Python for Data Science

Introduction

David TCHOUTA

Académie Française du Numérique
www.frenchtechacademie.fr
Tél/Whatsapp : +33 (0)7 49 62 72 49

February 24, 2022

"Vous pouvez y arriver si vous y croyez." Napoléon Hill

Table de matières

Objectifs du chapitre

Dans ce chapitre introductif, nous allons tenter de répondre à la question de pourquoi l'utilisation de Python en Data Science.

À la fin de ce chapitre, vous serez en mesure :

- de savoir les raisons pour lesquelles Python est de loin le langage le plus utilisé en Data Science.

Importance de Python en Data Science

Python est l'un des langages les plus utilisés en Data Science, et ce pour plusieurs raisons :

- Python est facile à prendre en main,
- La syntaxe est facile à lire et à comprendre,
- Il existe une multitude de librairies permettant d'effectuer des calculs complexes.

Attention !

Le code en Python est généralement plus concis en comparaison d'autres langages de programmation.

Exercice d'application 1

Quel est la sortie de ce programme ?

```
y = [ ]
for i in range(3):
    x.append(i*2)
print (y[2])
```

Statistiques

La Data Science utilise de multiples techniques et méthodes pour extraire des connaissances et des idées des données. Par exemple, les **Statistiques** sont les éléments de base en Data Analysis.

En Statistiques, les concepts tels que la **moyenne**, **l'écart-type**, les **quartiles**, la **médiane**, le **mode**, les **histogrammes** ou encore les **boîtes à moustaches** sont couramment manipulés.

Definition : Moyenne, Médiane et Mode

La **moyenne** est la somme des éléments d'un tableau divisé par le nombre d'éléments. Elle est sensible à de très fortes variations dues à de grandes ou de petites valeurs.

La **médiane** est la valeur centrale permettant de diviser le tableau en deux sous-tableaux de même taille, de part et d'autre de cette valeur. **Le tableau doit être préalablement ordonné par ordre croissant**. Si la taille du tableau est un nombre pair, la médiane et tout simplement la moyenne des deux valeurs centrales.
Moyenne et médiane sont qualifiées de **mesures de la tendance centrale**.

Le **mode** est l'élément du tableau qui appraît le plus.

Exercice d'application 2

Calcul de la moyenne, de la médiane et du mode

$y = [4, 5, 6, 5]$

Quels sont le mode, la moyenne, et la médiane de la série ci-dessus ?

Definition : Écart-type

Definition

L'écart-type est la mesure de la dispersion des valeurs d'un échantillon statistique autour de sa moyenne. C'est également la racine carrée de la variance ou encore la moyenne quadratique des écarts par rapport à la moyenne.

Pour calculer l'écart-type, on va donc calculer la moyenne des écarts au carré par rapport à la moyenne.

Attention !

Un écart-type faible signifie que les valeurs sont concentrées autour de la moyenne, tandis qu'un écart-type élevé indique que les valeurs sont réparties sur une plage plus importante.

Exercice d'application 3

Calcul de l'écart-type

y = [4, 5, 6, 5, 9]

Quel est l'écart-type de la série ci-dessus ?

Quels sont les éléments qui se situent dans + ou - un écart-type ?

1 ecart-type = moyenne + écart-type ou moyenne - écart-type

Exercice d'application 4

Calcul de l'écart-type

y = [4, 4, 4, 4, 4]

Sans éffectuer de calcul, Quel serait l'écart-type de la série ci-dessus ?

Statistiques et Python

La moyenne, la médiane et l'écart-type fournissent des informations sur les données, notamment où se trouvent les **valeurs centrales** et **comment les données sont distribuées**.

Attention !

Python fournit des librairies qui permettent de calculer directement ces statistiques descriptives.

Quiz 1

1) Quelle est la moyenne de la série ?

y = [2, 3, 4]

1. 3
2. 2
3. 4

2) Quel est l'écart-type de la série ?

y = [2, 2, 2]

1. 2
2. 0
3. 3

Quiz 2

3) Quelle est la médiane de la série ?

y = [2, 3, 4, 5]

1. 3
2. 4
3. 3.5

4) Quelle est la moyenne de la série ?

Si vous gagnez 1, deux fois, 4 quatre fois.

Python for Data Science

Chapitre 1 : Les Opérations Mathématiques avec Numpy

David TCHOUTA

Académie Française du Numérique
www.frenchtechacademie.fr
Tél/Whatsapp : +33 (0)7 49 62 72 49

February 24, 2022

"Ce n'est pas le vent qui décide de votre destination, c'est l'orientation que vous donnez à votre voile. Le vent est pareil pour tous." Jim Rohn

Table de matières

Objectifs du chapitre

Dans ce chapitre, nous allons passer en revue les opérations permises par la librairie **Numpy** (Numerical Python).

À la fin de ce chapitre, vous serez en mesure :

- de manipuler les tableaux
- d'éffectuer les opérations sur les tableaux

Utilité

Numpy (Numerical Python) est une librairie de Python permettant de **travailler avec les données numériques**.

Numpy comporte des **fonctions** et des **structures de données** qui peuvent effecteur de **nombreuses opérations mathématiques**.

Pour utiliser Numpy, vous devez commencer par **l'importer** comme suit :

```
import numpy as np
```

Attention !

np est le nom le plus fréquent utilisé pour importer Numpy.

La structure Numpy Array 1

Numpy dispose d'un type de tableau appelé **Numpy array**. Contrairement aux **listes** classiques en Python, les **Numpy arrays** sont **plus rapides et plus compacts**.

Les Numpy arrays sont créés grâce à la fonction **np.array(arg)**, **arg** étant l'argument **de type list**.

La structure Numpy Array 2

Example

y = np.array([4,6,8,10,12])

y est un Numpy array contenant 5 éléments accessibles grâce à leur index, qui commence à 0.

Attention !

Les Numpy arrays sont homogènes, ce qui signifie qu'ils contiennent des éléments d'un seul et même type, contrairement aux listes, qui peuvent contenir des éléments de types différents.

Exercice d'application 1

Définir et afficher un élément d'un Numpy array

y = [4, 6, 14, 65, 409]

Définissez y comme un Numpy array et affichez le deuxième élément du tableau préalablement défini.

Manipulation des Numpy Array

Les **Numpy arrays** sont souvent qualifiés de **ndarrays**, qui signifient **N-dimensional array**, car ils peuvent avoir plusieurs dimensions.

Example

y = np.array([[4,6,8,10], [1,4,12,11], [23, 43, 5,3], [6,7,21,4]])

y est un Numpy array à 2 dimensions, avec 4 colonnes et 4 lignes.

Attention !

Les **dimensions de chaque Numpy array** du **ndarray y doivent être identiques (matrice carrée)**, sinon un message d'érreur s'affiche à l'écran.

Quelques propriétés des Numpy Array 1

les np (Numpy array) ont plusieurs propriétés (accessibles en utilisant le **dot**) telles que :

- **ndim** : renvoie le nombre de dimensions (confer le nombre de crochets ouvrants).
- **size** : renvoie le nombre total d'éléments dans le np.
- **shape** : renvoie le couple d'entiers indiquant le nombre de lignes et de colonnes.

Quelques propriétés des Numpy Array 2

Example

y = np.array([[4,6,8,10], [1,4,12,11], [23, 43, 5,3], [6,7,21,4]])
print(y.ndim) ==> 2
print(y.size) ==> 16
print(y.shape) ==> (4, 4)

Exercice d'application 2

La propriété shape

Qui dispose de la propriété shape ?

- tuple
- list
- array

Quelques propriétés des Numpy Array 3

- **np.append()** permet d'ajouter un élément au np.
- **np.delete()** permet de supprimer un élément grâce à son index.
- **np.sort()** permet de trier un np.
- **np.arange()** permet de créer un np avec éventuellement un pas. Il est similaire à la fonction range().

Quelques propriétés des Numpy Array 4

Example

y = np.array([4,6,8,10])
Ajouter 5 à y
y = np.append(y, 5)
Supprimer le deuxième élément de y
y = np.delete(y, 1)
Trier y
y = np.sort(y)
Créons z = [6, 8, 10, 12]
z = np.arange(6, 13, 2)

Exercice d'application 3

Quel est le résultat de ce programme ?

