Chapitre 1-4
Installer son environnement de travail

1. Introduction

Il ne s'agit ici que de CPython, l'implémentation de référence de Python, et non de PyPy ou Jython.
Quel que soit votre système d'exploitation, vous pouvez installer Python en lisant ce chapitre puis, dans un second temps, installer des bibliothèques tierces au gré de vos besoins (cf. section Installer une bibliothèque tierce) et vous pourrez créer des environnements virtuels (cf. section Créer un environnement virtuel).
Si vous souhaitez installer d'un seul coup Python ainsi que Jupyter (anciennement IPython) et la plupart des bibliothèques scientifiques ou d'analyse de données, vous pouvez aller directement à la section Installer Anaconda, pour installer celui-ci en lieu et place de Python. Vous disposerez alors d'autres méthodes pour gérer les environnements virtuels et pour installer des bibliothèques tierces.

2. Installer Python

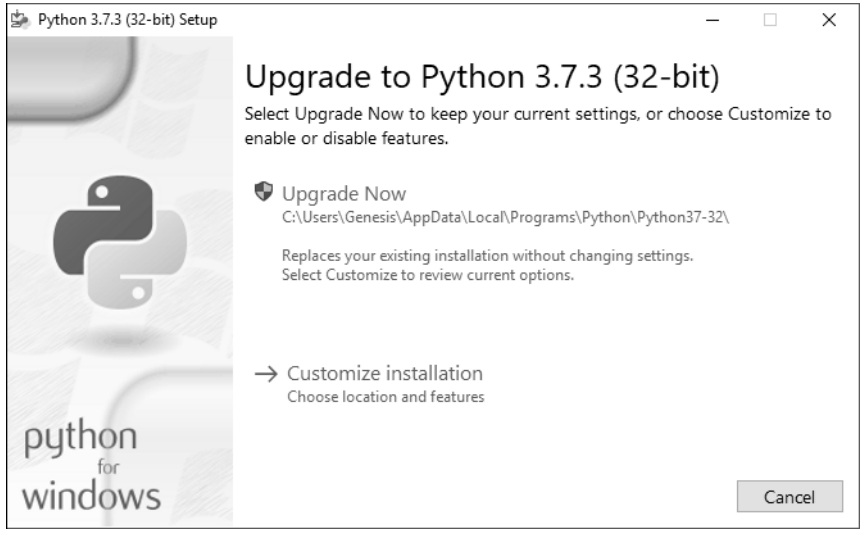
2.1 Pour Windows

Le système d'exploitation Windows requiert usuellement l'utilisation d'un installateur pour pouvoir installer un logiciel quel qu'il soit. Si vous disposez de Windows, vous devriez en avoir l'habitude. Python ne déroge pas à la règle.
Pour installer Python, vous devez donc aller sur le site officiel (https://www.python.org/downloads/) pour télécharger l'installateur adéquat. Comme vous pourrez le constater, on vous met en avant un accès rapide à la dernière version (au moment où ces lignes sont écrites, la 3.11.0), puis un accès aux dernières versions encore actives (actuellement la version 3.10 qui reçoit encore des corrections d'anomalies, puis les versions 3.9 à 3.7 qui reçoivent des corrections de sécurité uniquement).

Il est également possible de télécharger la toute dernière version de la branche 2.7 qui est en fin de vie (elle n'est plus mise à jour), car il existe encore de nombreux projets n'ayant pas encore migré.
Le support correctif dure 2 ans après la première sortie de la version et le support de sécurité dure 5 ans.
Pour notre part, nous vous conseillons la dernière 3.x, mais vous êtes libre d'installer celle que vous souhaitez ou même d'en installer plusieurs suivant vos contraintes, il n'y a pas d'objection à cela.
Une fois le téléchargement effectué, vous devez lancer l'installateur (et éventuellement passer quelques protections de votre système qui vous demande d'accorder votre confiance à cet installateur), pour observer l'écran suivant :

Comme vous pouvez le constater, il est possible de personnaliser l'installation en choisissant le chemin d'installation du logiciel ou en choisissant de ne pas sélectionner quelques fonctionnalités, mais nous ne le conseillons pas.
Nous vous recommandons en revanche de cocher la case **Add python.exe to PATH** afin de configurer la variable PATH du terminal pour rendre Python accessible plus facilement.

