

La data visualisation avec Python

Table des matières

I. Généralités	3
II. Exercice : Quiz	5
III. Bibliothèques pour la data visualisation en Python	6
IV. Exercice : Quiz	12
V. Essentiel	13
VI. Auto-évaluation	13
A. Exercice	13
B. Test	13
Solutions des exercices	14

I. Généralités

Durée : 1 h

Environnement de travail : Pc connecté à Internet

Contexte

Nous avons tous déjà entendu cette citation de Confucius : « *Une image vaut mille mots* ». De nos jours, chaque institution, entreprise, organisation et aussi association possèdent toutes des données en masse à cause de leurs archives et aussi à cause des collectes de données qu'ils font.

Par rapport à cette citation de Confucius, ces données sont équivalentes aux « *mille mots* » et l'image consiste à la visualisation de ces données. Parfois, on ne se rend pas compte du sens de nos données si on ne les a pas sous forme visuelle.

En effet, avec une bonne structure visuelle de ces volumes de données, l'exploitation et l'interprétation se fera plus facilement. Ces données possèdent en elles beaucoup d'informations à représenter pour améliorer et faciliter la prise de décision.

La data visualisation ou visualisation des données est une étape capitale de la data science. Il s'agit de rendre accessible et compréhensible les masses de données en les représentant graphiquement. Il est nécessaire de savoir les présenter d'une manière simple et convaincante pour que même des publics non techniques puissent comprendre.

Ce cours est une introduction à la visualisation de données en Python. On parlera notamment du langage de programmation Python, ensuite on verra ce qu'est la data visualisation et enfin les outils nécessaires pour faire de la data visualisation en Python. Ce cours vous permettra de concevoir des représentations visuelles simples et précises afin de rendre compréhensible un jeu de données rapidement.

Le langage de programmation Python

Python a été créé en 1989 par Guido Van Rossum et plusieurs contributeurs bénévoles. Il a été créé pour automatiser les éléments les plus difficiles de l'écriture de scripts afin de réaliser les prototypes d'applications. Maintenant il est très populaire sur le marché des développeurs qui l'utilisent pour créer des applications.

Par définition c'est un langage de programmation orienté objet et interprété. Il n'a pas besoin d'être compilé pour fonctionner, ce qui explique aussi qu'il possède des classes et des objets contrairement aux autres langages comme C.

Python favorise une programmation structurée et orientée objet. Parmi ses caractéristiques, il est doté d'un fort typage dynamique, d'une automatisation de la gestion de la mémoire et d'un système de gestion des exceptions. Il est multiparadigme, multiplateforme et portable sur tous les systèmes d'exploitation comme MacOS ainsi que Windows et ses variantes.

Par ces aspects, on peut dire que Python est un langage de programmation polyvalent. C'est un langage conçu pour être « *full stack* », c'est-à-dire qu'on peut avoir avec lui de multiples applications, contrairement à d'autres langages comme PHP qui a été conçu pour fonctionner seulement dans une version web.

Selon une enquête de *Stack Overflow Developer Survey*, en 2018, Python a fait une croissance des plus rapides parmi les langages de programmation. Son aspect polyvalent fait de lui une meilleure option pour la programmation Big Data. De plus, les bibliothèques pour les analyses et les manipulations de données les plus utilisées en Big Data, comme NumPy, Pandas, SciPy, sont toutes basées sur Python. Il y a aussi le framework d'apprentissage automatique le plus populaire comme TensorFlow qui est aussi écrit en Python.

On peut dire alors que Python évolue de plus en plus dans l'écosystème Big Data, ce qui fait de lui le meilleur choix de langage de programmation pour faire des analyses de gros volumes de données.

Définition La data visualisation

La data visualisation, ou visualisation de données, ou encore « *DataViz* », fait partie de la discipline des sciences de données. Il s'agit d'une méthode permettant de représenter graphiquement et visuellement des données statistiques qualitatives et quantitatives afin de faire apparaître des liens entre ces données.

C'est un outil efficace qui facilite et accélère la prise de décision. On accède directement à l'essentiel et grâce à elle il y a une simplification de la diffusion des informations. Par ailleurs, c'est une pratique que l'on côtoie au quotidien sans s'en rendre compte, comme pour la télévision par exemple, typiquement avec les sondages qui sont de puissants outils de communication.

Structure d'une visualisation

En générale, une visualisation est composée :

- D'un élément visuel
- De coordonnées
- D'un contexte
- D'une échelle

Exemple

Prenons par exemple un « *nuage de points* », l'élément visuel qui représente les données est la position des points.

La nature de l'échelle varie selon le type des données à représenter. Voici un tableau qui montre quelques types d'échelle :

TYPE DE VARIABLES / DONNÉES	ÉCHELLE
Quantitative	Linéaire ou logarithmique
Catégorique	Catégorique
Temps	Temporelle

Modèle et typologie de représentation

La forme est importante dans la visualisation. Avec plusieurs options de visualisation, il est quand même difficile de choisir le graphique qui correspond le mieux à nos données.