Quelle valeur affichera le programme ci-dessous ?

```
y = np.arange(2, 13, 5)

y = np.append(y, y.size)

y = np.sort(y)

print(y[1])
```

Reshape 1

Numpy utilise la fonction **reshape()** pour changer le shape, c'est-à-dire le nombre de lignes et de colonnes. Par exemple, vous pouvez changer un nd à une dimension en nd de 3 lignes, 3 colonnes.

Attention !

La taille du nd à modifier doit correspondre au produit ligne*colonne du nouveau nd, sinon un message d'erreur s'affiche à l'écran.

Reshape 2

Example

```
y = np.arrange(1,10)

z = y.reshape(3, 3)

print(z)
```

Exercice d'application 4

Reshape un nd

Complétez de manière à reshape le nd x contenant 20 éléments en un nd à 5 lignes.

x. ..... (5, ...)

Reshape 3

Avec la fonction **reshape()**, vous pouvez réaliser **l'opération inverse**, celle qui consiste **à transformer un nd de 2 dimensions en un nd à une dimension**.

Example

```
y = np.array([[4,6,8,10], [1,4,12,11], [23, 43, 5,3], [6,7,21,4]])

z = y.reshape(16)

print(z)
```

Attention !

Le même résultant peut-être obtenu en utilisant la fonction **flatten()**.

Exercice d'application 5

Reshape un nd

Quelle est la valeur retournée par ce programme ?

```
y = np.arange(1, 14, 3)

z = y.reshape(5, 1)

print(z[2][0])
```

Indexation et découpage des Numpy arrays 1

Les nd peuvent être indexés et découpés comme les listes classiques.

Example

y = np.arange(1, 13)
récupérer les 6 premiers éléments sous forme d'une liste :
print(y[:6])
récupérer les 3 premiers éléments :
print(y[0:3])
récupérer les éléments à partir de l'index 7 jusqu'à la fin :
print(y[7:])
récupérer les 4 derniers éléments de la gauche vers la droite :
print(y[-4:])
enlever les 4 derniers éléments :
print(y[:-4])

Exercice d'application 6

Récupérer le dernier élément d'un nd

Rédigez un programme permettant de récupérer et d'afficher le dernier élément du nd y. Ne pas tenir compte des semicolons.

Indexation et découpage des Numpy arrays 2

Vous pouvez récupérer uniquement les éléments qui satisfont une condition spécifique. De plus les conditions peuvent être combinées.

Example

Récupérons les éléments inférieurs à 6 :

```
y = np.arange(1, 13)

print(y[y<6])
```

Récupérons les éléments supérieurs à 6 et pairs :

```
print(y[(y>6) & (y%2==0)])
```

Exercice d'application 7

Quel est le résultat du programme ci-dessous ?

Quelle valeur renverra le programme ci-dessous ?

```
y = np.array([5, 6, 7, 8, 9])

z = x[x>7]

print(z.size)
```

Les opérations sur les tableaux 1

Il est possible d'effectuer les opérations mathématiques de base sur les nd grâce à ses fonctions natives. On peut ainsi calculer la somme, le minimum, le maximum ou encore multiplier tous les éléments d'un nd par un scalaire.

Example

```
y = np.array([5, 6, 7, 8, 9])

print(np.sum(y))
print(np.max(y))

print(np.min(y))

z = y * 2 // ça s'appelle la diffusion
print(z)
```

Exercice d'application 8

Complétez le programme ci-dessous

complétez le programme ci-dessous de manière à ce qu'il renvoie le maximum s'il est multiplié par 3 ?

y = np.array([9, 6, 8, 10, 7])

z = y 3

print(..... . .....(z))

Les opérations sur les tableaux 2

Numpy permet également de calculer la **moyenne, l'écart-type, la variance et la médiane** assez facilement.

Example

y = np.array([5, 6, 7, 8, 9])
Calcul de la moyenne :
print(**np.mean(y)**)
Calcul de la médiane :
print(**np.median(y)**)
Calcul de la variance :
print(**np.var(y)**)
Calcul de l'écart-type :
print(**np.std(y)**)

Exercice d'application 9

Complétez le programme ci-dessous

complétez le programme ci-dessous de manière à ce qu'il renvoie l'écart-type du nd y.

stdev = ... (.....)

print(stdev)

Quiz 1

1) Quelle la valeur renvoyée par ce programme ?

```
x = np.arange(3, 11)

z = x.reshape(2, 4)

print(z[1][2])
```

2) Compléter de manière à avoir uniquement les valeurs inférieurs à 10

val = ... [y < .....]

Quiz 2

3) Compléter de manière à avoir uniquement les valeurs inférieurs à la moyenne

..... [y ..... np. ..... (y)]

4) Quelle la valeur renvoyée par ce programme ?

```
x = np.arange(3, 7)

z = x * 2

print(x[:2].sum())
```

Python for Data Science

Chapitre 2 : Manipulation des données avec Pandas

David TCHOUTA

Académie Française du Numérique
www.frenchtechacademie.fr
Tél/Whatsapp : +33 (0)7 49 62 72 49

March 1, 2022

"Les personnes qui réussissent sont de grands rêveurs. Ils imaginent ce que leur avenir pourrait être et y travaillent chaque jour, motivé par leur vision afin d'atteindre leur objectif." Brian Tracy

Table de matières

Objectifs du chapitre

Dans ce chapitre, nous allons passer en revue les opérations permises par la librairie **Pandas** (vient de **panel data** : données sur plusieurs périodes de même individus).

À la fin de ce chapitre, vous serez en mesure :

- de manipuler les grands ensembles de données
- de lire les données, de les traiter et de les grouper

Utilité

Pandas est l'une des librairies de Python les plus populaires en Data Science. Facile d'utilisation, **Pandas** est **plus complet** que **Numpy** avec laquelle elle partage de nombreuses fonctions et propriétés.

Avec **Pandas**, vous pouver **lire, extraire, transformer et analyser** les données.Vous pouvez également réaliser des **calculs statistiques plus poussés**.

Pour utiliser Pandas, vous devez commencer par **l'importer** comme suit :
import pandas as pd

Attention !

pd est le nom le plus fréquent utilisé pour importer Pandas.

Series/Dataframe

Les deux **principales structures de données** de Pandas sont : les **series** et les **DataFrames**.

Les **series** sont essentiellement **une colonne de données ou un tableau à une dimension**, tandis que les **DataFrames** est un **tableau à plusieurs dimensions ou encore un ensemble de series**.

Example

taille	poids
165	73
193	89
200	90
178	84
180	85
169	78

Exercice d'application 1

Définition

Un DataFrame est fait de :

- variables
- list
- array
- series

Construction d'un DataFrame

La façon la plus simple de créer un DataFrame est d'utiliser un **dictionnaire** (Confer le cours sur les data structures).

Example

tab = {
'poids': [60, 74, 95, 60, 70],
'taille': [155, 169, 198, 182, 178]
}

Chaque clé (key) représente **le nom d'une colonne**, **le tableau** les **données de ces colonnes**.
Nous pouvons maintenant transformer ce dictionnaire en DataFrame de la manière suivante :
dtfr = pd.DataFrame(tab)

Exercice d'application 2

Nombre de colonnes

Combien de colonnes compte le DataFrame ci-dessous ?

```
tab = {
'poids': [60, 74, 95, 60, 70],
'taille': [155, 169, 198, 182, 178],
'code': [04, 69, 98, 82, 17],
'matricule': [0155, 0169, 0198, 0182, 0178]
}
dtfr = pd.DataFrame(tab)
```

- 3
- 2
- 4
- 5

Accession à un DataFrame

De façon **automatique**, le DataFrame crée un **index numérique** pour chaque ligne. Nous pouvons cependant **modifier ces index** d'une manière complétement **arbitraire** :

dtfr = pd.DataFrame(tab, index=['david', 'loic', 'henry', 'yan', 'laura'])

Grâce à la fonction **loc[ ]**, nous pouvons accéder à la ligne dont l'index a été customisé :

print(dtfr.loc["yan"])

Attention !

Remarquez que la fonction **loc** utilise les **crochets** contrairement aux fonctions standards.

Indexing/Selection

Pour selectionner une colonne, il suffit de spécifier son nom dans les crochets :
print(df["poids"])
Le résultat est une serie.

Pour selectionner plusieurs colonnes, il suffit de spécifier en argument une liste de noms de colonnes :
print(dtfr[["poids", "code"]])
Le résultat est un DataFrame.