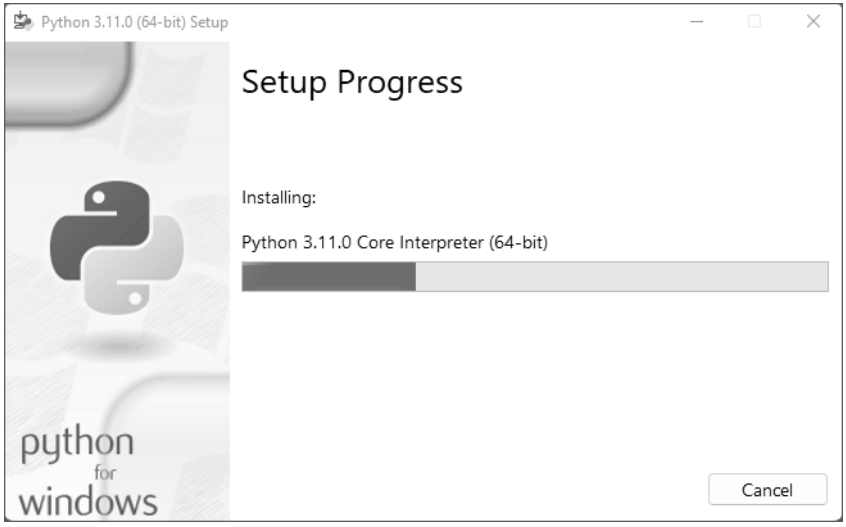
Si vous avez déjà une ancienne version de Python installée de la même branche (dans cet exemple, Python 3.7.2 est déjà installé), vous pourrez la mettre à jour à l'aide du même installeur :



Par contre, si vous avez déjà la version 3.7.1 et que vous installez la version 3.11, cette dernière ne viendra pas remplacer la précédente, mais s'installera à côté. Si vous souhaitez remplacer, il vous faudra donc désinstaller proprement la toute dernière version installée, ce qu'il est possible de faire en relançant l'installeur d'origine.

Nous vous encourageons à garder les installeurs sur votre PC, car ils pourraient devenir indisponibles au téléchargement si trop vieux.

Quel que soit le scénario, vous arriverez devant un écran vous montrant la progression de l'installation et vous n'aurez qu'à fermer la fenêtre une fois celle-ci terminée :



Vous êtes maintenant prêt à utiliser Python.

2.2 Pour Mac

Il faut savoir qu'une version de Python est déjà préinstallée sur Mac, car Mac OS X l'utilise pour ses propres besoins et Python est intégré à son propre cycle de développement. Cependant, si vous souhaitez une version différente de celle qui est déjà présente, vous pouvez l'installer, sachant qu' il n'y a pas de contre-indication à posséder plusieurs versions de Python sur la même machine.

Pour installer Python sur Mac OS X, la procédure à suivre est similaire à celle pour Windows. Il faut donc se rendre sur le site officiel (https://www.python.org/downloads/mac-osx/), télécharger un installateur correspondant à sa configuration et suivre les étapes.

Pour les utilisateurs de Mac, sachez que Python dispose d'une bonne intégration de ses spécificités, en particulier vis-à-vis de Objective-C, le langage de programmation avec lequel est développé Mac OS X, et Cocoa, interface de programmation de Mac OS X.

2.3 Pour GNU/Linux et BSD

Les différentes distributions libres utilisent nativement Python, notamment pour des parties sensibles. Python y est donc tout naturellement déjà installé, généralement sous la dernière version de la branche 2.x. Cependant, ici comme ailleurs, il n'y a pas d'objections à utiliser plusieurs versions de Python.
Le plus simple reste d'utiliser votre gestionnaire de paquets, ce qui peut se faire via un outil graphique, comme Synaptic pour Debian :