Afin de bien refléter ces représentations, et de bien passer l'information à émettre, il est primordial de connaître les types de représentation à utiliser pour être cohérent avec les données à représenter. Cela permet de faciliter la lecture et l'interprétation des visuels d'un simple coup d'œil.

En effet, le choix du graphique à utiliser dépend des types des données. Il existe de nombreux types de visualisation à connaître pour être efficace et pertinent dans la visualisation des données :

- La visualisation temporelle
- La visualisation des proportions
- La visualisation des relations
- La visualisation des différences

Connaître ces types de visualisation est important. Présentons brièvement alors ces types de visualisations pour aider à la prise de décision de nos choix de graphique.

- **La visualisation temporelle**

Le temps fait partie de la vie quotidienne et les choses évoluent dans le temps. La visualisation temporelle permet alors d'avoir une représentation générale des données sur une plage de temps donné.

Voici quelques exemples de graphiques pour représenter une visualisation temporelle :

- Gantt Chart (Pour les chronogrammes d'activités)
- Line Graph
- Histogramme

- **La visualisation des proportions**

Avec le traitement par proportion, on peut avoir le maximum, le minimum et une distribution globale d'un ensemble de données. La visualisation des proportions permet de représenter alors les parties d'un tout. Chaque valeur a ses significations.

Voici quelques exemples de graphiques pour représenter une visualisation des proportions :

- Camembert
- Bubble Chart
- Bubble Map

- **La visualisation des relations**

Les données entretiennent des relations entre eux. Savoir rechercher ces relations nécessite une grande réflexion. Cette relation peut révéler des informations importantes.

Voici quelques exemples de graphiques pour représenter une visualisation des relations :

- Network Diagram
- Nuage de points
- Radar Chart

- **La visualisation des différences**

La visualisation des différences permet la comparaison des variables importantes d'une portion de données.

Voici quelques exemples de graphiques pour représenter une visualisation des différences :

- Bar Chart
- Span Chart
- Population Pyramid

Exercice : Quiz

[solution n°1 p.15]

Question 1

Python a été développé par Guido Van Rossum.

- ☐ Vrai
- ☐ Faux

Question 2

Python est un langage de programmation polyvalent.

- ☐ Vrai
- ☐ Faux

Question 3

La data visualisation fait partie de la data science.

- ☐ Vrai
- ☐ Faux

Question 4

La forme est importante en data visualisation.

- ☐ Vrai
- ☐ Faux

Question 5

La data visualisation sert à représenter visuellement les données.

- ☐ Vrai
- ☐ Faux

III. Bibliothèques pour la data visualisation en Python

Une bibliothèque ou librairie est un regroupement de fonctions utilitaires prêtes à être utilisées sans les réécrire. Pour réaliser des visualisations de données en Python, il est nécessaire d'utiliser des bibliothèques conçues pour cela. Il existe plusieurs bibliothèques pour faire des visualisations de données en Python, mais regardons celles qui sont les plus utilisées et populaires.

Installation de Python et Anaconda :

Il existe une façon plus simple et plus rapide pour installer Python et ses librairies. Il s'agit d'installer « Anaconda ». Anaconda est une distribution Python et un environnement de développement pour la science des données, cela signifie qu'en installant Anaconda, diverses librairies sont installées aussi avec lui.

Pandas

Avant de faire une visualisation, il faut d'abord savoir manipuler et traiter les données. Le nom « *Pandas* » ne fait pas référence à l'animal mais il s'agit en fait de « *Panel Data* » et « *Python Data Analysis* ».

Pandas est une librairie de Python open source qui permet de manipuler et de faire des analyses de données d'une façon plus simple. C'est Wes McKinney qui a développé cette librairie en 2008. Pandas est basé sur la bibliothèque la plus populaire « *NumPy* » ce qui fait sa force. Elle offre plusieurs structures de données et opérations pour traiter les données.

Pandas est installé avec Anaconda. C'est-à-dire qu'après avoir installé Anaconda, on aura accès aussi à la librairie Pandas sans installer d'autres logiciels. Sinon, Pandas peut être installé aussi manuellement sur l'environnement de déploiement en lançant la commande suivante :

```
1 >> pip install pandas
```

Remarque

Pour pouvoir utiliser Pandas et avoir accès à ces opérations, il faut l'importer. L'importation de Pandas se fait par cette commande :

```
1 import pandas as pd
```

Cette ligne de code signifie qu'on importe la bibliothèque Pandas et qu'on va utiliser « *pd* » à sa place dans le reste du code. Le mot « *pd* » est donc un alias qu'on a donné pour Pandas. Cela permet d'écrire moins de code.

Il existe deux principales structures de données sur Panda pour la manipulation et le traitement de données, ce sont : les séries et les dataframes.

Les séries :

Une série est un tableau qui peut stocker tous types de données : entier, nombres à virgule, chaînes de caractère, etc. Il possède des étiquettes représentant l'index de la série. Par analogie à un tableau Excel, une série est équivalente à une colonne.

