions il faut importer la bibliothèque `math` ou la bibliothèque `random` en Python.

[cf. 3UrKfxp8]

Exercice : Appliquer la notion

Pour calculer la longueur de l'hypoténuse (c'est à dire le côté le plus long) d'un triangle rectangle, on utilise le théorème de Pythagore.

Ce théorème dit que la longueur de l'hypoténuse correspond à la racine carrée de la somme des longueurs au carrés des deux autres côtés.

Théorème de Pythagore[1]

Question 1

Compléter ce programme pour qu'il calcule et affiche la longueur de l'hypoténuse d'un triangle rectangle dont les deux autres côtés font respectivement 4 et 3 cm.

Quelle est la longueur de l'hypoténuse ?

```
/** JavaScript */
const length1 = 3
const length2 = 4
```

Pour calculer le volume d'un cube on multiplie la longueur de son côté par elle-même pour trouver l'aire de la base du cube.

Enfin, on multiplie cette aire par la longueur du côté pour trouver le volume du cube.

Question 2

Calculer et afficher le volume d'un cube de côté 5cm.

Indice :

Il faut multiplier la longueur du côté trois fois par elle-même. Si cette opération peut être réalisée uniquement à l'aide de multiplications, il pourrait être plus pertinent de calculer directement la puissance 3 de ce nombre...

Question 3

Écrire un programme qui affiche le reste de la division entière de 42 par 2.

Indice :

L'opération permettant de calculer le reste dans une division s'appelle le modulo.

[1] https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_de_Pythagore

Flottants et précision

[cf. EzGsketg]

Objectifs

- Comprendre le mode de stockage des nombres flottants en informatique ;
- Comprendre la notion de précision.

Mise en situation

En informatique on sait que toutes les données sont stockées en binaire. Pour les nombres décimaux, donc avec une virgule, on peut se demander comment stocker correctement la valeur sachant que la virgule n'est pas un nombre que l'on peut stocker : c'est un caractère. Une première approche pourrait être de simplement stocker la partie entière et la partie décimale séparément, mais ce n'est pas comme cela que ça fonctionne. C'est en réalité un peu plus complexe que cela, et en informatique on parle de nombres flottants, ou nombres à virgule flottante.

L'encodage des flottants

Les nombres flottants sont stockés sur 64 bits de façon à ce que le premier bit en partant de la gauche (on parle de bit de poids fort) détermine le signe : si ce bit vaut **0 le nombre est positif**, si ce bit vaut **1 le nombre est négatif**. Les onze bits suivants permettent d'encoder l'**exposant** et les cinquante deux bits restants permettent d'encoder la **mantisse**.

Les nombres sont dits flottants parce que la place de la virgule n'est pas fixe. Contrairement à ce que pourrait dicter l'intuition, il ne s'agit **pas** d'écrire les nombres avec un bit de signe, onze bits pour la partie entière et les cinquante deux bits restants pour la partie décimale.

Format de représentation des flottants en double précision

Définition

Les nombres flottants sont calculés ainsi, en **binaire** :

$$\texttt{nbFlottant} = \texttt{signe} * 2^{\texttt{exposant}} * \texttt{mantisse}$$

La place de la virgule dans un nombre flottant dépend donc de son exposant.

En décimal ***Exemple***

Il est plus simple d'utiliser le système décimal pour comprendre. Le nombre `-1.2345` sera représenté avec les données suivantes :

- Signe : `négatif`
- Exposant : `-4`
- Mantisse : `12345`

Le calcul donnera 10^{-4}, soit `0.0001`, multiplié par `12345`, soit `1.2345`, puis passé en négatif.

On constate que le changement de l'exposant, par `-5` par exemple, donnera le nombre `0.12345`. La place de la virgule est donc flottante, et dépend de la valeur de l'exposant.

Déclarer un flottant et l'utiliser

Exemple

```python
"""Pyhton."""
nb_float = 0.1
print(nb_float)
nb_float *= 0.1
print(nb_float)
nb_float += 1
print(nb_float)
```

```javascript
/** JavaScript */
let nbFloat = 0.1
console.log(nbFloat)
nbFloat *= 0.1
console.log(nbFloat)
nbFloat += 1
console.log(nbFloat)
```

En Python comme en JavaScript on déclare et on manipule les nombres flottants comme les autres nombres.

Imprécisions

Attention

Tous les nombres ne sont pas parfaitement *encodables* avec cette méthode. Les nombres qui correspondent à une somme de puissance de deux sont écrits de façon exacte, les autres le sont de façon approchées, ce qui dans certains cas peut poser des problèmes de calcul.

Problèmes d'approximation

Exemple

En mathématiques, 0.1 + 0.2 = 0.3. La réponse à la question « est ce que 0.2 ajouté à 0.1 vaut 0.3 ? » est « oui ».

