# Les entrées et les sorties

# Table des matières

# Contexte

**Durée** : 2h

**Environnement de travail** : Repl.it

**Pré-requis** : Aucun

[cf. UywMiBd8]

**Entrées et sorties en programmation**

Il est très courant pour un programme d'interagir avec un utilisateur ou des capteurs. Que ce soit pour demander de saisir un texte, afficher un résultat, ou récupérer la position de la souris, tout les échanges d'un programme se font avec ce que l'on appelle des périphériques. On parle de gestion des entrées et sorties, que l'on retrouve souvent sous le terme de **I/O**, pour *Input/Output* en anglais.

Ce module consiste à présenter les utilisations possibles des entrées et sorties textuelles, la conversion des entrées et leur usage en combinant les structures itératives.

# Sortie textuelle

[cf. oB6RNGih]

**Objectif**

- Savoir utiliser la sortie textuelle depuis un programme.

**Mise en situation**

L'écriture de texte vers l'écran, et plus particulièrement la console d'exécution d'un programme, est sans doute le premier exemple de sortie que l'on apprend dans un langage de programmation. Il s'agit par exemple du célèbre *Hello World*, qui consiste en l'affichage à l'écran d'une simple chaîne de caractères. C'est un exemple souvent utilisé pour découvrir un langage car il permet de voir la gestion d'une écriture sur la **sortie standard**. De plus c'est une fonction qui est quasi-systématiquement intégrée de base dans un langage, et qui est donc simple à utiliser.

### Sorties

*Définition*

En programmation, les sorties désignent les information qui sont envoyées du programme vers un **périphérique** de sortie comme un écran, une imprimante, un disque mémoire, etc. Ce sont des flux de données envoyés depuis l'intérieur de l'unité du traitement vers l'extérieur (d'où le terme de sortie, ou ***output***).

On appelle aussi ce flux de sortie la **sortie standard**, qui désigne le plus souvent l'écran.

*Fondamental*

**Écrire sur la sortie standard** revient à dire au programme d'afficher quelque chose à l'écran.

### Écrire en sortie

*Syntaxe*

Les langages mettent souvent à disposition des fonctions dont le nom est proche de l'anglais pour permettre d'écrire des phrases, des nombres, ou toute sorte de valeurs.

- En JavaScript, on utilise la fonction `console.log` avec le contenu à afficher en paramètre : `console.log('....')`
- En Python, la fonction `print` s'utilise de la même manière : `print('...')`

*Exemple*

```python
"""Python: affiche "Un message très important"."""
message = 'Un message très important'
print(message)
```

```javascript
/** JavaScript: affiche "Un message très important" */
const message = 'Un message très important'
console.log(message)
```

### Écrire les nombres

*Remarque*

Les fonctions d'affichage acceptent les caractères numériques : on peut simplement passer des nombres en paramètres ; ils seront transformés en chaînes de caractères vers la sortie.

Il est d'ailleurs souvent possible de passer plusieurs paramètres à ces fonctions pour afficher plusieurs éléments sur la sortie.

*Exemple*

```python
"""Python: affiche "1 2 3 Soleil"."""
print(1, 2, 3, 'Soleil')
```

```javascript
/** JavaScript: affiche "1 2 3 Soleil" */
console.log(1, 2, 3, 'Soleil')
```

### Formatage de chaîne

*Complément*

Il est utile de pouvoir injecter la valeur de variables dans une chaîne lors d'un affichage, comme le nom d'un utilisateur sur la page d'accueil d'un site internet. Il existe des syntaxes spéciales permettant de réaliser cette opération.

Notez l'utilisation du caractère `f` en Python, qui permet d'indiquer que la chaîne contient des valeurs à remplacer.

```python
name = 'Bobby'
# Affiche 'Bonjour Bobby!'
print(f'Bonjour {name}!')
```

Notez l'utilisation des caractères ` en JavaScript, qui permettent d'indiquer que la chaîne contient des valeurs à remplacer.

```javascript
const name = 'Bobby'
// Affiche 'Bonjour Bobby!'
console.log(`Bonjour ${name}!`)
```

### À retenir

Pour afficher des informations à l'écran, on utilise le flux de sortie, ou **sortie standard**, qui permet de les envoyer vers le périphérique écran.