Voici la syntaxe Python pour initialiser une série :

```
1 import pandas as pd
2 var = pd.Series(data, index=index)
```

La première ligne de code est l'importation de Pandas. Après avoir importé Panda, on a accès à la structure de données « *Series* ». On déclare une variable « *var* » pour initialiser une série. Dans cet exemple, on montre l'utilisation de l'alias à la place de *panda*. Et ici, on peut utiliser les différentes structures de données en Python à la place de « *data* », cela peut être un dictionnaire ou autre. L'index est une étiquette qui correspond à une ligne de la série. Pour mieux comprendre, illustrons ces explications par un exemple.

Exemple

Dans cet exemple, utilisons un dictionnaire comme data. En écriture Python cela donne :

```
1 import pandas as pd
2
3 ages = {"Marie": 19, "Mathieu": 12, "Pascal": 15}
4 serie = pd.Series(ages, index= ["Marie", "Mathieu", "Pascal" ])
5 print(serie)
```

La sortie de ce code est :

Marie	19
Mathieu	12
Pascal	15

La dataframe

La dataframe est une structure de données de Pandas, c'est d'ailleurs la plus puissante et la plus utilisée. Elle est bidimensionnelle, c'est-à-dire que la représentation des données se fait en lignes et en colonnes. Un dataframe peut aussi contenir de nombreux types de données ou mêmes structures de données différentes.

Voici la syntaxe pour utiliser un dataframe avec Python :

```
1 import pandas as pd
2 dataframe1 = pd.DataFrame(data, index=index , columns=colonne)
```

L'« *index* » représente les lignes et « *columns* » les colonnes. Regardons un exemple pour mieux comprendre.

Exemple

```
1 import pandas as pd
2 data1 = {"Nom": ["John", "Perez", "Colombe", "Michel"], "Age": [11, 24, 25, 69]}
3 df = pd.DataFrame(data1)
4 print(df)
```

La sortie de ce code est :

	Nom	Âge
0	John	11
1	Perez	24
2	Colombe	25
3	Michel	69

MATPLOTLIB

Après avoir vu un peu de notion sur les traitements de données, entrons maintenant dans la représentation graphique.

Matplotlib est aussi une bibliothèque Python permettant de produire des représentations graphiques de qualité. Il est parmi les bibliothèques les plus utilisées pour la visualisation des données en Python.

On peut générer plusieurs types de graphiques avec Matplotlib comme des histogrammes, des nuages de points, des spectres, etc. Il existe aussi des options pour des graphes en 3D. C'est une bibliothèque vraiment puissante, multiplateforme et on peut produire des graphiques en différents formats.

Matplotlib fonctionne généralement avec la bibliothèque Numpy, qui lui donnera aussi accès à des fonctions.

On peut installer la bibliothèque manuellement dans l'environnement de développement par la commande :

```
1 >> pip install matplotlib
```

Après l'avoir installée, on peut l'utiliser en l'important dans le fichier qui va contenir le code.

L'importation de la bibliothèque se fait comme suit :

```
1 - import matplotlib.pyplot as plt
```

Comme pour l'importation de Pandas, on donne aussi un alias « *plt* » pour écrire moins de code dans le reste du codage.

Remarque

Comme Matplotlib est généralement utilisé avec NumPy, il faut donc aussi importer cette bibliothèque. Illustrons l'utilisation de Matplotlib par un exemple plus concret pour mieux comprendre.