De même, 0.25 + 0.25 = 0.5, donc la réponse à la question « est ce que 0.25 ajouté à 0.25 vaut 0.5 ? » est « oui ».

Cependant, à cause des problèmes d'approximation des nombres flottants, ce n'est **pas** le cas en informatique.

```python
"""Python."""
print(0.1 + 0.2 == 0.3)
print(0.25 + 0.25 == 0.5)
```

```javascript
/** JavaScript */
console.log(0.1 + 0.2 === 0.3)
console.log(0.25 + 0.25 === 0.5)
```

Ces deux programmes affichent `false` puis `true`, parce que 0.1 **ne peut pas être encodé correctement**. Ainsi, quand les programmes additionnent 0.1 et 0.2, ils additionnent en réalité un nombre très proche de 0.1 et 0.2, donc le résultat obtenu n'est pas égal à 0.3 et le test est faux.

En revanche, 0.25 est encodé correctement donc la deuxième addition donne un résultat exact et le test est vrai.

Eviter les problèmes d'approximation *Conseil*

Pour éviter les problèmes d'approximation lors de calculs ou de comparaisons avec des flottants il faut travailler avec mathjs[1], une bibliothèque conçue pour gérer les problèmes d'approximations des nombres flottants en JavaScript.

toPrecision() *Complément*

Soit `x = 0.3333` :

- `x.toPrecision(2)` renvoie `0.33`.
- `x.toPrecision(2)` est de type chaîne de caractère, `x.toPrecision(2).substr(2)` renvoie `33`.

À retenir

Les nombres flottants sont stockés sur 64 bits, ils sont dits flottants parce que la place de la virgule dépend de l'exposant.

[cf. 9bOdNGQm]

[1] https://mathjs.org/

Exercice : Appliquer la notion

Question 1

Déclarer une variable `average` initialisée à 14.8 et calculer l'écart entre la moyenne et la note d'un étudiant qui a eu 9.1.

Question 2

Écrire un programme qui affiche le résultat de la multiplication de 0.1 par 0.1. Que peut-on constater ?

Les booléens

[cf. uvsXFcon]

Objectifs

- Comprendre ce que sont les booléens ;
- Savoir combiner des booléens avec l'algèbre booléenne.

Mise en situation

En programmation, les nombres ne servent pas qu'à représenter des nombres.

En effet, on les retrouve aussi dans la manière dont sont gérés d'autres types de variables, comme les **booléens**.

Le concept de booléen est issu de l'**algèbre booléenne**, et représentent les **valeurs de vérité** utilisées dans les raisonnements logiques.

En pratique, les booléens stockent 2 états possibles : **vrai** ou **faux**, et sont très utilisés pour évaluer les conditions en programmation.

Les booléens se basent très fortement sur les nombres.

Les booléens — *Définition*

Les booléens sont un type de variable pouvant prendre deux valeurs : vrai (*true*) ou faux (*false*). Ils représentent des valeurs de vérité, par exemple si une comparaison est vraie ou fausse.

Exemple

```
reality = false
console.log(reality)
```

L'opération booléenne ET — *Définition*

L'opération **ET** entre deux variables booléennes est vraie si les deux variables sont vraies, sinon elle est fausse.

Exemple

```
"""Python."""
underage = True
student = False
print(underage and student)
```

```
/** JavaScript */
const underage = true
const student = false
console.log(underage && student)
```

Les deux programmes affichent la valeur `false`.

L'opération booléenne OU

Définition

L'opération **OU** (non exclusive) entre deux variables booléennes est vraie si au mois l'une des deux variables est vraie, sinon elle est fausse.

Exemple

```python
"""Python."""
cinema_opened = True
bowling_opened = False
possible_party = cinema_opened or bowling_opened
print(possible_party)
```

```javascript
/** JavaScript */
const cinemaOpened = true
const bowlingOpened = false
const possibleParty = cinemaOpened || bowlingOpened
console.log(possibleParty)
```

Les deux programmes affichent la valeur `true`.

L'opération booléenne NON

Définition

L'opération **NON** sur un booléen renvoie l'inverse de ce booléen

Exemple

```python
"""Python."""
rain = True
sun = False
print('il fait beau:', not(rain))
print('il ne fait pas beau: ', not(sun))
```

```javascript
/** JavaScript */
const rain = true
const sun = false
console.log('il fait beau:' + !rain)
console.log('il ne fait pas beau: ' + !sun)
```

Les deux programmes affichent la valeur `false` puis la valeur `true`.

Implémentation en Python

Complément

En Python, le type `bool` est en fait une sous-classe du type `int` et les valeurs `True` et `False` sont des versions particulières de 1 et de 0. Ainsi le nombre 0 peut être considéré comme un booléen `False`, de même qu'une chaîne de caractères vide. À l'inverse, tous les nombres différents de 0 et toutes les chaînes de caractères qui ne sont pas vides peuvent être considérées comme des booléen `True`.