Attention !

Remarquez l'utilisation des double **crochets** pour selectionner deux colonnes ou plus.

Exercice d'application 3

Quel est le type de z ?

y = dtfr["taille"]

z = df[["code", "taille]]

- ligne
- serie
- DataFrame
- colonne

Slicing/découpage

La fonction **iloc[ ]** permet de selectionner les données en se basant sur leur index, comme pour les **lists** en Python.

Example

première ligne :
print(dtfr.iloc[0])

les deux premières lignes :
print(dtfr.iloc[:2])

les lignes 2 to 3 :
print(dtfr.iloc[1:3])

Exercice d'application 3

Récuupérer les 5 dernières lignes

Écrire un programme permettant de récupérer les 5 dernières lignes du DataFrame dtfr. Utilisez la fonction iloc.

Slicing/découpage sur une condition

On peut également récupérer les lignes qui satisfont une certaine condition comme suit :

print(dtfr[(dtfr['poids'] > 60) & (dtfr['taille'] > 170)])

Attention !

L'opérateur logique ou (—) peut également être utilisé.

Exercice d'application 4

Récupérer les lignes spécifiques

Écrire un programme permettant de récupérer les lignes dont le poids est supérieur à 71 ou la taille est supérieure à 167.

Lecture des données : la fonction read_csv() 1

La plupart du temps, les **données** sont **stockées** au **format CSV** (comma-separated values). Pandas est capable de lire de telles données et de les transformer directement en DataFrames.

Par exemple, nous allons utiliser un fichier CSV qui contient les données d'infection à la Covid en France (accessible depuis le lien : https://www.coronavirus-statistiques.com/open-data/), appelé **open_stats_coronavirus.csv**.

Lecture des données : la fonction read_csv() 2

La fonction **read_csv()** permet de **lire les données du fichier csv et de les transformer en DataFrame**.

dtfr = pd.read_csv("https://www.coronavirus-statistiques.com/corostats/openstats/open_stats_coronavirus.csv", sep = ";")

Attention !

Vous remarquerez la présence de l'argument **sep** indiquant que **les colonnes des données sont séparées par des semicolons(;)**. En l'absence de cet argument, vous avez des semicolons qui apparaîssent dans les données, preuve d'une mauvaise lecture des données.

Lecture des données : la fonction read_csv() 3

Une autre manière de lire les données consiste à télécharger et à stocker le fichier csv sur votre disque local, puis indiquer le chemin d'accés à ce fichier dans la fonction **read_csv()**.

dtfr = pd.read_csv("...chemin d'acces.../open_stats_coronavirus.csv", sep = ";")

Attention !

Pandas permet également la **lecture** d'autres fichiers tels que les **JSON** ou même les b**ases de données SQL**.

Les fonctions head() et tail()

La fonction **head()** affiche les **premières lignes du DataFrame**, tandis que la fonction **tail()** affiche les **dernières lignes du DataFrame**.

Example

Tester les lignes de code suivantes :

```
print(dtfr.head())

print(dtfr.head(10))
```

Exercice d'application 5

Afficher les 8 dernières lignes

Écrire un programme permettant d'afficher les 8 dernières lignes de dtfr.

La fonction info()

La fonction info() permet d'avoir les informations sur le DataFrame telles que le nombre de lignes, de colonnes et les types de données.
dtfr.info()

On constate que Pandas génére automatiquement les index. On peut les modifier en définissant nos propres index grâce à la fonction **set_index()**. La date est un bon choix d'index, car elle est unique pour chaque ligne.
dtfr.set_index("date", inplace=True)

L'argument **inplace = True** signifie que les **changements se feront directement sur notre DataFrame**, pas besoin de l'affecter à un nouveau DataFrame.

La fonction drop()

La fonction drop() permet de supprimer une ligne ou une colonne.

dtfr.drop('code', axis=1, inplace=True)

axis = 1 signifie qu'on va **supprimer une colonne** (en l'occurence la colonne code).

axis = 0 signifie qu'on va **supprimer une ligne**.

Création de nouvelle colonne

Pandas permet de créer de nouvelles colonnes. Par exemple, **créons la colonne mois** en se basant sur la colonne date :

Example

```
dtfr['mois'] = pd.to_datetime(dtfr['date'],
format="%Y-%m-%d").dt.month_name()
```

Attention !

Bien respecter le format de la date. Par exemple, si **l'année** est constituée de **4 chiffres**, le **format** est : %**Y**. Si elle est composée de 2 chiffres, le format est : %**y**, sans oublier le slash, le tiret ou le point qui sépare l'année, le mois et le jour.

Exercice d'application 6

Création de deux nouvelles colonnes : jour et tauxdeces

1) Écrire un programme permettant de créer la colonne **jour** en se basant sur la colonne date de notre fichier open_stats_coronavirus.csv.

2) Écrire un programme permettant de créer la colonne **tauxdeces** en se basant sur la colonne **deces** et la colonne **cas** de notre fichier open_stats_coronavirus.csv.
Sachant le taux de décès est le rapport entre le nombre de décès sur le nombre de cas déclarés.

Statistiques descriptives

Grâce à la fonction **describe()**, nous pouvons avoir directement les**principales statistiques des colonnes numériques** de notre fichier.Cette fonction renvoie **la moyenne, l'écart-type, le minimum, le maximum**, etc.

dtfr.describe()

Attention !

On peut également avoir le statistiques descriptives d'une colonne spécifique en la spécifiant :

dtfr['cas'].describe()

Exercice d'application 7

Quel est le résultat du programme suivant ?

```
tab = {
'poids': [60, 74, 95, 60, 70],
'taille': [155, 169, 198, 182, 178],
'code': [14, 69, 98, 82, 17],
'matricule': [10155, 20169, 30198, 40182, 50178]
}

dtfr = pd.DataFrame(tab)
print(dtfr['poids'].mean())
```

- 70.8
- 73
- 71.8
- 72

La fonction value_counts()

Pour compter le nombre de saisies mensuelles, nous pouvons utiliser la fonction value_counts(). Autrement, cette fonction compte le nombre de saisies par mois (fréquences absolues), à partir desquelles, nous pouvons déduire les fréquences relatives.

print(dtfr['mois'].value_counts())

La fonction groupby()

La fonction **groupby()** permet de **regrouper les données** (par exemple mois) et de calculer par la suite le nombre de cas d'infections mensuelles (en faisant la somme des saisies journalières d'un mois).

print(dtfr.groupby('mois')['cas'].sum())

Le nombre de cas annuels est donné par :
print(dtfr['cas'].sum())

Attention !

De même, nous pouvons calculer les statistiques : **mean(), min(), max()** correspondant à **chaque groupe** (grâce au **groupby()**).

Exercice d'application 7

Écrire un programme

1) Écrire un programme permettant de compter le nombre de saisies journalières en France depuis le début de l'épidémie.

2) Calculer la moyenne du nombre décès en France par mois.

NB: Les données sont cumulées et correspondent aux données récoltées auprès de 178 pays dans le monde.

Quiz 1

1) Écrire un programme

Écrire un programme permettant de récupérer les données de la France uniquement.

2) Affichage des tranches de données

Écrire un programme permettant d'afficher les 20 premières valeurs.

Quiz 2

3) Écrire un programme

Écrire un programme permettant de récupérer les données de la France uniquement.

4) Affichage des tranches de données

Écrire un programme permettant d'afficher les 20 premières valeurs des données de l'exercice 1.

Quiz 3

5) Quel est le résultat du programme suivant ?

```
tab = {
'poids': [60, 74, 95, 60, 70],
'taille': [155, 169, 198, 182, 178]
}

dtfr = pd.DataFrame(tab)

dtfr['poidtaille'] = dtfr['poids'] + dtfr['taille']

print(dtfr.iloc[3]['poidtaille'])
```

- 293
- 248
- 242

Quiz 4

6) Suppression d'une colonne

Supprimez la colonne tauxdeces créée à l'exercice d'application 6, plus haut.

Python for Data Science

Chapitre 3 : Visualisation des données avec Matplotlib

David TCHOUTA

Académie Française du Numérique
www.frenchtechacademie.fr
Tél/Whatsapp : +33 (0)7 49 62 72 49

March 3, 2022

"Le chemin vers la réussite passe par l'apprentissage continu de nouvelles connaissances." Napoleon Hill

Table de matières

Objectifs du chapitre

Dans ce chapitre, nous allons passer en revue la création des graphiques, des diagrammes et des figures permises par la librairie **Matplotlib**.

