Les enregistrements

Table des matières

Introduction

Le développeur doit souvent représenter des concepts ayant plusieurs caractéristiques et qu'il n'est pas simple de représenter via une simple variable. Par exemple, un site de e-commerce possède plusieurs informations sur un client, comme son nom, son adresse, etc. Si le site souhaite regrouper toutes ces informations au sein d'une seule variable, il est nécessaire d'utiliser une nouvelle structure de données que l'on appelle des enregistrements.

Dans ce module nous allons découvrir ce qu'est un enregistrement, comment l'utiliser, et comment il est implémenté dans les langages de programmation.

I Notion d'enregistrement

Objectifs

- Comprendre l'utilité des enregistrements ;
- Savoir déclarer un enregistrement en JavaScript.

Mise en situation

Pour un développeur, il peut être pratique de stocker plusieurs valeurs ayant un lien au sein d'une même variable. Pour réussir cela, il est nécessaire d'utiliser des enregistrements.

Enregistrement Az Définition

Un enregistrement est un type correspondant à un agrégat de variables (appelées **champs**) de types déjà définis. Ainsi, une seule variable aura plusieurs **composantes** ayant chacune son propre type. On peut donc voir un enregistrement comme une collection de variables.

Enregistrement d'une voiture Exemple

Dans un programme, on peut définir le type structuré `voiture` qui aura les composantes suivantes : *« marque »*, *« couleur »* et *« année de production »*. Les deux premiers sont des chaînes de caractères et le dernier un entier. Toutes les variables de type `voiture` dans le programme posséderont ces composantes.

```
Type voiture:
    - marque: chaine de caractères
    - couleur: chaine de caractères
    - année: entier
```

Simplification de la manipulation

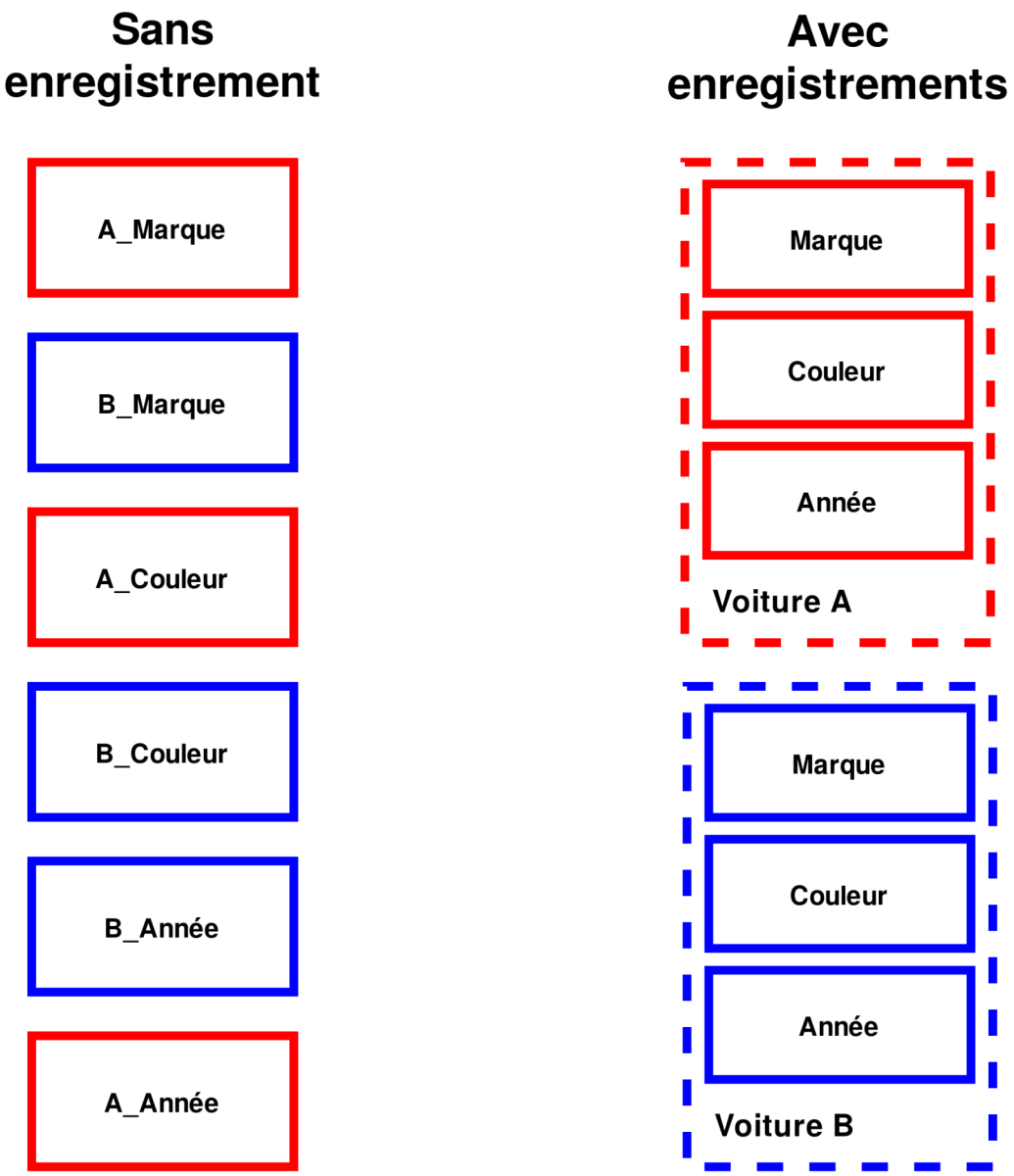
Utiliser des enregistrements peut simplifier les programmes et leur lecture. Voici un exemple où l'on souhaite comparer deux voitures A et B. Dans le premier cas, l'opération est effectuée sans enregistrement contrairement au second qui en utilise. Cela apporte une meilleure lisibilité et cela réduit le nombre de variables à gérer pour le programmeur.

```
Si A_marque == B_marque:
    Afficher "Même marque"
FinSi
Si A_couleur == B_couleur:
    Afficher "Même couleur"
FinSi
Si A_année == B_année:
    Afficher "Même année"
FinSi
```

```
Si voiture_A.marque == voiture_B.marque:
    Afficher "Même marque"
FinSi
Si voiture_A.couleur == voiture_B.couleur:
    Afficher "Même couleur"
FinSi
Si voiture_A.année == voiture_B.année:
    Afficher "Même année"
FinSi
```