Il suffit alors de faire une recherche sur le mot-clé **`python`** pour voir les différentes versions (sur une ancienne Debian, ce sont Python 2.6, 2.7 et 3.2).
Par contre, tous les paquets python3-xxxxx que vous pouvez voir ici sont des bibliothèques tierces et non Python lui-même. Nous en parlerons plus tard dans ce chapitre.
Une fois les paquets souhaités sélectionnés, il ne manque plus qu'à les installer en cliquant sur le bouton **Appliquer**.
Notez que tout ceci peut se faire par la simple ligne de commande, toujours en utilisant votre gestionnaire de paquets qui peut être apt-get, aptitude, yum, emerge, pkg_add ou autre.
Voici par exemple pour une distribution Debian ou Ubuntu :

```
$ sudo aptitude install python3
```

Ceci ne permet cependant pas de choisir la version que l'on souhaite, à moins d'aller trouver des sources alternatives. Si l'on veut avoir la toute dernière version de Python, il faudra la plupart du temps passer par la compilation.

2.4 Par la compilation

Compiler Python n'est pas en soi une tâche très complexe. C'est par contre souvent une tâche imposée lorsque l'on ne travaille pas avec des conteneurs. En effet, en entreprise, on développe souvent des applications qui sont destinées à être hébergées. Il est alors impératif de travailler sur votre propre poste avec une version de Python qui soit la même que celle existante sur la machine de production.

Sous GNU/Linux, mais aussi sous d'autres systèmes, il est possible de compiler la version de Python que l'on souhaite. Après tout, Python n'est rien d'autre qu'un programme écrit en C. Pour ce faire, il faut aller télécharger le code source (https://www.python.org/downloads/source/), qui prend la forme d'une archive, puis décompresser celle-ci, se placer dans le répertoire ainsi obtenu et taper ces quelques commandes :

```
$ ./configure –prefix=/path/to/my/python/directory
$ make
$ sudo make altinstall
```

Notez que dans cette dernière ligne, nous n'utilisons pas la commande **`make install`**, qui aurait pour effet de remplacer votre Python système par le Python que vous compilez, ce qui pourrait avoir des conséquences indésirables voire désastreuses.

Notez également que vous choisissez lors de la configuration le chemin dans lequel vous placerez vos bibliothèques Python. En général, l'usage veut que l'on utilise **`/opt`**, mais il n'y a pas de règle, tout dépend des pratiques de votre entreprise ou votre expérience en la matière.

Si vous venez d'installer Python 3.5 par cette méthode, vous aurez alors maintenant accès à ce programme en l'appelant ainsi, depuis votre terminal :

```
$ python3.5
```

Par cette même méthode, vous pouvez installer les dernières versions (https://www.python.org/download/pre-releases/) de Python qui ne sont pas encore sorties (alphas ou betas), ce qui vous permet de les tester en avant-première !

Notons que, par cette méthode, toutes les bibliothèques de Python ne fonctionneront pas. En effet, lorsqu'elles ont besoin d'autres bibliothèques C, il faut effectuer des compilations croisées et utiliser les différents en-têtes de ces bibliothèques. C'est le cas par exemple pour faire fonctionner Curses, ReportLab (génération de fichiers PDF) ou encore PyUSB (accès aux ports matériels USB).

Dans ce cas-là, la commande **`./configure`** devra recevoir des arguments supplémentaires et vous devrez trouver un tutoriel en ligne pour vous indiquer la démarche, laquelle peut être plus ou moins complexe.

2.5 Pour un smartphone

Installer une machine virtuelle Python sur un smartphone est possible. Pour Android, la procédure est assez simple puisqu'il existe un produit dédié (http://qpython.com/), tout comme sur Windows Phone (https://apps.microsoft.com/store/detail/python-39/9P7QFQMJRFP7). Pour iOS, c'est une autre paire de manches (https://github.com/linusyang/python-for-ios) étant donné que l'utilisateur est enfermé dans un système sur lequel il n'a aucun contrôle.

3. Installer une bibliothèque tierce

Remarque

Si vous abhorrez le terminal, sachez que vous pouvez installer une bibliothèque tierce depuis votre IDE, ce qui sera probablement plus aisé pour vous.