À la fin de ce chapitre, vous serez en mesure :

- de construire des graphiques, des diagrammes, des figures,
- et de les customiser.

Importation de la librairie Matplotlib

Pour utiliser Matplotlib, vous devez commencer par **l'importer** comme suit :

import matplotlib.pyplot as plt

Attention !

plt est le nom le plus fréquent utilisé pour importer Matplotlib.

Matplotlib et Pandas 1

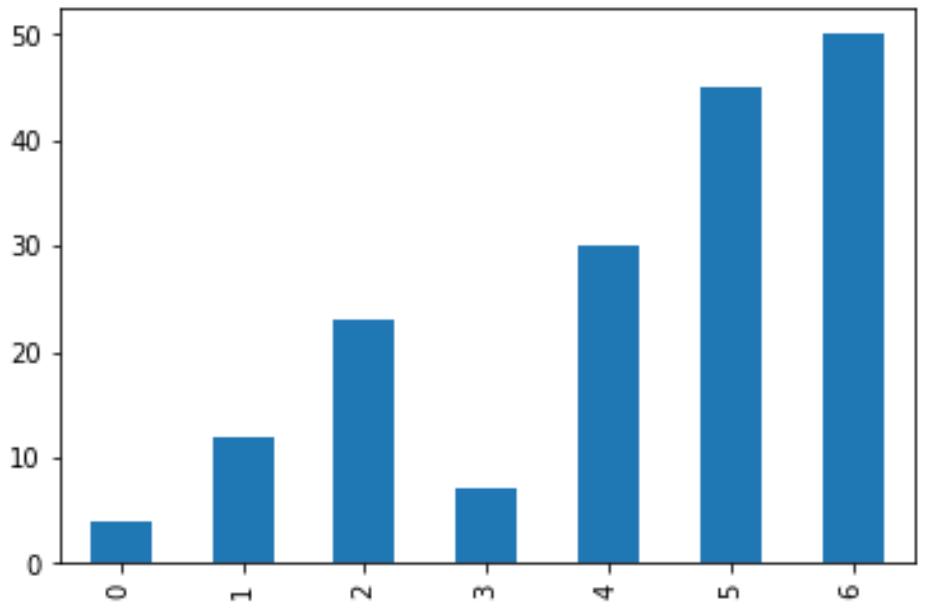
Matplotlib fonctionne très bien avec Pandas.

Créons notre premier graphique avec des données quelconques d'une série.

```
tab = pd.Series([4, 12, 23, 7, 30, 45, 50])
```

```
tab.plot(kind='bar')
```

Bar chart

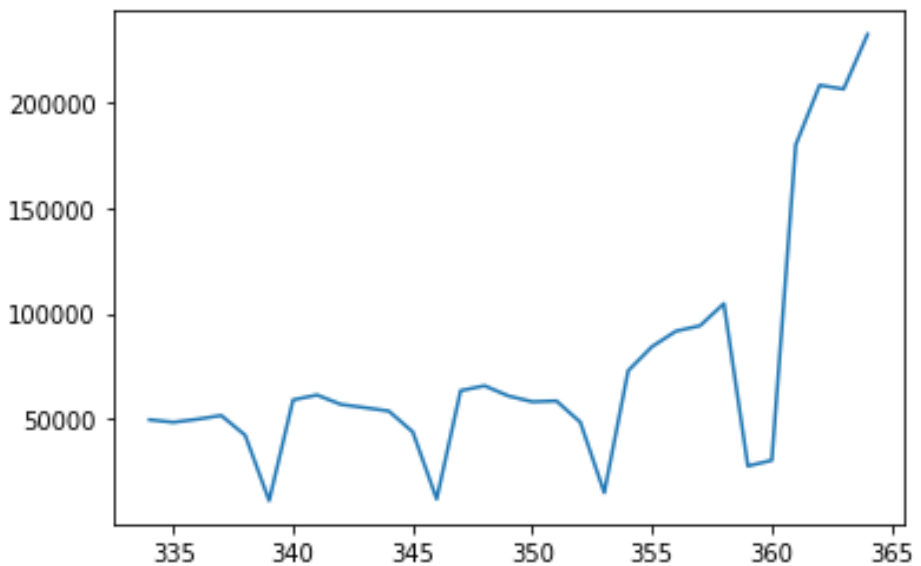


Les valeurs de la série sont, **par défaut**, sur **l'axe des ordonnées**, tandis que **les index** (**par défaut**) sont **en abscisses**. Pour l'instant, nous n'avons éffectué aucune customisation de l'histogramme.

Line chart 1

Matplotlib permet de construire de nombreux graphiques tels que les graphiques linéaires (**line chart**). Pour le faire, nous utilisons la fonction **plot()** sur notre DataFrame.

dtfr[dtfr['month']=='December']['casret'].plot()



Exercice d'application 1

Écrire un programme

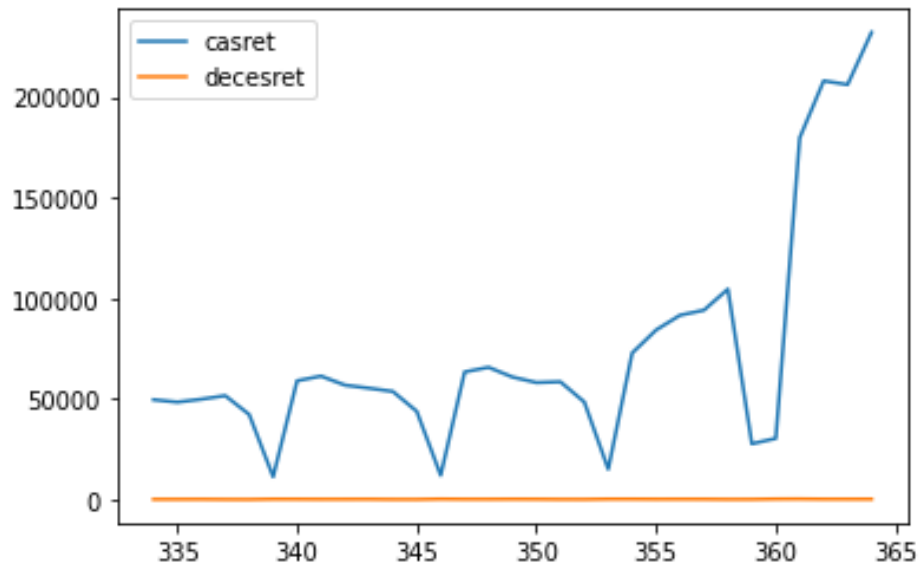
Écrire un programme permettant de tracer un graphique linéaire du nombre de décès.

NB: Les données sont cumulées et correspondent aux données récoltées auprès de 178 pays dans le monde.

Line chart 2

Vous pouvez tracer d'autres lines sur le même graphique. Ajoutons la line représentant le nombre de décès journaliers en France en 2021.

(dtfr[dtfr['month']=='December'])[['casret', 'decesret']].plot()



Exercice d'application 2

Quel est le nombre de line charts du programme suivant ?