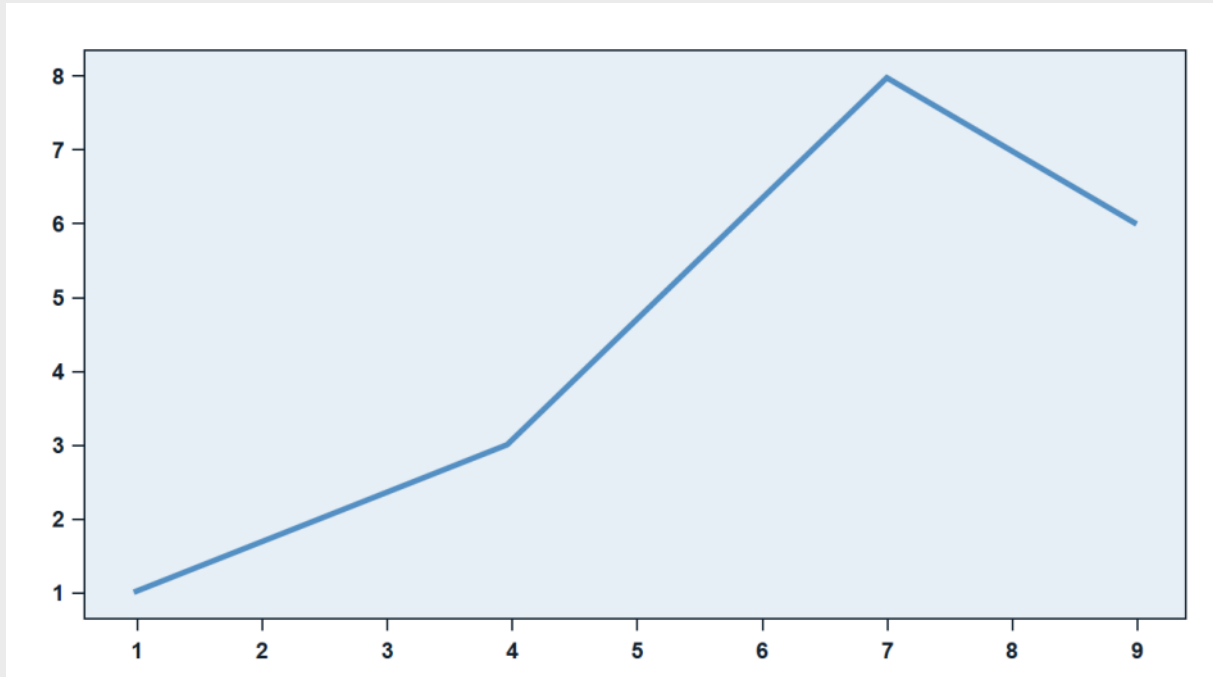
Exemple

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.array([1, 4, 7, 9])
5 y = np.array([1, 3, 8, 6])
6 plt.plot(x, y)
7
```



```
8 plt.show()
```

Le résultat de ce script sera un graphique :



On a importé d'abord les deux bibliothèques Numpy et Matplotlib pour avoir accès à ses fonctions. On a déclaré une variable `x` qui va contenir les données sur l'axe des abscisses dans la représentation et une variable `y` pour l'axe des ordonnées. On peut les remplacer selon les données à représenter mais ici, on a juste utilisé la fonction « `array` » de Numpy pour initialiser un tableau contenant des chiffres.

La fonction « `plot` » de Matplotlib permet alors le traçage de la courbe. Et la fonction « `show()` » permet ainsi de l'afficher.

Remarque

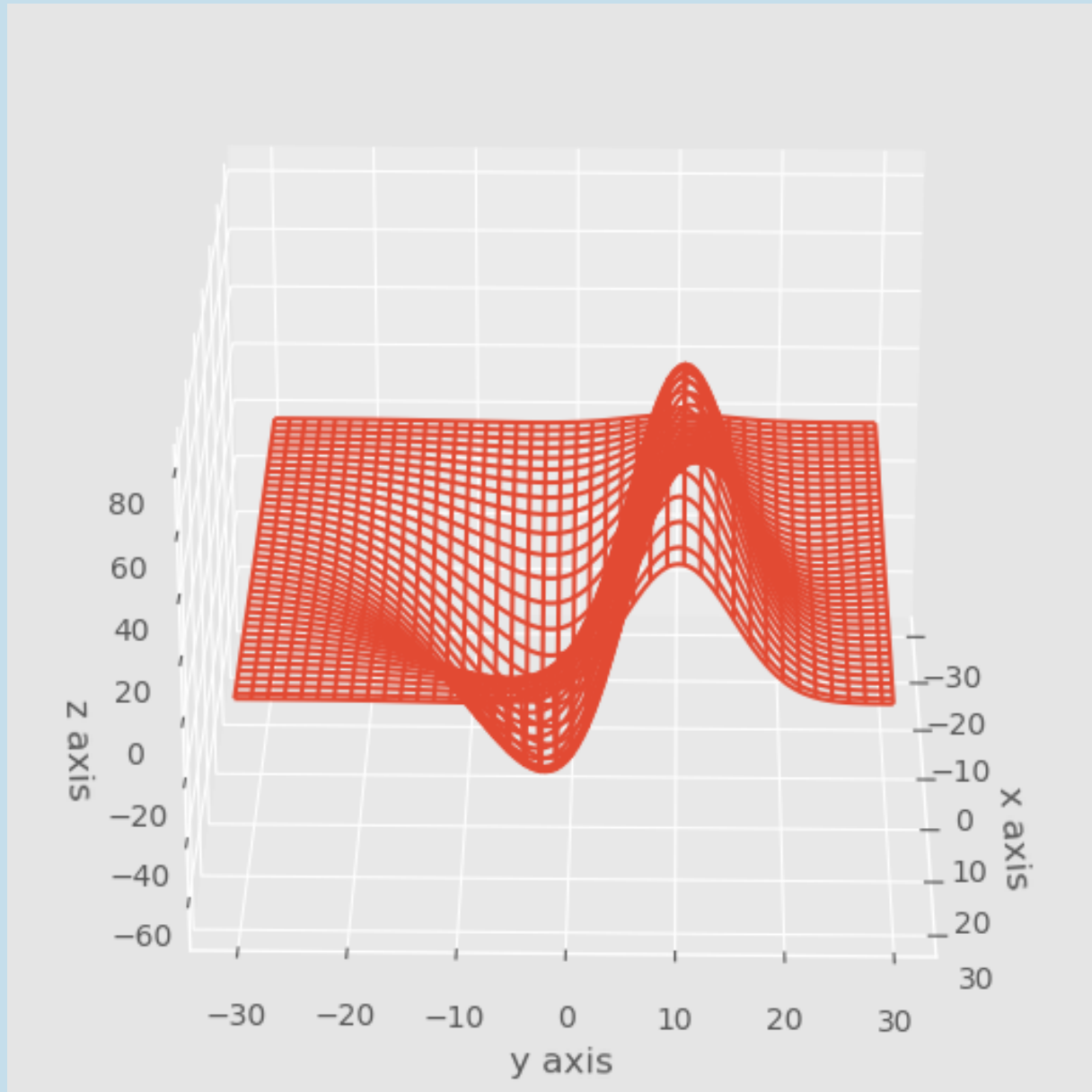
On peut donner des titres, des légendes et même colorier les courbes avec d'autres fonctions offertes par la bibliothèque. Sans entrer dans les détails, regardons un autre exemple plus complexe pour vous donner un aperçu de ce que Matplotlib est capable de faire.

À l'aide des commandes suivantes :