Gestion de la mémoire

Bien qu'il y ait moins de variables, le programme utilisera toujours autant d'espace mémoire. L'espace mémoire sera simplement *« mieux »* ordonné car les composantes seront stockées ensemble en mémoire. Les variables dont le type est voiture seront plus volumineuses que des variables de type simple (entier, etc.).



Implémentation en JavaScript

Méthode

En JavaScript, il est possible de créer des enregistrements grâce au type `object`. Voici un exemple :

```
/** JavaScript : modifie l'enregistrement de la voiture de Bob. */
const carBob = { // On initialise toutes les composantes
  brand: 'Renault',
  color: 'bleu',
  year: 2012
}

carBob.year = 2020 // On met à jour une composante

console.log(carBob.color) // On affiche une des composantes

```

Accéder à la valeur d'une composante

Syntaxe

Il existe deux syntaxes pour accéder à la valeur d'une composante :

```
/** JavaScript : accède aux informations de l'enregistrement de la voiture de
Bob avec deux manières. */
const carBob = { // On initialise toutes les composantes
  brand: 'Renault',
  color: 'bleu',
  year: 2012
}

// Les deux syntaxes suivantes retournent la même valeur
console.log(carBob.color, carBob['color'])

```

À retenir

- Les enregistrements permettent d'avoir des variables représentant des objets plus complexes.
- En JavaScript, le type `object` peut être utilisé pour créer des enregistrements.

II Exercice : Appliquer la notion

Question

[solution n°1 p. 26]

Voici un extrait de code. Le compléter pour qu'il affiche *« Bonjour Jean Dupont »*.

```
1 const user = {?}
2
3 console.log("Bonjour", user.firstName, user.lastName)
```

Indice :

`user` est un enregistrement qui doit contenir deux variables : une représentant le prénom, et une autre représentant le nom.

III Boucles et enregistrements

Objectif

- Savoir itérer sur les composantes des enregistrements.

Mise en situation

De la même manière que les tableaux, il est parfois utile de pouvoir parcourir un enregistrement, par exemple pour afficher l'ensemble de ses composantes. Dans le cas des tableaux, on itère simplement sur les différents éléments. Pour les enregistrements, on itère sur les composantes, ce qui permet de récupérer leur nom et leur valeur.

Itérer sur les composantes d'un enregistrement

Lorsque les enregistrements possèdent de nombreuses composantes ou que leur nombre est variable, il est pratique de pouvoir effectuer une opération sur chacune des composantes sans avoir à énumérer toutes les possibilités dans le code. Cela permet de raccourcir le code et d'améliorer un peu sa lisibilité.