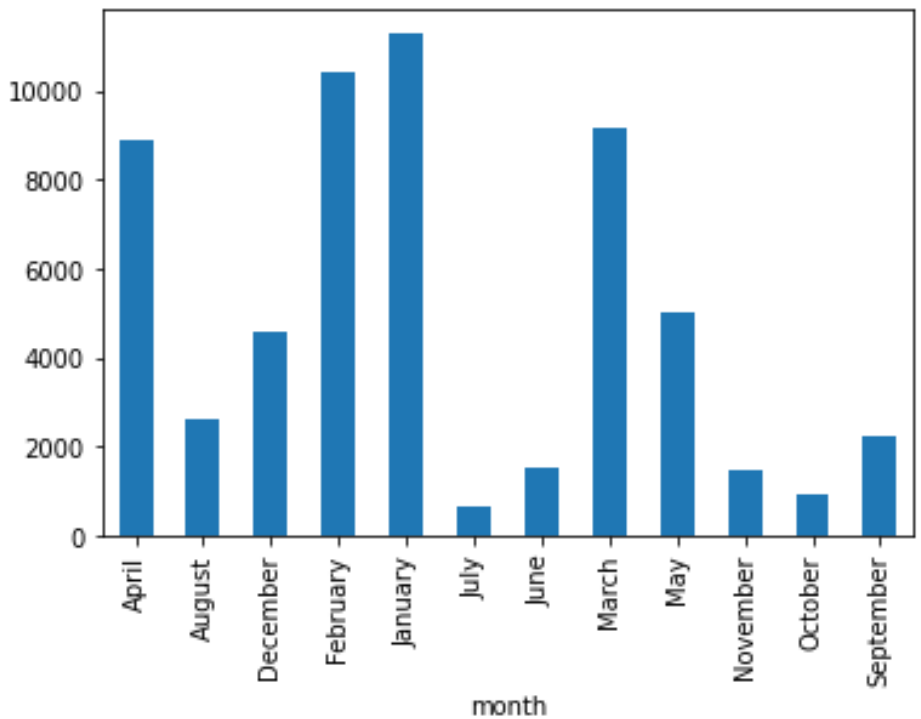
tab[['poids', 'taille', 'code']].plot()

- 2
- 3
- 1
- 0

Bar chart 1

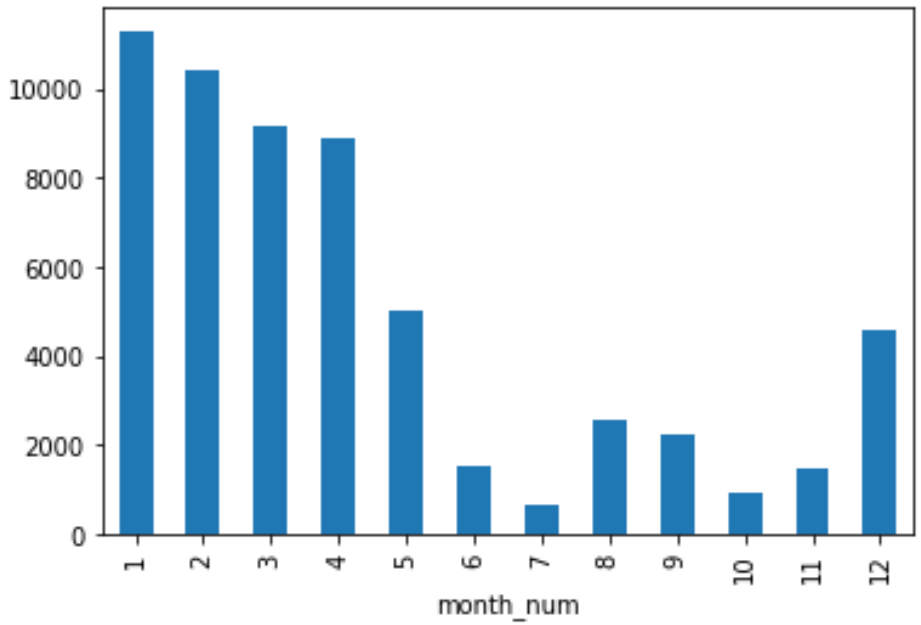
La fonction **plot()** peut prendre **l'argument kind**. Pour les **diagrammes en barres**, **kind="bar"**.

(datafr.groupby('month')['decesret'].sum()).plot(kind="bar")



Bar chart 2

(datafr.groupby('month_num')['decesret'].sum()).plot(kind="bar")



Attention !

Nous regroupons les données de décès par mois et par la suite, nous calculons la somme des décès mensuels.

Exercice d'application 3

Écrire un programme

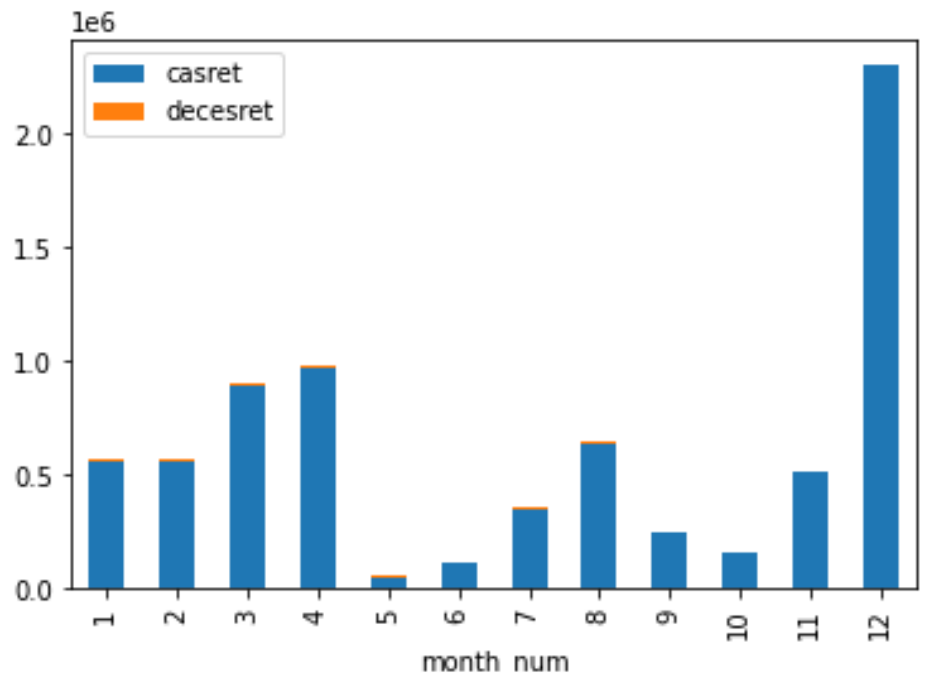
Écrire un programme permettant de tracer un graphique en barre du nombre de cas.

NB: Les données sont cumulées et correspondent aux données récoltées auprès de 178 pays dans le monde.

Stacked bar chart 1

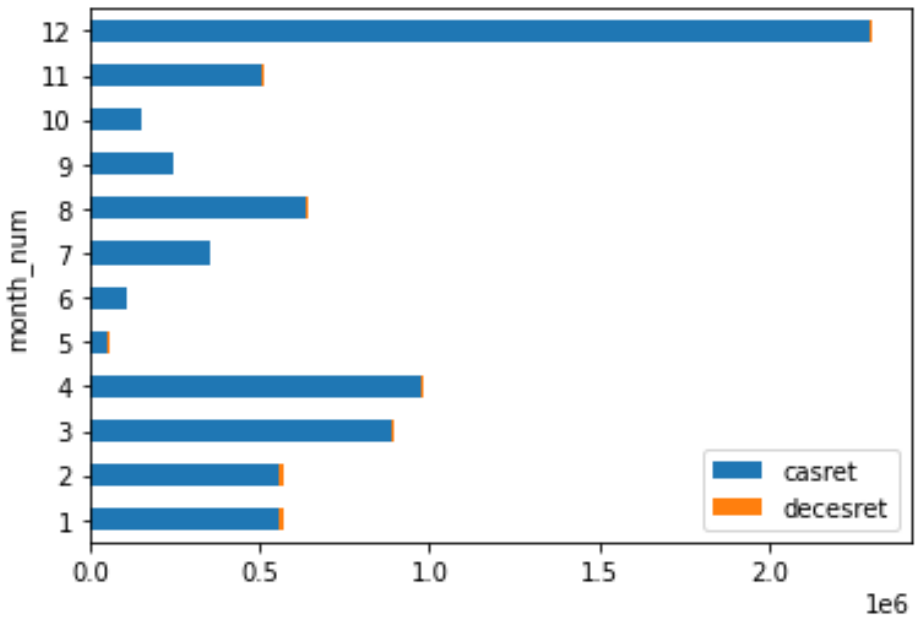
L'argument stacked=True permet de spécifier si le **graphique en barre** doit être **empilé ou pas**.

datafr.groupby('month_num')[['casret', 'decesret']].sum().plot(kind="bar", stacked=True)



Stacked bar chart 2

datafr.groupby('month_num')[['casret', 'decesret']].sum().plot(kind="barh", stacked=True)



Attention !

L'argument **kind = barh** peut être utilisé pour créer des **barres horizontales**.

Exercice d'application 4

Écrire un programme

Écrire un programme permettant de tracer un graphique en barre horizontale empilé du nombre de cas moyens par mois.