```
1 import matplotlib.pyplot as plt
2 from matplotlib import style
3 from mpl_toolkits.mplot3d import axes3d
4
5 style.use('ggplot')
6
7 fig_1 = plt.figure()
8 axe_1 = fig_1.add_subplot(111, projection='3d')
9
10 x, y, z = axes3d.get_test_data()
11
12 axe_1.plot_wireframe(x, y, z, rstride=3, cstride=3)
13
14 axe_1.set_xlabel('x axis')
15 axe_1.set_ylabel('y axis')
```

```
16 axe_1.set_zlabel('z axis')
17
18 plt.show()
```

On peut voir qu'on a utilisé quelques fonctions dans cet exemple, la sortie de ce code est alors le graphique en 3D suivant :



SEABORN

Seaborn est aussi une bibliothèque pour la visualisation des données. C'est une bibliothèque qu'on a ajouté à Matplotlib pour remplacer certains réglages et ajouter des nouvelles fonctionnalités. Elle résout les complexités de Matplotlib.

Par exemple, Matplotlib ne peut pas interagir avec les dataframes de Pandas. Et c'est pour résoudre ces défauts que Seaborn a été ajouté. C'est-à-dire qu'elle utilise toujours Matplotlib en coulisse. Elle possède aussi diverses fonctionnalités pour les représentations graphiques : elle offre plusieurs types de visualisation.

Comme toutes bibliothèques, elle requiert aussi une installation pour pouvoir l'utiliser. L'installation est toujours la même en changeant juste le nom de la librairie à installer, comme la commande suivante :

```
1 >> pip install seaborn
```

Et pour pouvoir l'utiliser dans le fichier contenant nos codes, il faut aussi l'importer.

```
1 import seaborn as sns
```

Pour mieux comprendre sa syntaxe et aussi quelques-unes de ses fonctionnalités, regardons un exemple.

Exemple

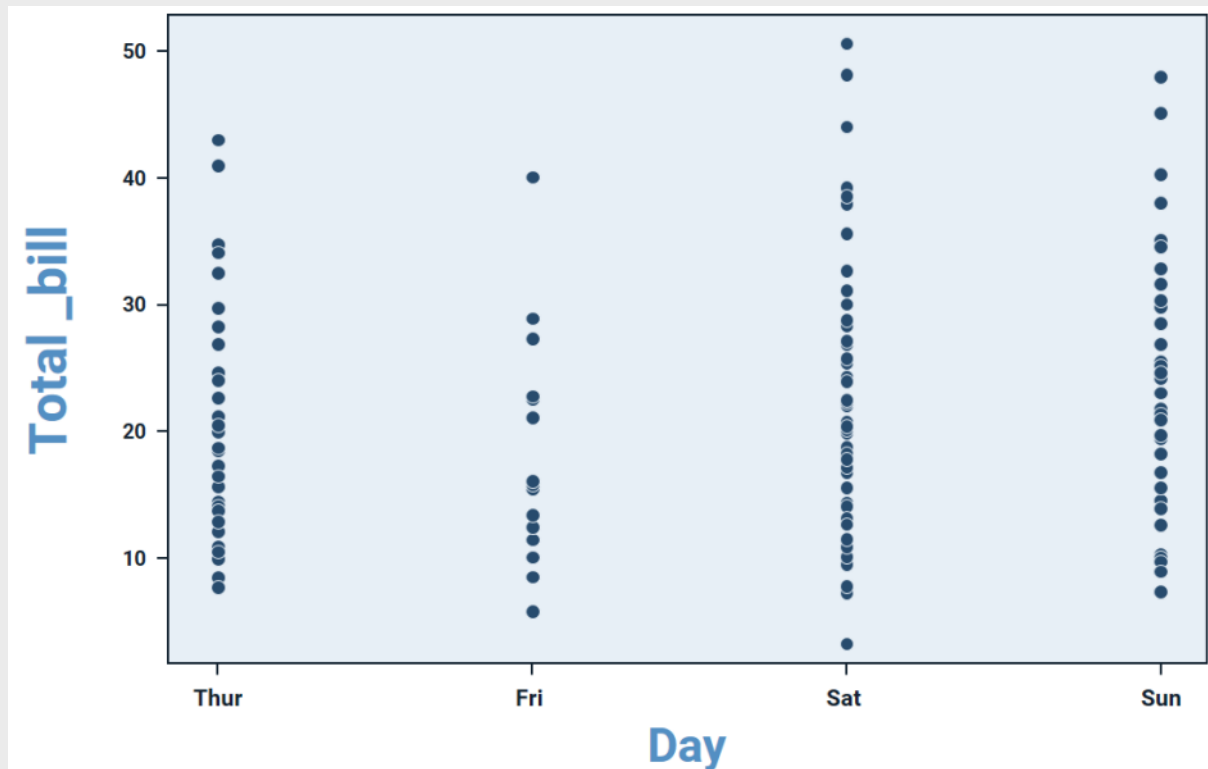
Dans cet exemple, on va utiliser un générateur de données de seaborn pour nous donner des datas à traiter.

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 tip_1 = sns.load_dataset("tips")
5 sns.scatterplot(x="day", y="total_bill", data=tip_1)
6
7 plt.show()
```