Raccourcir le code

Exemple

Voici deux exemple en JavaScript qui montre bien le gain en lignes de code. Le second utilise une boucle qui facilite la compréhension du code. En effet, sans cette boucle, un lecteur devrait lire attentivement pour savoir si certaines composantes ont été ignorées. La boucle convoie l'idée que l'opération est faite sur **toutes** les composantes sans exception. De plus, la boucle permet d'économiser des lignes et du temps pour le développeur.

```javascript
/** JavaScript : Affiche les informations de la voiture de Bob. */
const carBob = {
  brand: 'Renault',
  color: 'bleu',
  year: 2012,
  engine: 'V1337',
  country: 'France',
  fuel: 'diesel'
}

console.log('brand', carBob.brand)
console.log('color', carBob.color)
console.log('year', carBob.year)
console.log('engine', carBob.engine)
console.log('country', carBob.country)
console.log('fuel', carBob.fuel)

```

```javascript
/** JavaScript : Affiche les informations de la voiture de Bob en utilisant une boucle for. */
const carBob = {
  brand: 'Renault',
  color: 'bleu',
  year: 2012,
  engine: 'V1337',
  country: 'France',
  fuel: 'diesel'
}
```

```
10
11 for (const composante in carBob) {
12   console.log(composante, carBob[composante])
13 }
14
```

for..of : Parcourir les attributs d'un enregistrement

Syntaxe

```
1 for (let property in objetVar) {
2   // property est le nom de la composante
3 }
```

Cette syntaxe pourrait se traduire par *« Pour chaque `composante` dans l'enregistrement `objetVar` faire... »*.

for..in : parcourir les valeurs d'un enregistrement

Syntaxe

```
1 for (let value of Object.values(objetVar)){
2   // value est la valeur de la composante
3 }
```

Cette syntaxe pourrait se traduire par *« Pour chaque `valeur` dans l'enregistrement `objetVar` faire... »*.

À retenir

- Il est possible d'itérer sur les composantes d'un enregistrement.
- L'itération peut s'effectuer sur le nom des composantes ou sur leur valeur.
- Itérer sur les composantes d'un enregistrement permet de gagner en lisibilité et est indispensable pour des très grands enregistrements.

IV Exercice : Appliquer la notion

Question

[solution n°2 p. 26]

Compléter le code suivant pour qu'il affiche les valeurs de toutes les composantes de cet enregistrement, sauf `pseudo`.

Utiliser une boucle `for`.

```
const user = {
  pseudo: 'Dupont',
  birthYear: 1990,
  birthMonth: 12,
  birthDar: 12
}
```

Indice :

La syntaxe `for...in` est à privilégier, car une condition est posée sur le **nom** d'une composante.

V Tableaux d'enregistrements

Objectif

- Savoir gérer des tableaux d'enregistrements.

Mise en situation

Nous avons vu que les enregistrements sont très utiles pour décrire des concepts complexes, par exemple une voiture. Si l'on souhaite ensuite manipuler un grand nombre d'enregistrements similaires, on va naturellement se tourner vers les tableaux. En effet il est possible de simplement créer un tableau d'enregistrements, et de l'utiliser comme tout les tableaux classiques. Encore plus loin, il est possible de créer un enregistrement qui soit imbriqué dans la composante d'un autre enregistrement, lui-même dans un tableau d'enregistrement. Nous allons voir que pour gérer ces structures de plus en plus complexes, un format standard de représentation a été mis en place.

Imbriquer les enregistrements

Il est possible de créer des tableaux ou des listes d'enregistrements. Ces variables sont alors vues comme des imbrications d'enregistrements.

Calculer le montant d'un commande

Exemple

Cela peut être utile pour calculer le montant d'un panier sur un site de e-commerce. Chaque produit est représenté par un enregistrement contenant son nom et son prix. Voici un exemple :

```
/** JavaScript : calcul du prix d'un panier. */
const basket = [
  { name: 'souris', price: 10.5 },
  { name: 'clavier', price: 25 },
  { name: 'écran', price: 50 }
]

let sum = 0

for (const item of basket) {
  sum = sum + item.price
}

console.log(sum)

```

Le format JSON

Az Définition

JSON, pour JavaScript Object Notation, est un format de données textuel dérivé de la notation des enregistrements en JavaScript. Ce format est très pratique pour le transfert de données structurées. Le format de ces données est très proche de celui d'un enregistrement ou d'un tableau d'enregistrements. Voici un extrait de JSON représentant des employés :

```
{
  "firstName": "Ada",
  "lastName": "Lovelace",
  "birth": 1815,
  "death": 1852,
  "jobs": ["programmeuse", "mathématicienne", "poétesse", "traductrice", "ingénieure"],
  "institution": "Université de Cambridge"
}
```

Utilisation du format JSON

⊕ Complément

Le format JSON est utilisé essentiellement pour le transfert de petites quantités de données. Il s'est aujourd'hui distingué sur le Web pour la communication entre le client et le serveur. Dans l'exemple ci-dessus, le client aurait demandé la liste des employés au serveur. Le serveur envoie le document JSON et le client se charge d'intégrer les informations contenues dans le JSON à la page Web.