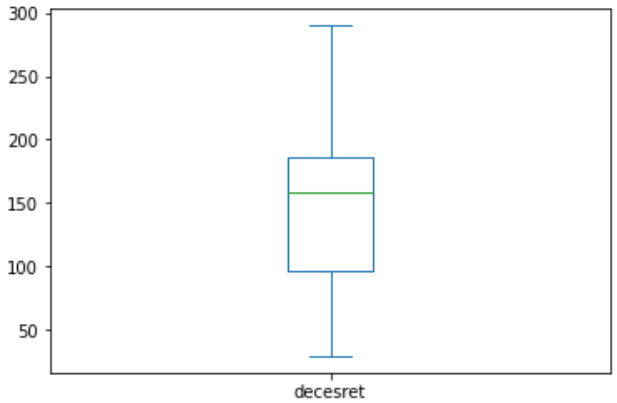
NB: Les données sont cumulées et correspondent aux données récoltées auprès de 178 pays dans le monde.

Box plot ou boîte à moustaches 1

Les **box plot** sont généralement utilisées pour **visualiser la distribution des données d'une variable**. Ce **type de graphique correspond** à la fonction **describe()**.

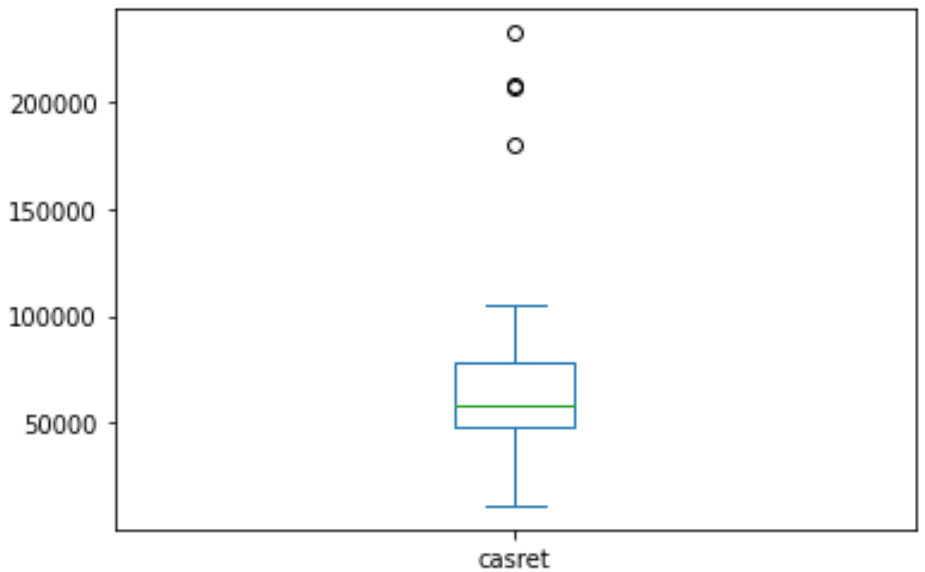
datafr[datafr['month']=='December']['decesret'].plot(kind="box")

Box plot ou boîte à moustaches 2



La **line verte** correspond à la **médiane**, **la première ligne bleue horizontale de la boîte** correpond au **1er quartile** (25% des données), La **seconde ligne bleue horizontale de la boîte** (située au-dessus de la ligne verte) correspond au **3éme quartile** (75% des données). Les **les petits segments horizontaux bleues situés aux extrémités** correspondent au **minimum et au maximum** de la variable.

Box plot ou boîte à moustaches 3

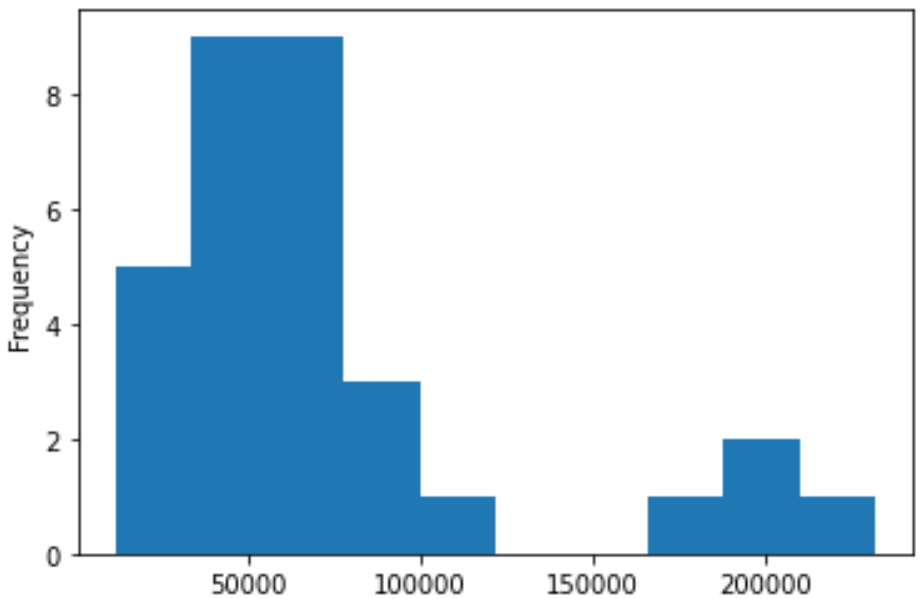


Les **cercles** correspondent aux **points abérrants** (point qui diffère significativement des autres points).

Histogramme 1

L'**histogramme** représente les **fréquences de groupe de données**, contrairement aux **bar charts** qui sont les p**oints individuels de la donnée**. Autre spécificité, l'histogramme n'a pas d'espace entre les barres.

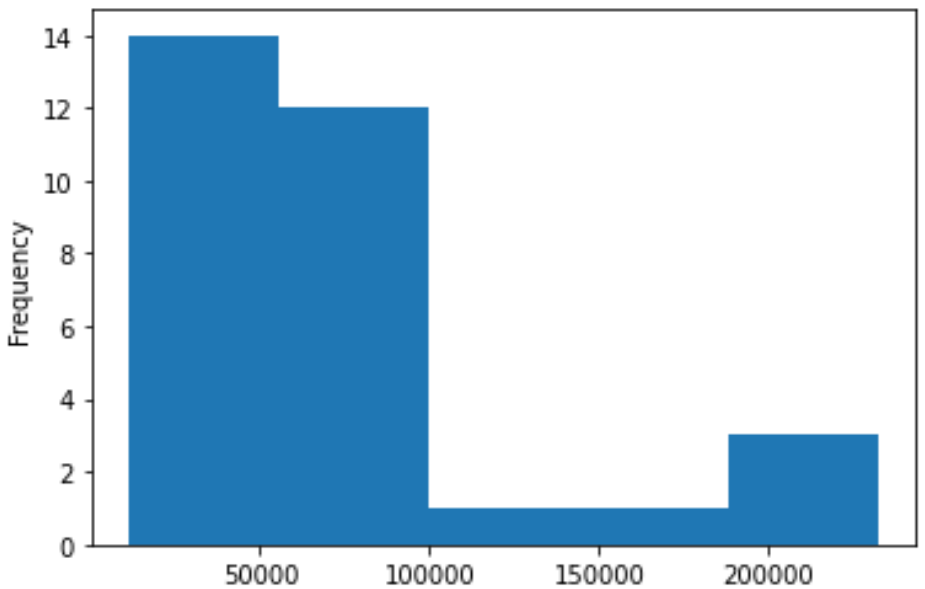
datafr[datafr['month']=='December']['casret'].plot(kind="hist")



Histogramme 2

Vous pouvez spécifier le nombre de bins comme suit :

datafr[datafr['month']=='December']['casret'].plot(kind="hist", bins = 5)



Exercice d'application 5

Écrire un programme

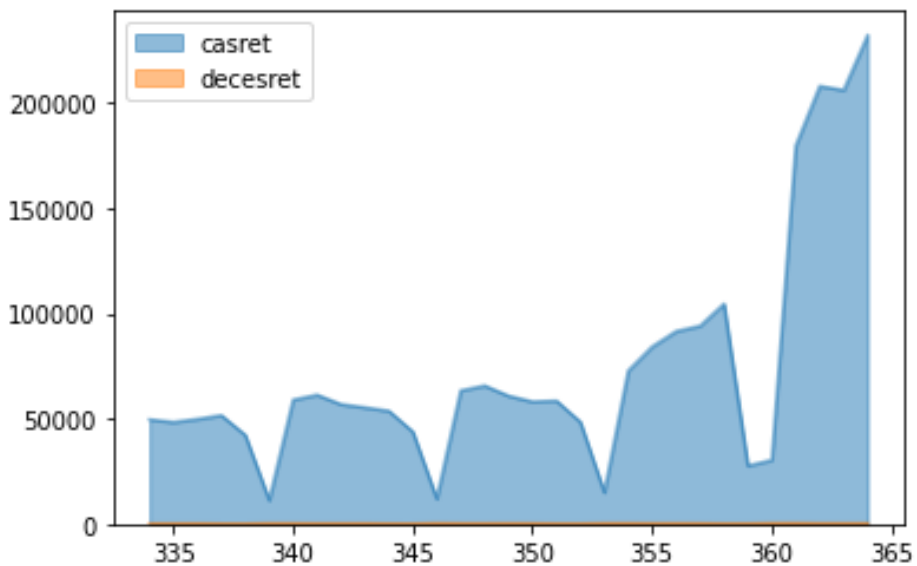
Écrire un programme permettant de tracer un histogramme avec un nombre de bins à 5 du nombre de cas au mois de décembre.