Ici, la fonction « `load_dataset` » permet de charger des données fakers de Seaborn nommés « `tips` ». On a stocké ses données dans la variable « `tip_1` ».

La fonction « `scatterplot` » nous permet ensuite de tracer les courbes en lui donnant des arguments comme la valeur des axes des abscisses « `x` » et des ordonnées « `y` » ainsi que le data à prendre. « `day` » et « `total_bill` » sont des colonnes dans les données « `tips` ».

Et enfin pour l'afficher, on utilise la fonction `plt.show()`. Après avoir exécuté ce script, on aura alors un graphique comme ceci :



Remarque

Néanmoins, on peut changer le type de graphe en utilisant les fonctions correspondantes dans la bibliothèque. Et on peut mettre aussi des légendes, il existe des propriétés pour cela.

Exercice : Quiz

[solution n°2 p.15]

Question 1

Qu'est-ce qu'une bibliothèque en Python ?

- ☐ C'est une librairie regroupant des fonctions prêtes à l'emploi
- ☐ C'est une variable
- ☐ C'est une structure de données

Question 2

Le mot « *Pandas* » est tiré de l'animal.

- ☐ Vrai
- ☐ Faux

Question 3

Quelles sont les deux principales structures de données de Pandas ?

- ☐ Les dataframes
- ☐ Les tuples
- ☐ Les séries

Question 4

Qu'est-ce que Matplotlib ?

- ☐ C'est une bibliothèque utilisée pour des représentations graphiques
- ☐ C'est une fonction
- ☐ C'est une méthode

Question 5

Seaborn est une bibliothèque.

- ☐ Faux
- ☐ Vrai

V. Essentiel

Python fait partie des langages de programmation les plus matures. Il est possible avec ce langage de faire diverses analyses et traitement de données. La visualisation des données fait partie de ces traitements de données, en effet, cela permet l'exploration des données dans le but de communiquer des informations d'une façon visuelle. La data visualisation est un outil simple, efficace et rapide qui permet d'améliorer et d'aider sur la prise de décision.

Même en étant un langage scientifique, Python seul ne peut pas faire toutes ces tâches, il possède plusieurs bibliothèques pour être plus performants dans ces traitements. On peut citer alors Pandas, qui est une bibliothèque assez puissante et incontournable pour les manipulations ainsi que les traitements de données. Elle offre les structures de données les plus puissantes comme les séries et les dataframes.

Pour la data visualisation, on a vu deux des bibliothèques les plus populaires et les plus utilisées en Python. Ce sont Matplotlib et Seaborn. Matplotlib a vu le jour en premier, c'est une bibliothèque basée sur une autre bibliothèque nommée « NumPy ». Elle est assez forte pour pouvoir faire des visualisations de données complexes en Python. Elle peut produire divers types de graphiques, même des 3D. Elle offre aussi plusieurs fonctions à exploiter pour des visualisations plus avancées.

Néanmoins, Matplotlib possède quelques défauts comme son incapacité à traiter les dataframes de Pandas, par exemple. C'est pour cela que Seaborn, qui est aussi une bibliothèque pour les visualisations de données, a été créée et ajoutée à Matplotlib. Elle fonctionne toujours sur Matplotlib mais résout les problèmes de ce dernier en offrant d'autres fonctionnalités. On peut donc dire que Seaborn a été créée pour compléter les lacunes de Matplotlib.

VI. Auto-évaluation

A. Exercice

Vous êtes un data analyste dans une entreprise. On vous demande de suivre le flux de voiture garée dans le parking durant les jours de la semaine. Sachant que pour une semaine 1, on a les données suivantes :

Lundi = 5, mardi = 9, mercredi = 7, jeudi = 10, vendredi = 15, samedi = 5 et dimanche = 0.

Question 1

[solution n°3 p.16]

Représenter ces données graphiquement pour voir quel jour on a le plus de voitures garées afin de mieux gérer le parking à l'avenir.

Question 2

[solution n°4 p.17]

Représentons ces données en barres pour bien regarder les données de chaque jour.

B. Test

Exercice 1 : Quiz

[solution n°5 p.18]

Question 1

L'importation des bibliothèques est nécessaire.