Imbrication d'enregistrements

⊕ Complément

Il est possible d'imbriquer les enregistrements. Cela signifie que la valeur d'une composante d'un enregistrement sera elle-même un enregistrement comme dans l'exemple ci-dessous avec la date de naissance qui est un enregistrement.

```
/** JavaScript : imbrication d'enregistrements. */
const user = {
  lastName: 'Tesla',
  firstName: 'Nikola',
  birth: { day: 10, month: 7, year: 1856 },
  job: 'ingénieur'
}

```

À retenir

- En JavaScript, il est possible de gérer des collections d'enregistrements grâce à des tableaux.
- Le format JSON est dérivé des enregistrements et est très utilisé pour le transfert de données structurées.

VI Exercice : Appliquer la notion

Question 1

[solution n°3 p. 26]

Ce tableau d'enregistrements décrit une liste d'employés. Compléter ce code pour calculer l'âge moyen des ingénieurs.

```
const employees = [
  { firstName: 'John', lastName: 'Doe', age: 25, job: 'ingénieur' },
  { firstName: 'Bob', lastName: 'Smith', age: 38, job: 'chercheur' },
  { firstName: 'Jeanne', lastName: 'Smith', age: 40, job: 'ingénieur' },
  { firstName: 'Mathieu', lastName: 'Simpson', age: 59, job: 'secrétaire' },
  { firstName: 'Constance', lastName: 'Martin', age: 40, job: 'directeur' },
  { firstName: 'Robert', lastName: 'Peter', age: 30, job: 'ingénieur' },
  { firstName: 'Richard', lastName: 'Stallman', age: 67, job: 'chercheur' }
]

```

Indice :

On pourra utiliser la syntaxe `for...of` pour parcourir le tableau `employees` et accéder aux enregistrements individuels.

Question 2

[solution n°4 p. 26]

Quel est le résultat pour l'enregistrement donné comme exemple ?

VII Objets et autres structures complexes

Objectifs

- Comprendre le lien entre tableau associatif et enregistrement ;
- Découvrir le concept d'objet en programmation.

Mise en situation

Le concept d'enregistrements, bien que très pratique, n'est pas implémenté dans tous les langages car il existe d'autres structures complexes très proches qui pourront répondre à des besoins similaires.

Lien entre tableau associatif et enregistrement

Un **tableau associatif** est une type complexe proche des enregistrements. On parle également de **dictionnaires**.

Dans un tableau associatif, on lie une **clé** (chaîne de caractères ou nombre) à une **valeur**. Un tableau associatif est donc une collection de clés associées à une valeur.

La différence entre enregistrement et dictionnaire réside dans la nuance entre **nom de la composante** et **clé**.

Dans un enregistrement, on accède à une valeur grâce au nom de la composante. Dans un tableau associatif, on y accède grâce à une clé.

D'un point de vue théorique, il existe une différence entre enregistrement et tableau associatif mais celle-ci est souvent perdue lors de l'implémentation dans les différents langages de programmation.

Enregistrements et tableaux associatifs en JavaScript

Syntaxe

En JavaScript, le type `object` est utilisé pour simuler des enregistrements, tandis que le type `Map` est utilisé pour les tableaux associatifs.

Si les deux entités semblent similaires, la méthode d'accès et de création est assez différente.

```
1  // Enregistrement utilisant le type object
2  const userEnregistrement = {
3    firstName: 'Jean',
4    lastName: 'Dupont'
5  }
6
7  // Accès aux composantes avec la syntaxe var.nom
8  console.log(userEnregistrement.lastName)
9
10 // Tableau associatif utilisant le type Map
11 const userDico = new Map([
12   ['firstName', 'Jean'],
13   ['lastName', 'Dupont']
14 ])
15
16 // Accès aux valeurs avec la syntaxe var.get(clé)
```

```
console.log(userDico.get('lastName'))
```

Les dictionnaires en Python

Syntaxe

En Python, il n'existe pas d'implémentation native semblable à un enregistrement, il n'est possible que de faire des dictionnaires grâce au type `dict`. Sa syntaxe est proche de la déclaration d'un enregistrement JavaScript :

```
utilisateurDico = {  # type dict
  "prenom": "Jean",
  "nom": "Dupont"
}

print(utilisateurDico["nom"])
```

Quand utiliser l'un ou l'autre ?