3.1 À partir de Python 3.4

Pour installer une bibliothèque tierce, vous devez simplement connaître son nom. Celui-ci est généralement assez intuitif. Par exemple, la bibliothèque permettant de communiquer avec un serveur Redis s'appelle redis.
Il peut y avoir des variations. Par exemple, la bibliothèque de référence pour traiter du XML est lxml et, plus complexe, celle pour BeautifulSoup est bs4. En recherchant comment répondre à un besoin sur le Net ou sur PyPi (https://pypi.python.org/pypi), vous trouverez rapidement une bibliothèque de référence.
Sur des sujets plus confidentiels, il vous arrivera de trouver plusieurs petites bibliothèques. Vous pourrez alors les tester et choisir celle que vous utiliserez pour votre projet.
Sachez que vous pouvez aussi conduire une recherche directement depuis votre terminal :

```
$ pip search xml
$ pip search soup
```

Cela vous donnera une liste de bibliothèques accompagnée d'une courte description, à la manière de ce que font les gestionnaires de paquets sous Linux (lesquels sont écrits en Python, au passage).
Sachez que **`pip`** existe quel que soit votre système d'exploitation (vous devez être familier avec le terminal de votre système, cependant) et que depuis la version 3.4 de Python, il est installé automatiquement avec celui-ci. Si ce n'est pas votre cas, consultez la section suivante : Pour une version inférieure à Python 3.4.
`pip` est un outil formidable. Si vous utilisez une version de Python qui est celle du système, vous utiliserez alors la commande **`pip`** pour gérer les bibliothèques. Si vous utilisez une autre version, telle que Python 3.5, alors vous utiliserez la commande **`pip-3.5`**. Pour Python 3.3, ce sera **`pip-3.3`**. Dans les exemples suivants, il vous faudra prendre en compte cette particularité.
Cet outil vous permettra d'installer une bibliothèque à sa dernière version ainsi que toutes les bibliothèques dépendantes. En effet, il n'est pas rare qu'une bibliothèque de Python ait besoin d'une autre bibliothèque (ou de plusieurs) pour fonctionner. Par exemple, l'installation de redis se fait par cette commande :

```
$ pip install redis
```

On peut aussi choisir la version à installer :

```
$ pip install -Iv redis==2.10.5
```

Ou mettre à jour la bibliothèque à une version précise :

```
$ pip install -U redis==2.10.5
```

Ou à la dernière version :

```
$ pip install -U redis
```

Et on peut la désinstaller :

```
$ pip uninstall redis
```

Une fonctionnalité très importante permet d'obtenir la liste des bibliothèques déjà installées (quelle que soit la manière dont elles ont été installées) :

```
$ pip freeze
```

Ce que l'on peut mettre dans un fichier :

```
$ pip freeze > requirements.txt
```

Pour installer tous les paquets ainsi listés, il faut procéder ainsi :

```
$ pip install -r requirements/base.txt
```

Cette méthode est particulièrement utile dans le cadre d'un environnement virtuel ; nous y reviendrons.
Il est possible de retrouver des informations sur un paquet déjà installé :

```
$ pip show django-redis
---
Name: django-redis
Version: 4.3.0
Location: /path/to/my/env/lib/python3.4/site-packages
Requires: redis
```

On voit ici que le paquet **`django-redis`** a une dépendance vers **`redis`** : en l'installant, on installe automatiquement **`redis`**.
Mettre à jour ce paquet met à jour automatiquement les dépendances :

```
$ pip install -U django-redis
```

Si on ne veut pas mettre à jour les dépendances, on peut procéder ainsi :

```
$ pip install -U --no-deps django-redis
```

On peut aussi installer plusieurs bibliothèques en même temps :

```
$ pip install django-redis==4.3.0 bs4 lxml
```

Cette commande installera donc automatiquement redis s'il n'est pas installé, car il est déclaré comme dépendance.
Cette commande a cependant des limites. En effet, si vous installez une bibliothèque tierce qui utilise une bibliothèque C, vous devrez disposer des en-têtes C correspondants (paquets **`dev`** pour Debian ou **`devel`** pour Fedora). Il faut donc avoir un peu de pratique dans ce genre de situation pour savoir déjouer ces pièges.