[cf. 6RCytkLm]

# Exercice : Appliquer la notion

## Question 1

On veut réaliser un test de culture générale auquel les joueurs répondent sur papier. La première question demande la valeur de **PI**.

Stocker la valeur *« Question 1 : Quelle est la valeur de PI ? »* dans une constante `question1`, puis l'afficher.

## Question 2

Il faut ensuite donner la réponse. Afficher la bonne réponse en utilisant une constante **PI** avec un message *« Réponse à la question 1 : 3.14 »*.

# Entrée textuelle


[cf. 6ugAOaLG]

**Objectifs**

- Savoir utiliser l'entrée textuelle ;
- Savoir récupérer des informations de l'utilisateur.

**Mise en situation**

Avec la communication du programme vers l'utilisateur, il est indispensable que l'inverse soit possible : la communication de l'utilisateur vers le programme. Les deux se complètent et permettent les interactions entre le programme et l'utilisateur en leur permettant d'échanger des informations.

**Entrées** — *Définition*

En programmation, les entrées désignent les informations envoyées par un périphérique d'entrée comme le clavier vers le programme. Ce sont des flux de données envoyés depuis l'extérieur vers l'unité de traitement (d'où le terme d'entrée, ou ***input***).

On appelle aussi ce flux de données l'**entrée standard**, qui désigne souvent le flux qui transporte les entrées du clavier vers le processeur.

*Fondamental*

**Lire l'entrée standard** revient alors à dire au programme de récupérer les données qui ont été écrites par l'utilisateur.

**Lire les entrées** — *Syntaxe*

La lecture de données en entrée se fait à l'aide de fonction pré-existantes. Elles permettent de récupérer du texte écrit au clavier par l'utilisateur.

En JavaScript, on utilise la fonction `prompt` qui prend en paramètre le texte à afficher : `prompt('...')`.

En Python, la fonction `input` demande d'entrer du texte directement dans la console et prend comme paramètre le texte à affiche comme indication : `input('...')`.

*Exemple*

```python
"""Python: demande d'entrer son nom dans la console."""
input('Entrer votre nom : ')
```

```javascript
/** JavaScript: ouvre une fenêtre qui demande d'entrer son nom */
prompt('Entrer votre nom : ')
```

### Récupérer l'entrée

***Attention***

Telle quelle, une instruction de lecture n'est pas bien utile puisque le résultat est perdu. Il faut veiller à le récupérer pour le stocker dans une variable afin de le conserver et de l'utiliser plus tard.

On peut notamment l'utiliser pour vérifier certaines conditions, comme le fait qu'il ne soit pas vide, ou qu'il corresponde à ce qui était demandé.

***Exemple***

```python
"""Python: demande de réponde par oui ou par non."""
response = input('Êtes-vous étudiant: (oui/non)')

if response == 'oui' or response == 'non':
  print('Je note cela')
```

```javascript
/** JavaScript: ouvre une fenêtre qui demande d'entrer son nom */
const response = prompt('Êtes-vous étudiant: (oui/non) ')

if (response === 'oui' || response === 'non') {
  console.log('Je note cela')
}
```

### Type du résultat

***Attention***

Le résultat obtenu à partir d'une entrée est une chaîne de caractères. Si l'utilisateur entre un nombre, celui-ci sera également récupéré comme une chaîne de caractères.

#### À retenir

Les fonctions de lecture sont utiles pour demander des informations à l'utilisateur et les utiliser dans le programme en les stockant dans une variable.

[cf. PVcPLLOC]

# Exercice : Appliquer la notion

## Question 1

On souhaite demander le mot de passe d'un utilisateur lorsqu'il crée son compte.

Récupérer la réponse de l'utilisateur dans une variable `password`, en lui ayant demandé *« Veuillez entrer votre mot de passe : »*.

Afficher enfin *« Le mot de passe "xxx" est-il correct ? »* en remplaçant par le mot de passe donné.