Conseil

Les frontières entre les deux concepts sont floues. Leur utilisation dépend essentiellement de la **sémantique**. Voici une *règle générale* pour choisir :

- Les enregistrements sont privilégiés pour représenter la structure, c'est-à-dire les éléments, d'un concept : une voiture, une personne, un livre...
- Les tableaux associatifs sont privilégiés pour stocker des **paires** indépendantes les unes des autres : associer un jour à une température, un élève à sa moyenne... Aucune notion de squelette ou de structure globale n'entre en jeu.

Vers la notion d'objet

La grande limite des enregistrements est le dynamisme de ses composantes. En effet, lorsqu'on possède deux enregistrements décrivant des voitures, il n'est pas possible de fixer des composantes obligatoires. Avec ce type de variables, il est alors possible d'avoir des déconvenues comme dans l'exemple ci-dessous. Il y a deux enregistrements de voitures mais il sera très compliqué de les utiliser ensemble car d'une part, ils ne possèdent pas les mêmes composantes. D'autre part, une des composantes possède un nom légèrement différent entre les deux enregistrements.

```
const carA = {
  brand: 'Renault',
  year: 1998
}

const carB = {
  brand: 'Peugeot',
  color: 'Bleue',
  yearManufacture: 2005
}

```

Les objets

Fondamental

Pour éviter ce problème, on utilise des **objets**, dont les attributs (équivalent des composantes) sont fixés à l'avance. Ainsi, il faut au préalable spécifier la structure des objets, puis on pourra créer plusieurs variables de ce type qui auront exactement les mêmes attributs.

Créer un type objet qui sera utilisé par plusieurs variables permet de standardiser les attributs/composantes ainsi que leur nom.

```
Type Voiture:  // On appelle ce type une "classe"
  - marque: chaine de caractère
  - couleur: chaine de caractère
  - annee: nombre


voitureA = nouveau Voiture() // On crée un nouvel objet de la classe Voiture
voitureA.marque = "Renault"
voitureA.couleur = "Rouge"
voitureA.annee = 1998

voitureB = nouveau Voiture()
voitureB.marque = "Peugeot"
voitureB.couleur = "Bleue"
voitureB.annee = 2005
```

Confusion en JavaScript

Attention

Le type `object` en JavaScript sert à la fois à représenter des enregistrements et des objets. La syntaxe vue jusque ici sert à créer des **enregistrements** via le type `object` de JavaScript.

Implémenter un objet

Syntaxe

Un objet se base sur une **classe**, qui est le squelette que devra respecter l'ensemble des objets. Voici comment créer une classe `Voiture` puis instancier des objets de cette classe en JavaScript, puis en Python.

```
/** JavaScript : définition de la classe Voiture et instanciation d'objets
Voiture. */
class Car {
  constructor () {
    this.brand = '' // Valeur par défaut
    this.color = ''
    this.year = 0
  }
}

const carA = new Car() // On crée un nouvel objet de la classe Voiture
carA.brand = 'Renault'
carA.color = 'Rouge'
carA.year = 1998

const carB = new Car()
carB.brand = 'Peugeot'
carB.color = 'Bleue'
carB.year = 2005

```

```python
"""Python: définition de la classe Voiture et instanciation d'objets Voiture."""
class Car():
  brand = ""  # Valeur par défaut
  color = ""
  year = 0

car_a = Car() # On crée un nouvel objet de la classe Voiture
car_a.brand = "Renault"
car_a.color = "Rouge"
car_a.year = 1998

car_b = Car()
car_b.brand = "Peugeot"
car_b.color = "Bleue"
car_b.year = 2005
```

Cette fois-ci, les noms des variables composant les objets sont fixés à l'avance par la classe, et tous les objets `Car` pourront être utilisés de la même façon, sans risque.

À retenir

- Certains langages, comme le Python, n'ont pas d'implémentation des enregistrements et utilisent uniquement des dictionnaires qui sont un type structuré très proche.
- Pour avoir des variables dont les composantes sont standardisées, il est intéressant d'utiliser des objets.