Chapitre 4-4
L'algorithme k-means

1. Objectif du chapitre

Les chapitres précédents ont abordé des exemples de deux types d'algorithmes de Machine Learning : les algorithmes de régression et de classification. Ce chapitre porte sur l'algorithme k-means, appelé l'algorithme des k-moyennes en français, qui est un algorithme simple à comprendre et qui fait partie des algorithmes de clustering les plus connus et les plus utilisés.

L'algorithme k-means a été introduit par J. McQueen en 1967. C'est un algorithme non supervisé qui permet de répartir un ensemble de n observations en k clusters. L'objectif après l'application de l'algorithme k-means sur un jeu de données est que chaque cluster contienne des observations homogènes et que deux observations de deux clusters différents soient hétérogènes.

Les domaines d'application de l'algorithme k-means sont nombreux. Par exemple, il est très utilisé pour la segmentation des clients à des fins de marketing, ou encore pour l'isolation des motifs dans les images, car justement les images présentent souvent des régions homogènes, notamment en matière d'intensité lumineuse.

De manière générale, le succès de l'algorithme k-means et de ses versions réside dans sa simplicité et sa capacité à traiter des données de grande taille.

À la fin de ce chapitre, le lecteur aura abordé :

Le fonctionnement de k-means via des illustrations.

– Les étapes principales de l'algorithme k-means classique.
– L'application de l'algorithme k-means avec Scikit-learn.
– L'impact des valeurs extrêmes sur les performances de l'algorithme k-means.
– La recherche de la valeur optimale du paramètre *K* de l'algorithme k-means.
– Les avantages et les inconvénients ainsi que les variantes de l'algorithme k-means.

2. k-means du point de vue géométrique

Comme précisé au début de ce chapitre, l'algorithme k-means est très intuitif et simple à comprendre. Avant d'entrer dans les détails, il faut noter que k-means, comme tous les algorithmes de clustering, ne nécessite pas l'étiquetage des données, car c'est une procédure non supervisée.

De façon informelle, étant donné *n* observations à répartir sur *k*clusters, k-means choisit initialement, de manière aléatoire, *k* observations parmi les *n* observations, comme étant les centres des *k* clusters recherchés. Chacune des *n* observations sera associée au cluster dont le centre est le plus proche parmi les *k* centres choisis initialement. Une fois que toutes les observations sont associées à leurs clusters respectifs, le centre de chaque cluster est recalculé en fonction des observations qu'il contient. Puis, de nouveau, chacune des observations est associée au cluster dont le centre est le plus proche de cette observation par rapport à tous les centres des autres clusters. Ces opérations de recalcul des centres des clusters puis d'association des observations aux clusters les plus proches sont répétées jusqu'à ce qu'un critère d'arrêt soit atteint.

L'algorithme k-means utilise une fonction pour calculer les distances entre les observations et les centres des clusters. Ce calcul des distances peut être basé sur la distance euclidienne, la distance de Manhattan ou toute autre fonction permettant de mesurer la dissimilarité entre les observations.

Pour mieux comprendre cet algorithme de clustering, cette section déroule l'algorithme k-means sur un exemple simple. Soit six observations a, b, c, d, e et f à répartir sur deux clusters C_1 et C_2 ; supposons que la distance utilisée est la distance euclidienne classique. Ces six observations sont définies dans un espace à deux dimensions et leurs coordonnées sont indiquées dans le tableau suivant :

Axes	a	b	c	d	e	f
x	2	4	2	4	10	10
y	4	4	2	2	2	4

Figure 10-1 : un simple jeu de données avec leurs coordonnées en deux dimensions

Remarque

Pour rappel, la distance euclidienne entre deux observations A=(xA,yA) et B=(xB,yB) est calculée grâce à la formule :

$$\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

Avant de faire un traitement quelconque sur les données, il est toujours intéressant de les visualiser lorsque c'est possible. Dans cet exemple, les données sont définies dans un espace à deux dimensions, donc elles peuvent être facilement visualisées sur deux axes comme dans la figure 10-2 ci-dessous.