## Question 2

Ajouter un test, juste après avoir récupéré le mot de passe, permettant de vérifier que le résultat entré par l'utilisateur ne soit pas vide, et n'afficher la deuxième instruction que si le test est vérifié.

### Indice :

Une chaîne vide vaut `''`. Avec un `if`, on peut comparer la valeur de `password` avec la chaîne vide `''`.

# Entrée non textuelle


[cf. oq9GSvoT]

**Objectifs**

- Savoir récupérer des informations non textuelles ;
- Savoir convertir du texte.

**Mise en situation**

Les données attendues lors d'une lecture de l'entrée standard ne sont pas toujours des chaînes de caractères. Il faut alors convertir l'information en nombre, booléen, date, etc.

L'entrée standard permet de récupérer les phrases écrites par l'utilisateur, c'est-à-dire des chaînes de caractères. Pourtant, si l'on veut obtenir son âge et que celui-ci est récupéré dans une chaîne de caractères, les opérations sur celui-ci, comme des additions, des comparaisons, etc., ne donneront pas le résultat escompté.

Il est donc nécessaire de **convertir** le résultat d'une lecture dans ce cas.

**Conversion** *Définition*

Convertir un élément permet de changer son **type**, comme transformer un chiffre en chaîne de caractères, ou inversement une chaîne en chiffre.

**Types** *Rappel*

Le type d'un élément est en quelque sorte sa nature :

- un nombre (entier, décimal, etc.),
- une string (chaîne de caractères),
- un booléen (vrai ou faux),
- une date,
- etc.

**Convertir une valeur** *Syntaxe*

- En JavaScript, on peut convertir un élément grâce aux fonctions `String()`, `Boolean()`, `Number()`, `Date()`, etc.
- En Python, on dispose des fonctions `int()`, `float()`, `str()`, etc.

*Exemple*

```python
"""Python."""
number = '42'
print(type(number)) # affiche le type 'str'

number = int(number) # transforme la chaîne en entier
print(type(number)) # affiche le type 'int'
```

```javascript
/** JavaScript */
let number = '42'
console.log(typeof number) // affiche 'string'

number = Number(number) // transforme la chaîne en nombre
console.log(typeof number) // affiche 'number'
```

**Not a Number**

*Attention*

Il faut toujours partir du principe qu'une entrée d'un utilisateur peut être erronée et ne pas correspondre à ce qui est attendu, comme lire une lettre alors que l'on demande un chiffre.

Il faut donc vérifier les entrées lorsqu'une conversion en nombre suit.

- 1ère approche : on convertit puis on vérifie le résultat.
- 2ème approche : on teste si la variable est convertible, et on convertit seulement si c'est le cas.

**Demande d'un entier**

*Exemple*

En Python, voici comment utiliser l'approche n°2 : on teste d'abord la chaîne avec `isnumeric()` qui retourne `True` ou `False`. Si c'est `True` , on peut convertir.

```python
"""Python."""
age = input('Votre âge : ')

if not age.isnumeric():
  print('Réponse erronée')
else:
  print(age)
```

En JavaScript, voici comment utiliser l'approche n°2 : on teste si la chaîne représente un nombre avec la fonction `isNaN()`. Cette fonction renvoie `true` si la valeur testée n'est pas un nombre (*Not a Number*).

On la convertit ensuite avec la fonction `Number()`.

```javascript
/** JavaScript */
let age = prompt('Votre âge :')

if (!isNaN(age)) {
  age = Number(age)
  console.log(age)
} else {
  console.log('Réponse erronée !')
}

```

**Demande d'une date**

*Exemple*

En Python, la fonction `strptime` (*string parse time*) prend deux arguments : la chaîne à convertir et le format dans lequel elle est écrite.

En JavaScript, la classe `Date` peut transformer une chaîne **AAAA-MM-JJ**, par défaut, en date.