VIII Exercice : Appliquer la notion

Question

[solution n°5 p. 26]

Écrire la définition de la classe `Utilisateur` afin que le code suivant soit fonctionnel :

```
class User {
  constructor () {
    ?
  }
}

const jean = new User()
jean.lastName = 'Dupont'
jean.firstName = 'Jean'
jean.age = 28

console.log('Bonjour', jean.firstName, jean.lastName)
```

Indice :

L'objet `jean` fait référence à deux attributs : `lastName` et `firstName`. Ces attributs doivent être déclarés dans la classe, c'est-à-dire dans le squelette que respectent les objets User.

IX Exercice final

Exercice 1 : Quiz - Culture

[solution n°6 p. 27]

Exercice

Un enregistrement permet d'économiser de l'espace mémoire.

- A Vrai
- B Faux

Exercice

Voici la définition d'une variable en JavaScript, quel est son type, retourné par *typeof* ?

```
1 const utilisateur = {
2   prenom: 'Tyler',
3   nom: 'Durden'
4 }
5 console.log(typeof (utilisateur))
6
```

- A Map
- B Array
- C dict
- D object

Exercice

Voici la définition d'une variable en JavaScript, quel est son type, retourné par *typeof* ?

```
1  const utilisateurs = [
2    {
3      prenom: 'Tyler',
4      nom: 'Durden'
5    },
6    {
7      prenom: 'Marla',
8      nom: 'Singer'
9    }
10 ]
11 console.log(typeof (utilisateurs))
12
```

- A Map
- B Array

C dict

D objet

Exercice

Quand est-il préférable d'utiliser un tableau associatif plutôt qu'un enregistrement ?

A Quand il y a peu de données à stocker.

B Quand on veut représenter la structure d'un objet tel qu'une personne.

C Quand la performance d'accès aux données est un besoin critique.

Exercice 6 : Quiz - Méthode

[solution n°7 p. 28]

Exercice

Quel type JavaScript permet d'implémenter des enregistrements ?

A array

B record

C object

D string

E map

Exercice

Laquelle de ces structures de données n'est pas directement implémentée en Python ?

A Liste

B Chaîne de caractères

C Enregistrement

D Dictionnaire

Exercice

En JavaScript, nous avons déclaré une classe `Product`. La déclaration suivante est-elle correcte pour créer un objet de la classe `Product` ?

```
computer = Product()
```

A Oui

B Non

Exercice

En JavaScript, nous avons déclaré une classe `Product`. Nous souhaitons itérer sur le nom des **attributs** d'un produit dont la variable se nomme `computeur`. Par quel terme doit-on remplacer le ? pour réussir à itérer ?

```
for (let champ ? computeur) {}
```

Exercice 11 : Quiz - Code

[solution n°8 p. 30]

Exercice

Déterminer ce qui est affiché par le programme suivant.

```
const framabook = {
  title: 'Vieux flic et vieux voyou ',
  author: 'Frédéric Urbain',
  year: 2015
}

console.log(framabook.year)

```

Exercice

Déterminer **le nombre de lignes** affiché par le programme suivant.

```
const framabook = {
  title: 'Working Class Heroic Fantasy',
  author: 'Simon Giraudot',
  year: 2018
}

for (const field in framabook) {
  console.log(framabook[field])
}

```

Exercice

Sans l'exécuter, déterminer le nombres de lignes affichés par le programme suivant.

```
const framabook = {
  title: 'Traces',
  author: 'Stéphane Crozat',
  year: 2018
}

for (const field in framabook) {
  if (field === 'year') {
    console.log(framabook[field])
  }
  if (framabook[field] === 'Traces') {
    console.log(field)
  }
}

```

X Exercice : Défi

On souhaite développer un programme qui demande à l'utilisateur d'entrer une liste de produits à mettre dans un panier et qui les enregistre grâce à une imbrication d'enregistrements. Ensuite, le programme itère sur les enregistrements et retourne le prix du panier. Voici le programme à compléter :

```
let otherItem = true
const listItem = []
while (/* A COMPLETER */) {
  const nameItem = prompt('Donner le nom du produit')
  const priceItem = Number(prompt('Donner le prix du produit'))
  const item = /* A COMPLETER */
  listItem.push(item)

  otherItem = (prompt('Autre produit? O ou N?') === 'O')
}

let sum = 0
/* A COMPLETER */

console.log(sum)

```

Question 1

[solution n°9 p. 31]

Quelle est la condition à compléter dans la boucle `while` ?

Indice :

On dispose d'une variable booléenne nommée `otherItem`, qui indique si l'utilisateur veut insérer un autre produit.

Question 2

[solution n°10 p. 31]

Compléter la ligne contenue dans la boucle `while` :

```
const item = /* A COMPLETER */
```

Indice :

`item` est un enregistrement composé du nom du produit et de son prix.