NB: Les données sont cumulées et correspondent aux données récoltées auprès de 178 pays dans le monde.

Area plot 1

Pour créer un graphique de zone/surface appelé **area plot**, vous pouvez **changer le kind** comme suit :

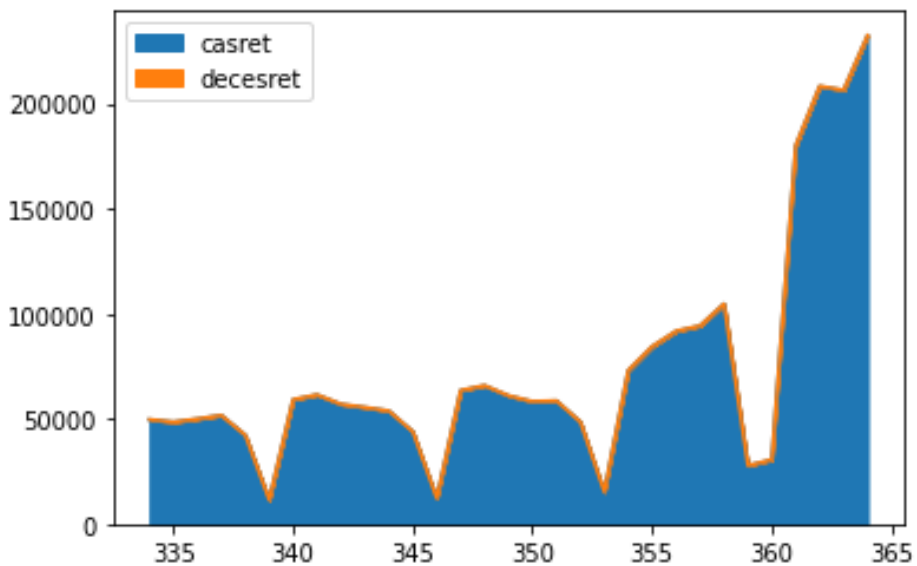
datafr[datafr['month']=='December']['casret'].plot(kind="area", stacked = False)



Area plot 2

Par défaut, stacked = False. Si vous le changez à **True**, vous obtenez le graphique suivant :

datafr[datafr['month']=='December']['casret'].plot(kind="area", stacked = True)



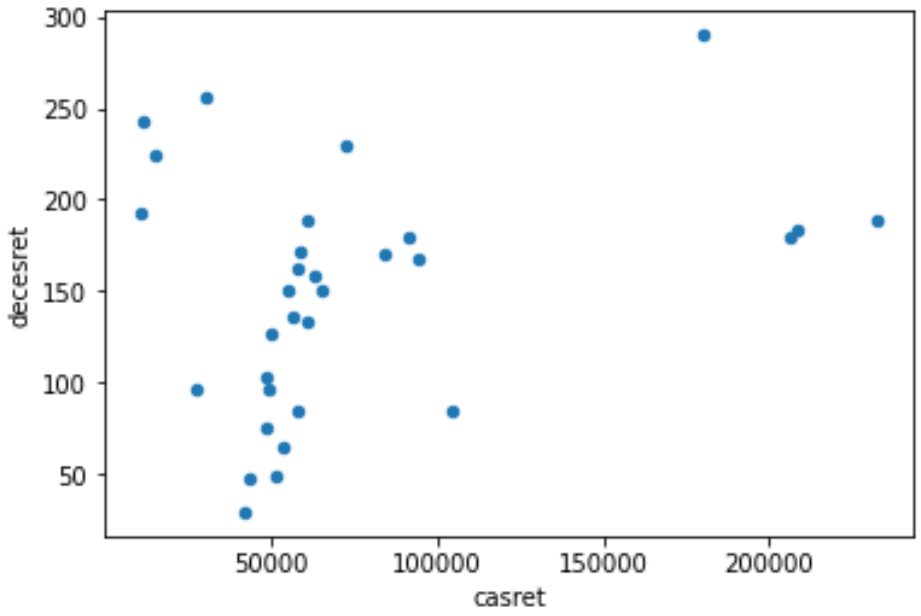
Scatter plot 1

Les **scatters plots ou cross-plots** permettent de **visualiser la relation entre deux variables**. Par exemple, le nombre de cas retraité (casret) et le nombre de décès retraité.

Nous devons spécifier les colonnes en x et en y qui doivent être utilisées pour le graphique.

Scatter plot 2

datafr[datafr['month']=='December']['casret'].plot(kind="scatter" x=casret, y=decesret)



Les données semblent dispersées, d'où le nom de ce type de graphique : **scattered qui signifie dispersé**.

Exercice d'application 6

Combien de points ?

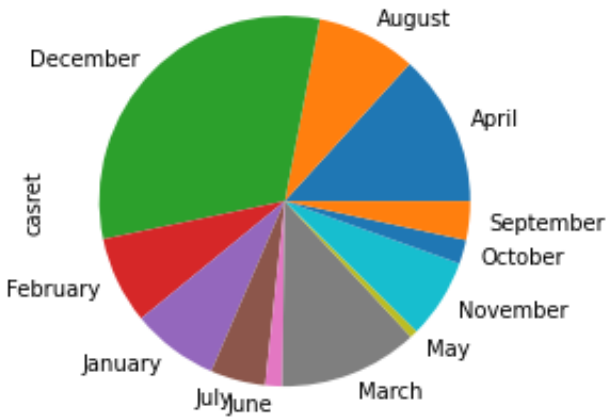
Nous avons un DataFrame de deux colonnes, le poids en 2015 et le poids en 2022, chacun contenant 10 valeurs.

Combien de points le scatter plot de ces deux variables seront affichés ?

- 100
- 20
- 10

Pie chart

Vous pouvez créer des **pie charts** (diagramme circulaire) en **changeant le kind** comme suit :
(datafr.groupby('month')['casret'].sum()).plot(kind="pie")



Les diagrammes circulaires sont généralement utilisés pour illustrer les pourcentages, notamment lorsqu'on a jusqu'à 6 catégories.

Exercice d'application 6

Écrire un programme

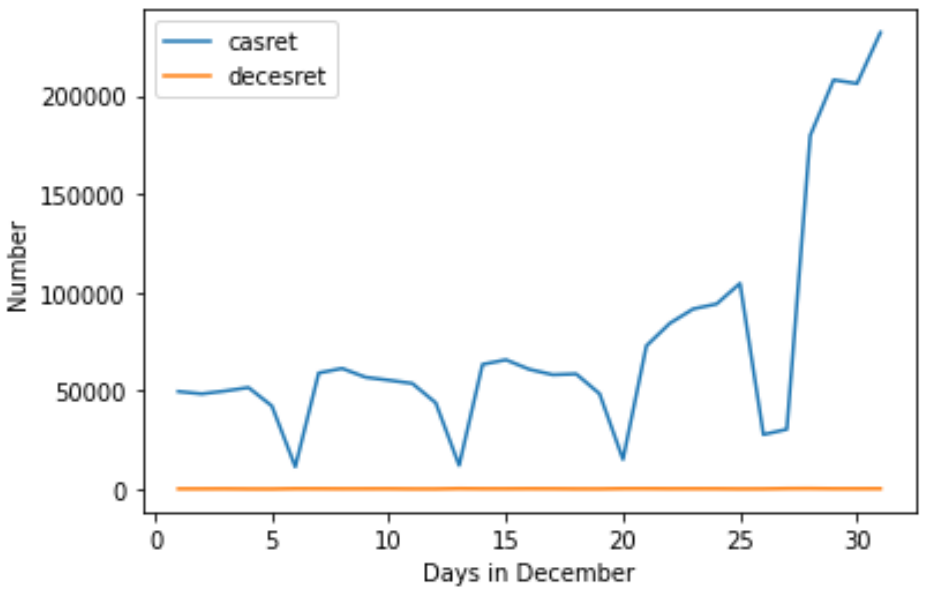
Écrire un programme permettant de tracer un pie chart du nombre de cas par mois.