Remarque

Même lorsque les données sont définies dans un espace à grande dimension, supérieur à deux ou à trois dimensions, il existe des méthodes qui permettent de les visualiser en deux ou trois dimensions, avec une perte d'informations qu'on espère minime. Ces méthodes sont appelées les méthodes de réduction de domaines. Le chapitre Analyse en composantes principales présente l'analyse du même nom, qui est l'une des méthodes de réduction de domaines les plus connues et qui permet d'avoir une vue en deux dimensions des données définies initialement dans un espace à grande dimension.

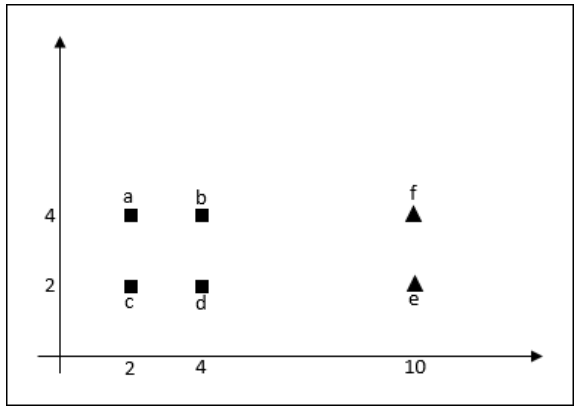


Figure 10-2 : représentation graphique en deux dimensions des données

Ce graphique montre clairement que les observations a, b, c et d, représentées par des carrés, sont très proches entre elles au sens de la distance euclidienne, par rapport aux deux observations e et f. Également, les deux dernières observations, représentées par des triangles, sont très proches entre elles.

Pour cet exemple, en se basant donc sur la distance euclidienne, un algorithme de clustering efficace proposerait certainement de répartir ces six observations dans les deux clusters C_1 et C_2 comme dans la figure 10-3 ci-dessous.

En effet, avec la distance euclidienne, cette répartition est optimale. La section suivante définit de façon plus formelle la notion de solution optimale pour un algorithme k-means et pour un nombre de clusters fixe.

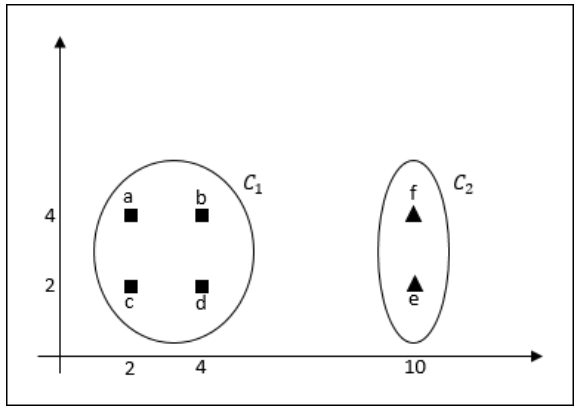


Figure 10-3 : répartition des six points dans les deux classes C_1 et C_2

En suivant les étapes classiques de l'algorithme k-means, le résultat optimal de la figure 10-3 peut être obtenu comme suit :

1. L'algorithme k-means commence initialement par sélectionner de façon aléatoire deux observations parmi les six observations disponibles. Les deux observations ainsi sélectionnées vont être considérées comme les centres des deux clusters recherchés C_1 et C_2. Ici, nous supposons que l'algorithme k-means recherche un nombre de clusters qui est égal à 2.

   Dans cet exemple, supposons que les deux observations a et d sont sélectionnées aléatoirement. Ces deux observations vont être considérées comme les centres respectifs des clusters C_1 et C_2. L'algorithme k-means calcule les distances entre chacune des six observations avec les centres a et d. Les résultats sont reportés dans le tableau ci-dessous :

Les centres des clusters	a	b	c	d	e	f
Centre de C_1=a=(2,4)	0	2	2	2.8284	8.2462	8
Centre de C_2=d=(4,2)	2.8284	2	2	0	6	6.3245

Figure 10-4 : distances entre les observations et les centres de C_1 et C_2

2. Une fois que k-means dispose de toutes les distances entre toutes les observations et les deux centres a et d, il procède à l'association entre les observations et les clusters. Par exemple, l'observation e va être associée au cluster C_2, puisqu'elle est plus proche du centre de C_2 que du centre de C_1. À la suite de cette étape, les deux clusters C_1 et C_2 vont être constitués comme suit $C_1 = \{a, b, c\}$ et $C_2 = \{d, e, f\}$

   Lorsqu'une observation est à la même distance des clusters C_1 et C_2, alors elle est affectée à l'un de ces deux clusters de manière aléatoire. Dans notre exemple nous avons affecté de manière arbitraire les deux observations b et c au cluster C_1.