Question 3

[solution n°11 p. 31]

Ajouter une instruction d'affichage de la liste des produits et exécuter le code avec les valeurs suivantes :

- Pomme à 0.1€
- Poire à 0.2€

Quel affichage obtenez-vous ?

Question 4

[solution n°12 p. 31]

Donner le code qui permet de calculer la somme du prix des produits dans la variable `sum`.

Indice :

Une boucle `for` serait utile pour itérer sur le tableau des enregistrements.

Question 5

[solution n°13 p. 31]

Donner le programme JavaScript complet.

Pour améliorer notre code, nous souhaitons utiliser un objet plutôt qu'un enregistrement pour définir un produit.

Question 6

[solution n°14 p. 32]

Donner la définition de la classe `Item.`

Indice :

Comme les enregistrements définis jusqu'à maintenant, la classe `Item` déclarera deux attributs : un pour le nom, et un pour le prix.

Il est utile d'initialiser ces attributs à des valeurs "nulles" (0, chaîne vide...).

Question 7

[solution n°15 p. 32]

Donner le programme modifié en remplaçant les enregistrements par des objets.

Indice :

Un nouvel objet Item peut être déclaré avec la syntaxe `const item = new Item()`.

Il faut ensuite mettre à jour les valeurs de ses attributs.

Conclusion

La notion d'enregistrement est un premier pas vers l'utilisation de données complexes. Nous avons vu qu'ils permettent de décrire un concept ne pouvant se résumer à un simple nombre ou texte, en le décomposant en variables ayant un type standard. Chaque variable est une composante de l'enregistrement, avec une valeur. Comme pour les tableaux, il est possible de parcourir l'ensemble des champs d'un enregistrement, par exemple pour l'afficher.

Nous avons aussi évoqué la représentation JSON, qui permet de représenter des tableaux et enregistrements imbriqués les uns dans les autres, formant des ensembles de données encore plus complexes.

Enfin, nous avons vu la différence entre tableaux associatif, qui représentent des paires d'éléments, et enregistrements, qui décrivent la structure d'une entité. Les objets permettent de définir la structure d'une entité une bonne fois pour toutes, à l'avance.

Solutions des exercices

Solution n°1

[exercice p. 7]

```
const user = {
  firstName: 'Jean',
  lastName: 'Dupont'
}

console.log('Bonjour', user.firstName, user.lastName)
```

Solution n°2

[exercice p. 10]

```
const user = {
  pseudo: 'Dupont',
  birthYear: 1990,
  birthMonth: 12,
  birthDar: 12
}

for (const composante in user) {
  if (composante !== 'pseudo') {
    console.log(user[composante])
  }
}

```

Solution n°3

[exercice p. 13]

```
let counter = 0
let sum = 0
for (const employee of employees) {
  if (employee.job === 'ingénieur') {
    sum = sum + employee.age
    counter = counter + 1
  }
}

const average = sum / counter
```

Solution n°4

[exercice p. 13]

31.666666666666668

Solution n°5

[exercice p. 18]

```
class User {
  constructor () {
    this.lastName = ''
    this.fistName = ''
    this.age = 0
  }
```

```
}

const jean = new User()
jean.lastName = 'Dupont'
jean.firstName = 'Jean'
jean.age = 28

console.log('Bonjour', jean.firstName, jean.lastName)

```

Solution n°6

[exercice p. 19]

Exercice

Un enregistrement permet d'économiser de l'espace mémoire.

A Vrai

B Faux

L'espace mémoire n'est pas réduit, il est dans le meilleur des cas mieux ordonné.

Exercice

Voici la définition d'une variable en JavaScript, quel est son type, retourné par *typeof* ?

```
const utilisateur = {
  prenom: 'Tyler',
  nom: 'Durden'
}
console.log(typeof (utilisateur))

```

A Map

B Array

C dict

D object

Les enregistrements, en JavaScript, sont **simulés** avec le type `object`.

Exercice

Voici la définition d'une variable en JavaScript, quel est son type, retourné par *typeof* ?

```
const utilisateurs = [
  {
    prenom: 'Tyler',
    nom: 'Durden'
  },
  {
    prenom: 'Marla',
    nom: 'Singer'
  }
]
console.log(typeof (utilisateurs))

```

A Map

B Array

C dict

D objet

Il s'agit bien ici d'un tableau d'enregistrements, mais en JavaScript le type tableau n'existe pas, il n'y a qu'un seul type non primitif, l'objet : les tableaux sont donc aussi des objets.

developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Types_et_grammaire

Exercice

Quand est-il préférable d'utiliser un tableau associatif plutôt qu'un enregistrement ?