Implémentation en JavaScript

Complément

En JavaScript, une chaîne de caractères vide, une valeur `undefined` et le nombre 0 sont assimilables à des booléens `false` mais ils n'en sont pas (ils sont appelés `falsy` pour les différencier du type booléen). Tous les nombres différents de 0 et toutes les chaînes de caractères non-vides sont appelés `truthy`.

Pour obtenir un booléen à partir d'un entier ou d'une chaîne, on utilise la fonction `Boolean()` :

- `Boolean('')` renvoie la valeur booléenne `false`, tout comme `Boolean(0)` par exemple.
- `Boolean('Hello')` renvoie la valeur booléenne `true`, tout comme `Boolean(42)` par exemple.

À retenir

Les booléens sont un type particulier d'entiers valant 0 ou 1. Il est possible de faire des opérations **ET**, **OU** et **NON** avec.

[cf. aaRBmVlo]

Exercice : Appliquer la notion

Question 1

Quel est le résultat de l'opération **ET** entre un booléen vrai et un booléen faux en JavaScript ?

Indice :

L'opération **ET** s'écrit `&&` en JavaScript.

Question 2

Afficher l'inverse d'un booléen faux en JavaScript.

Indice :

L'opération inverse s'écrit `!` en JavaScript.

Question 3

Quel est le résultat d'une opération **OU** entre une chaîne de caractères vide et le nombre `0`, en JavaScript ?

Indice :

On utilise la fonction `Boolean` pour convertir une chaîne ou un entier en booléen.

Indice :

L'opération **OU** s'écrit `||` en JavaScript.

Essentiel

[cf. U2Olko7f]

Les nombres sont les variables de base que l'on manipule en informatique. Derrière cette simplicité se cache tout de même certaines subtilités, comme les nombres flottants ou l'utilisation des nombres pour les booléens. L'informatique ayant une histoire commune avec les mathématiques, la plupart des langages de programmation offrent un vaste panels d'opérations qu'il est possible de réaliser sur des nombres. Cela va de la simple addition, au calcul d'arrondi, en passant par la génération de nombres aléatoires.

Quiz

Exercice 1 : Quiz - Culture

Exercice

En JavaScript, quelle phrase est vraie concernant la valeur `undefined` ?

- ○ C'est la même chose que `false`
- ○ C'est assimilable à `false`
- ○ C'est la même chose que `null`

Exercice

Si `bool1` est vrai et `bool2` est vrai, que vaut `bool1` OU `bool2` ?

Exercice

Quels nombres peuvent être considéré comme des `truthy` ?

- ☐ Les nombres entiers
- ☐ Les nombres strictement positifs
- ☐ Les nombres strictement négatifs
- ☐ Tous les nombres

Exercice

Sur combien de bits est encodé un nombre entier en JavaScript ?

Exercice 6 : Quiz - Méthode

Exercice

Quel opérateur permet de calculer le reste d'une division entière en JavaScript ?

- ○ /
- ○ //
- ○ %
- ○ modulo()

Exercice

Quelle est la meilleure façon d'affecter à une variable le résultat de la multiplication de 2 par 3 en JavaScript ?

- `const x = 2 * 3`
- `const x= 2 * 3`
- `const x = 2*3`
- `const x=2*3`

Exercice

Quel est le résultat de l'opération **6 * (4 + 5) / (3 -1)** en JavaScript ?

- 11
- 27
- 14.5

Exercice

Dans un nombre flottant en JavaScript, sur combien de bits est encodée la mantisse ?

- 8 bits
- 32 bits
- 64 bits
- 52 bits

Exercice

Quel est le signe d'un flottant dont le bit de signe vaut 1 ?

- Positif
- Négatif

Exercice

Le nombre 0.125 s'écrit aussi avec la représentation 2^{-3}. En JavaScript, la valeur 0.125 sera représentée de façon :

- Exacte
- Approchée

Exercice 13 : Quiz - Code

Exercice

Qu'affiche le programme JavaScript suivant ?

```
/** JavaScript */
console.log(sqrt(25))
```

- ○ 5
- ○ 25
- ○ error

Exercice

Que renvoie la combinaison de booléens suivante ?

```
/** JavaScript */
const bool1 = false
const bool2 = true
const bool3 = false
console.log((bool1 && bool2) || bool3)
```

- ○ true
- ○ false

Exercice

Qu'affiche le programme suivant ?

```
/** JavaScript */
let x = 1
console.log(x = 3)
```

Exercice

Quel est le type de la variable `const x = 3000000` en JavaScript ?

Crédits des ressources

Format de représentation des flottants en double précision p. 14

http://creativecommons.org/licenses/zero/3.0/fr/, Par GMjeanmatt — Travail personnel, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=7318466