   La figure suivante présente graphiquement les deux clusters C_1 et C_2 obtenus à la suite de cette étape :

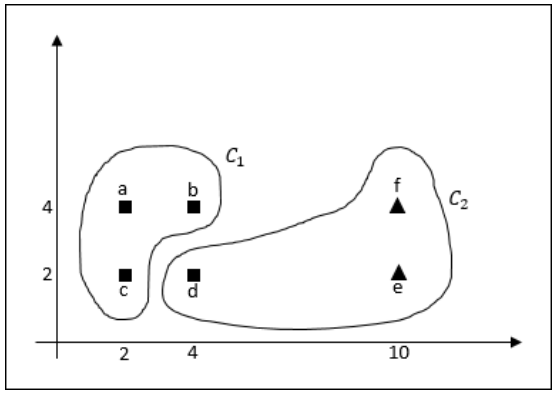


   Figure 10-5 : affectation des observations aux clusters C_1 et C_2 après la première itération

3. Lors de la première étape, l'algorithme k-means a choisi de façon aléatoire a et d en tant que centres des deux clusters recherchés. À l'étape 2, les deux clusters C_1 et C_2 ont été concrètement construits.

À cette étape, l'algorithme k-means calcule le centre du cluster C_1 en utilisant les observations a, b et c et calcule le centre du cluster C_2 en utilisant les observations d, e et f.

Ainsi : le centre de $C_1 = \left(\frac{1}{3}(2+2+4), \frac{1}{3}(4+4+2)\right) = (2.66, 3.33)$

le centre de $C_2 = \left(\frac{1}{3}(4+10+10), \frac{1}{3}(2+2+4)\right) = (8, 2.66)$

Les centres des deux clusters C_1 et C_2 sont représentés avec le symbole étoile dans la figure 10-6 ci-dessous :

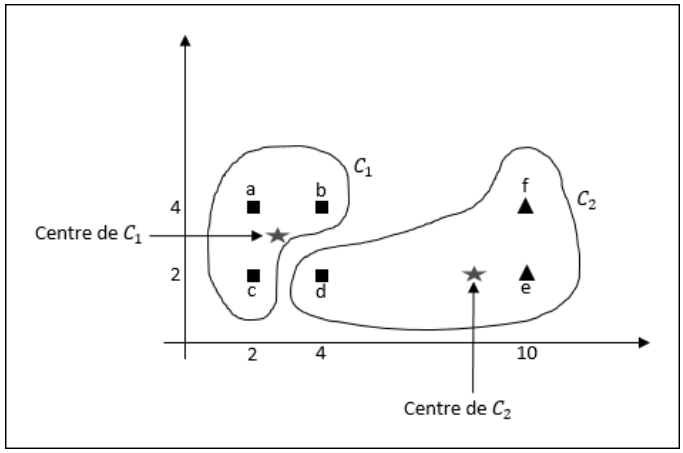


Figure 10-6 : les centres des clusters C_1 et C_2

4. À cette étape, les six observations sont réparties de nouveau sur les deux clusters C_1 et C_2 en se basant sur les nouveaux centres calculés à l'étape précédente. Les distances entre les observations et les centres de C_1 et C_2 sont reportées dans le tableau suivant :

Les centres des clusters	a	b	c	d	e	f
Centre de C_1= (2.66,3.33)	0.9404	1.4981	1.4847	1.8879	7.4595	7.3505
Centre de C_2= (8,2.66)	6.1478	4.2184	6.0361	4.0540	2.1060	2.4074

Figure 10-7 : distances entre les observations et les centres de C_1 et C_2