A Quand il y a peu de données à stocker.

B Quand on veut représenter la structure d'un objet tel qu'une personne.

C Quand la performance d'accès aux données est un besoin critique.

Solution n°7

[exercice p. 20]

Exercice

Quel type JavaScript permet d'implémenter des enregistrements ?

A array

B record

C object

D string

E map

Les enregistrements n'existent pas en tant que tel en JavaScript : le type `object` permet de les simuler et peut se comporter comme un enregistrement.

Exercice

Laquelle de ces structures de données n'est pas directement implémentée en Python ?

A Liste

B Chaîne de caractères

C Enregistrement

D Dictionnaire

Python implémente les concepts d'objet et de dictionnaire (tableau associatif), mais pas d'enregistrement.

Exercice

En JavaScript, nous avons déclaré une classe `Product`. La déclaration suivante est-elle correcte pour créer un objet de la classe `Product` ?

```
1 computer = Product()
```

A Oui

B Non

Pour instancier un objet d'une classe en JS, il faut utiliser la syntaxe `new` :

```
1 computer = new Product()
```

Exercice

En JavaScript, nous avons déclaré une classe `Product`. Nous souhaitons itérer sur le nom des **attributs** d'un produit dont la variable se nomme `computeur`. Par quel terme doit-on remplacer le ? pour réussir à itérer ?

```
1 for (let champ ? computeur) {}
```

in

La syntaxe `for...in` permet d'itérer sur les noms des composantes d'un enregistrement ou sur les noms des attributs d'un objet.

Solution n°8

[exercice p. 21]

Exercice

Déterminer ce qui est affiché par le programme suivant.

```
const framabook = {
  title: 'Vieux flic et vieux voyou ',
  author: 'Frédéric Urbain',
  year: 2015
}

console.log(framabook.year)

```

2015

Exercice

Déterminer **le nombre de lignes** affiché par le programme suivant.

```
const framabook = {
  title: 'Working Class Heroic Fantasy',
  author: 'Simon Giraudot',
  year: 2018
}

for (const field in framabook) {
  console.log(framabook[field])
}

```

3

Exercice

Sans l'exécuter, déterminer le nombres de lignes affichés par le programme suivant.

```
const framabook = {
  title: 'Traces',
  author: 'Stéphane Crozat',
  year: 2018
}

for (const field in framabook) {
  if (field === 'year') {
    console.log(framabook[field])
  }
  if (framabook[field] === 'Traces') {
    console.log(field)
  }
}

```

2

Solution n°9

[exercice p. 23]

```
while (otherItem) {
...
}
```

Solution n°10

[exercice p. 23]

```
const item = {
  name: nameItem,
  price: priceItem
}
```

Solution n°11

[exercice p. 23]

```
console.log(listItem)
```

On obtient l'affichage suivant :

```
Donner le nom du produit> Pomme
Donner le prix du produit> 0.1
Autre produit? O ou N?> O
Donner le nom du produit> Poire
Donner le prix du produit> 0.2
Autre produit? O ou N?> N
[ { name: 'Pomme', price: 0.1 }, { name: 'Poire', price: 0.2 } ]
```

Solution n°12

[exercice p. 24]

```
for (const item of listItem) {
  sum = sum + item.price
}
```

Solution n°13

[exercice p. 24]

```
let otherItem = true
const listItem = []
while (otherItem) {
  const nameItem = prompt('Donner le nom du produit')
  const priceItem = Number(prompt('Donner le prix du produit'))
  const item = {
    name: nameItem,
    price: priceItem
  }
  listItem.push(item)

  otherItem = (prompt('Autre produit? O ou N?') === 'O')
}

console.log(listItem)

let sum = 0
for (const item of listItem) {
  sum = sum + item.price
}

console.log(sum)
```

Solution n°14

[exercice p. 24]

```
class Item {
  constructor () {
    this.name = ''
    this.price = 0
  }
}
```

Solution n°15

[exercice p. 24]

```
class Item {
  constructor () {
    this.name = ''
    this.price = 0
  }
}

let otherItem = true
const listItem = []
while (otherItem) {
  const item = new Item()
  item.name = prompt('Donner le nom du produit')
  item.price = Number(prompt('Donner le prix du produit'))
  listItem.push(item)

  otherItem = (prompt('Autre produit? O ou N?') === 'O')
}

let sum = 0
for (const item of listItem) {
  sum = sum + item.price
}

console.log(sum)

```