- ☐ Vrai
- ☐ Faux

Question 2

La définition de l'alias d'une bibliothèque est importante.

- ☐ Faux
- ☐ Vrai

Question 3

Quelle fonction est utilisée pour tracer une courbe ?

- ☐ plot()
- ☐ show()
- ☐ load_dataset()

Question 4

Quelle fonction est utilisée pour afficher une figure ?

- ☐ plot()
- ☐ show()
- ☐ load_dataset()

Question 5

On peut personnaliser une figure faite en Matplotlib et Seaborn.

- ☐ Vrai
- ☐ Faux

Solutions des exercices

Exercice p. 5 Solution n°1**Question 1**

Python a été développé par Guido Van Rossum.

☒ Vrai

☐ Faux


 Guido Van Rossum a inventé Python en 1989 avec d'autres contributeurs.

Question 2

Python est un langage de programmation polyvalent.

☒ Vrai

☐ Faux


 Python est multiplateforme, on peut faire avec lui tous types de programmation que ce soit web, système ou réseau, et même des applications mobiles.

Question 3

La data visualisation fait partie de la data science.

☒ Vrai

☐ Faux


 La data visualisation est une étape dans la data science.

Question 4

La forme est importante en data visualisation.

☒ Vrai

☐ Faux


 Pour être cohérent et pertinent dans la représentation graphique des données, il faut savoir bien choisir le type de graphisme à utiliser.

Question 5

La data visualisation sert à représenter visuellement les données.

☒ Vrai


☐ Faux

 La data visualisation, ou visualisation des données, permet de représenter les données à l'aide des graphismes.

Exercice p. 12 Solution n°2


Question 1

Qu'est-ce qu'une bibliothèque en Python ?

- ☒ C'est une librairie regroupant des fonctions prêtes à l'emploi
- ☐ C'est une variable
- ☐ C'est une structure de données
-  Une bibliothèque est aussi appelée librairie.


Question 2

Le mot « *Pandas* » est tiré de l'animal.

- ☐ Vrai
- ☒ Faux
-  Pandas veut dire Panel Data, c'est une bibliothèque Python faite pour manipuler les données.


Question 3

Quelles sont les deux principales structures de données de Pandas ?

- ☒ Les dataframes
- ☐ Les tuples
- ☒ Les séries
-  Les séries et les dataframes sont les structures de données les plus puissantes pour le traitement et les manipulations des données en Pandas.


Question 4

Qu'est-ce que Matplotlib ?

- ☒ C'est une bibliothèque utilisée pour des représentations graphiques
- ☐ C'est une fonction
- ☐ C'est une méthode
-  Matplotlib est une bibliothèque Python qui permet de tracer des graphes.

Question 5

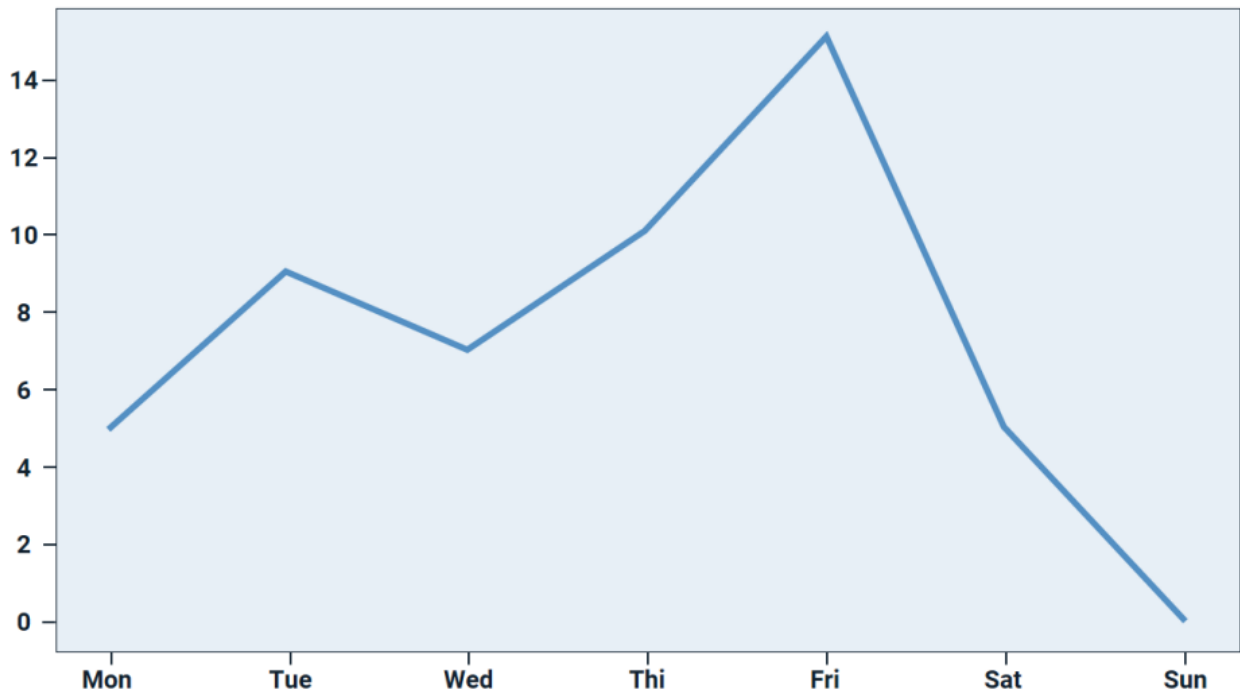
Seaborn est une bibliothèque.