Les manipulations de date sont très sensibles aux formats.

```python
"""Python: convertit la réponse en date."""
import time

birthday = input('Quelle est votre date de naissance? (JJ MM AAAA)\n')

print(time.strptime(birthday, '%d %m %Y')) # format %d %m %Y = JJ MM AAAA
```

```javascript
/** JavaScript: convertit la réponse en date */
let birthday = prompt('Quelle est votre date de naissance? (AAAA-MM-JJ)')

console.log(Date(birthday))
```

**À retenir**

La conversion permet, lors de la lecture d'une entrée, d'adapter la valeur obtenue à l'usage que l'on veut en faire.

[cf. VozusaC8]

# Exercice : Appliquer la notion


## Question 1

On veut vérifier qu'un client a le droit d'accéder à une attraction à sensations fortes.

Réaliser un programme JavaScript qui demande à l'utilisateur de renseigner sa taille et la récupère dans une variable `height`.

Convertir cette taille en nombre ; si la taille est mal donnée, `height` sera mis à zéro.

### Indice :

Ne pas oublier de vérifier que la conversion de la taille donne bien un nombre et ne vaut pas `NaN`.

## Question 2

À partir du code précédent, afficher un message *« C'est bon, vous pouvez passer »* si la taille atteint au moins 1m52.

Si ce n'est pas le cas, afficher *« Désolé, il vous manque x cm pour pouvoir passer »* en indiquant la taille manquante pour atteindre la taille requise.

# Boucles d'entrées/sorties


[cf. 5COjoRnc]

**Objectifs**

- Savoir utiliser les entrées/sorties dans une boucle ;
- Savoir demander des informations jusqu'à obtenir une réponse valide.

**Mise en situation**

En programmation il y a un adage célèbre qui dit : *« Never trust user input »*, ce qui veut dire *« ne jamais faire confiance à ce que l'utilisateur saisit »*. En effet lorsque l'on attend une information de la part d'un utilisateur, il faut toujours s'assurer que l'information entrée est bien celle que l'on souhaite, par exemple qu'elle a le bon format. Si l'on ne fait pas cette vérification, la saisie d'une mauvaise information, comme par exemple une date au lieu d'un prénom, risque de faire planter notre programme. Il est donc nécessaire d'implémenter un mécanisme de vérification, qui se répète tant que la saisie n'est pas correcte.

**Itérations** ***Méthode***

Les structures itératives permettent d'effectuer des boucles pour répéter des instructions, soit jusqu'à un certain nombre de fois, soit jusqu'à ce qu'une condition soit remplie. Les mots clés `for`, `while` et `do while` permettent ces types de boucles.

**Entrées et itérations conditionnelles**

Les entrées sont particulièrement sujettes à des erreurs de la part de l'utilisateur. On ne peut pas savoir avec certitude si sa réponse va correspondre à ce qui est attendu. Dans le cas où la réponse est trop courte, trop longue, n'est pas au bon format (pour un e-mail ou un téléphone par exemple), etc. il faut souvent redemander une information, et ce potentiellement plusieurs fois.

Les **structures itératives** utilisant une condition sont alors particulièrement adaptées.

**Prompt en boucle**

Pour demander une information jusqu'à ce que celle-ci soit correcte, on peut tester la ou les conditions après le *prompt* ou l'*input*, et répéter la demande tant que les critères ne sont pas remplis. Pour cette usage, la boucle `while` est indiquée.

**Vérifier la taille de la réponse** ***Exemple***

```
/** JavaScript: boucle tant que la réponse fait moins de 12 caractères */
let password = prompt('Entrez un mot de passe valide (12 caractères minimum)')

while (password.length < 12) {
  password = prompt('Mot de passe trop court. Réessayez')
}

console.log('Mot de passe accepté')
```

```python
"""Python: boucle tant que la réponse fait moins de 12 caractères."""
password = input('Entrez un mot de passe valide (12 caractères minimum)\n')

while len(password) < 12:
  password = input('Mot de passe trop court. Réessayez\n')

print('Mot de passe accepté')
```

**À retenir**

Combiner les boucles avec la lectures d'entrées est un bon moyen de réitérer une demande en vérifiant que la réponse est correcte.