- ☐ Faux
- ☒ Vrai
-  Seaborn est une bibliothèque ajoutée à Matplotlib pour fournir de nouvelles fonctionnalités et régler quelques problèmes de Matplotlib.

On utilise les commandes suivantes :

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 axe_x = ["Mon", "Tue", "Wed", "Thi", "Fri", "Sat", "Sun"]
5 axe_y = [5, 9, 7, 10, 15, 5, 0]
6 plt.plot(axe_x, axe_y)
7
8 plt.show()
```

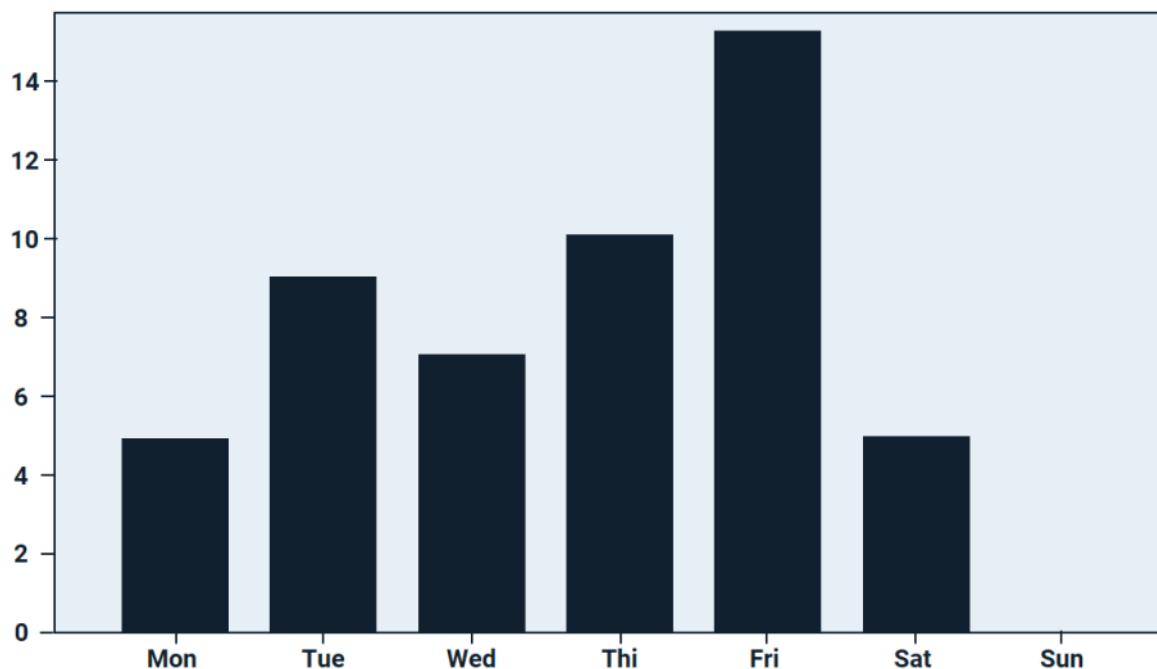
Le résultat est :



p. 13 Solution n°4

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 axe_x = ["Mon", "Tue", "Wed", "Thi", "Fri", "Sat", "Sun"]
5 axe_y = [5, 9, 7, 10, 15, 5, 0]
6 plt.bar(axe_x, axe_y)
7
8 plt.show()
```

Le résultat est :




Exercice p. 13 Solution n°5

Question 1

L'importation des bibliothèques est nécessaire.

☒ Vrai

☐ Faux


 Il est nécessaire d'importer les bibliothèques à utiliser dans le fichier contenant les codes afin de pouvoir utiliser les fonctions qu'elles possèdent.

Question 2

La définition de l'alias d'une bibliothèque est importante.

☒ Faux

☐ Vrai

 La définition de l'alias est juste faite pour éviter d'écrire plus de code.

Question 3

Quelle fonction est utilisée pour tracer une courbe ?

☒ plot()

☐ show()

☐ load_dataset()

Q plot() est une fonction de Matplotlib qui permet le traçage des courbes.

Question 4

Quelle fonction est utilisée pour afficher une figure ?

- ☐ plot()
- ☒ show()
- ☐ load_dataset()

Q La fonction show() de Matplotlib permet l'affichage d'un graphe.

Question 5

On peut personnaliser une figure faite en Matplotlib et Seaborn.

- ☒ Vrai
- ☐ Faux

Q Il existe des fonctions ou propriétés pour personnaliser une figure en ajoutant par exemple des légendes ou en coloriant les courbes.