[cf. w2Y5u9TW]

# Exercice : Appliquer la notion


## Question 1

On veut réaliser un quiz qui pose une question et la répète tant que la réponse n'est pas trouvée.

La première question est : *« En quelle année fut signé le décret d'abolition de l'esclavage en France ? »*

La bonne réponse, *« 1848 »*, est stockée dans une constante .

Écrire le programme qui pose la question une fois, récupère la réponse dans une variable et la compare à la bonne réponse, puis affiche *« Bonne réponse »* si le joueur trouve. Tant que la réponse n'est pas la bonne, afficher *« Réessayez »*.

## Question 2

Ajouter une limite de 4 tentatives pour trouver la réponse. *« Bonne réponse »* ne doit être affiché que si le joueur a trouvé la réponse. Sinon, afficher *« Dommage »*.

### Indice :

On utilise un compteur incrémenté de 1 à chaque boucle. La condition du compteur inférieur à 4 doit être prise en compte dans la condition du `while`.

# Essentiel


[cf. lLwANhaO]

Il existe un grand nombre de périphériques différents : clavier, souris, écran, microphone, etc. Chacun de ces périphériques permet à un programme de lire ou écrire des informations pour interagir avec le monde extérieur. Dans ce module nous avons surtout abordé les échanges textuels, avec le clavier et l'écran, qui s'avèrent être les plus utilisés. Nous avons aussi vu qu'il était important de bien maîtriser le format des données reçues en entrée, et de systématiquement vérifier que l'utilisateur saisit ce que l'on souhaite réellement.

# Quiz


## Exercice 1 : Quiz - Culture

**Exercice**

Quelles affirmations sont vraies ?

- ☐ Afficher un message à destination de l'utilisateur sur un écran tactile est une sortie.
- ☐ Un flux d'informations du processeur vers l'écran est une écriture.
- ☐ Lire les informations en entrée, c'est lire l'affichage du programme à l'écran.
- ☐ Le clavier et l'écran sont deux périphériques de sortie.

**Exercice**

Quelles affirmations sont correctes ?

- ☐ Les informations données par l'utilisateur ne doivent pas être considérées comme fiables.
- ☐ Un `output` est une entrée depuis un périphérique.
- ☐ Un flux d'un ordinateur vers une imprimante est un flux de sortie.
- ☐ En lisant une entrée de l'utilisateur, on récupère forcément au moins un caractère.

## Exercice 4 : Quiz - Méthode

**Exercice**

Comment peut-on afficher le chiffre 4 en sortie standard ?

- ☐ `console.log(String(4))`
- ☐ `console.log(4)`
- ☐
```
const NB = 4
console.log(NB)
```

**Exercice**

Récupérer un entier en entrée en JavaScript suppose que :

- ☐ L'utilisateur n'a utilisé que des chiffres.
- ☐ Une conversion en nombre est nécessaire.
- ☐ La conversion produit une erreur si ce n'est pas un nombre valable.

**Exercice**

Que vaut la constante `languages` quand l'utilisateur répond « deux » ?

Notez que la conversion s'effectue **avant** la vérification.

```
/** JavaScript */
const languages = Number(prompt('Combien de langues parles-tu ?'))

if(isNaN(languages)) {
  console.log(languages, 'n\'est pas un nombre')
} else {
  console.log('C\'est bien noté')
}
```

# Exercice 8 : Quiz - Code

**Exercice**

Quel code récupère un nombre au lieu d'une chaîne de caractères ?

- `Number(money) = prompt('Combien voulez-vous retirer ?')`
- `money = prompt('Combien voulez-vous retirer ?').toInt()`
- `money = Number(prompt('Combien voulez-vous retirer ?'))`
- `money = prompt('Combien voulez-vous retirer ?') + 0`

**Exercice**

Qu'affiche le programme ?

```
/** JavaScript */
console.log(typeof NaN)
```

- string
- null
- number

**Exercice**

Qu'affichera le programme suivant ?

```
/** JavaScript */
const solution = 0
const answer = prompt('Quel est le premier chiffre pair?')
console.log(solution === answer)
```

- error
- false
- true
- Tout dépend de la réponse