NB: Les données sont cumulées et correspondent aux données récoltées auprès de 178 pays dans le monde.

Formatage de la légende 1

Vous pouvez spécifier une **légende** à l'aide des arguments **xlabel** et **ylabel** et l'afficher ou pas, comme suit :

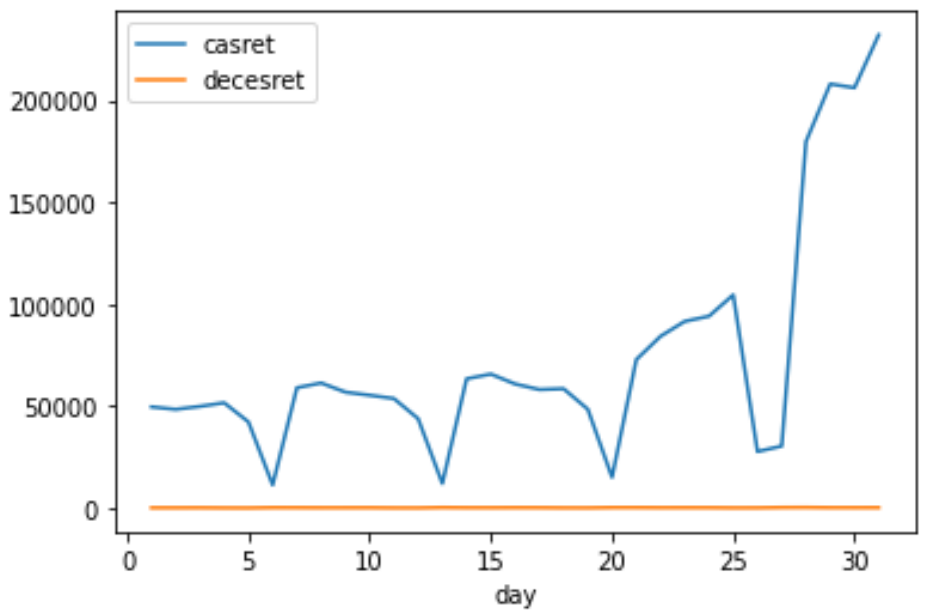
```
(datafr[datafr['month']=='December'])[['casret',
'decesret']].plot(kind='line', legend =True)
plt.xlabel('Days in December')
plt.ylabel('Number')
```



Formatage de la légende 2

Per défaut, Pandas sélectionne **le nom de l'index** comme **xlabel**, tandis le **ylabel** est laissé à **vide**.

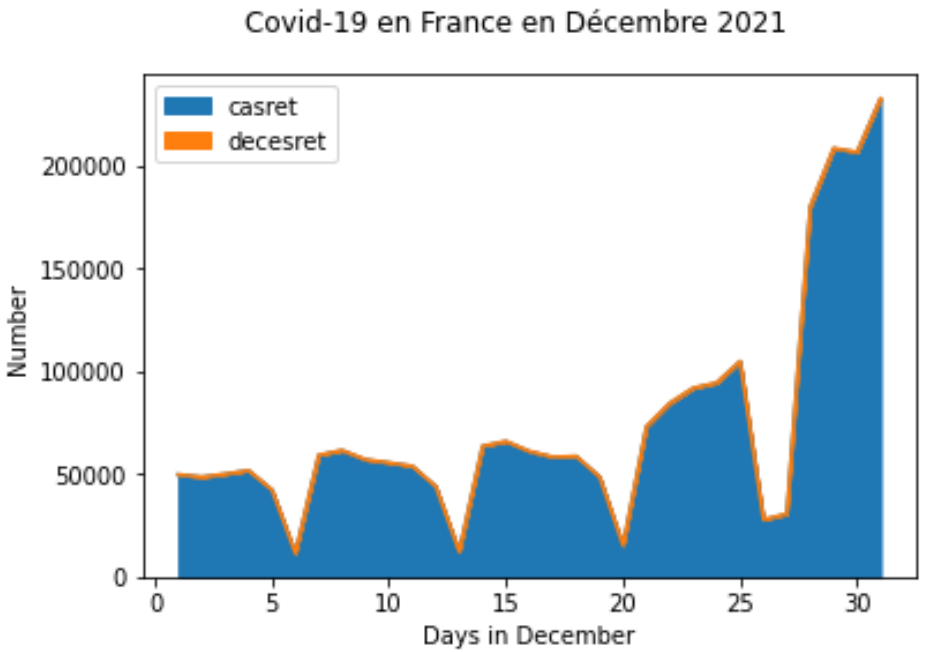
(datafr[datafr['month']=='December'])[['casret', 'decesret']].plot(kind='line', legend =True)



Formatage du titre

La ligne suivante permet de donner un titre au graphe :

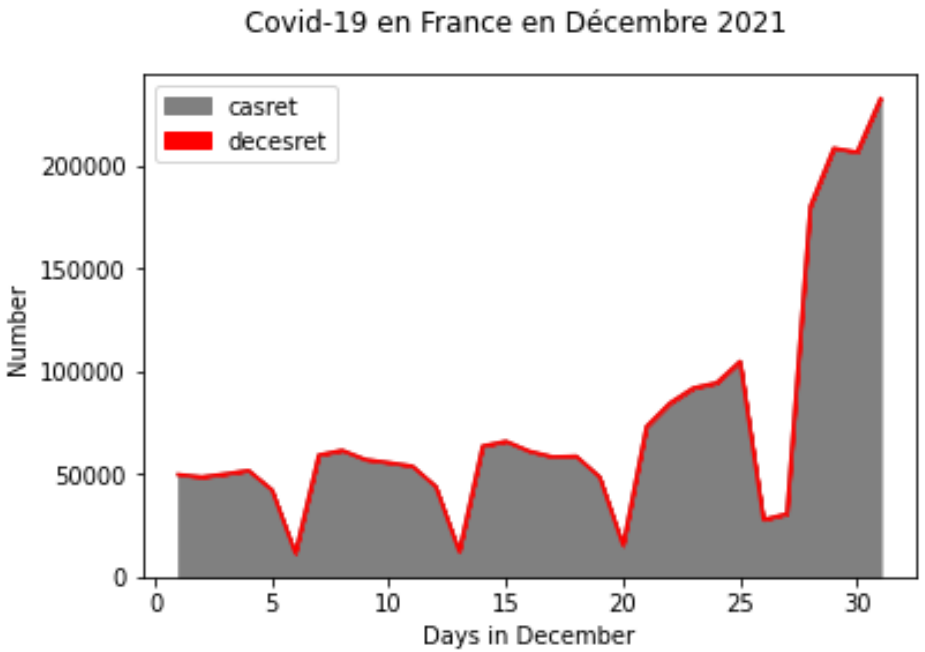
plt.suptitle("Covid-19 en France en Décembre 2021")



Formatage de la couleur

Nous pouvons aussi **changer la couleur** grâce à l'attribut **color**.

```
(datafr[datafr['month']=='December'])[['casret',
'decesret']].plot(kind='area', legend =True, color=['grey', 'red'])
```



Formatage

Attention !

Tous ces attributs (color, xlabel, ylabel, legend, title, etc.) sont valables pour la plupart des graphiques.

Exercice d'application 7

Construire un graphique customisé

Reprendre le graphique ci-dessus, cette fois-ci avec une kind ='bar' et les couleurs verte et bleue.

NB: Les données sont cumulées et correspondent aux données récoltées auprès de 178 pays dans le monde.

Quiz 1

1) Qu'affiche un histogramme ?

- les écart-types
- les moyennes
- la distribution des fréquences

2) Écrire un programmme

Écrire un programme permettant d'avoir le nom de l'axe x :'jour' et l'axe y: 'cas'. Reprendre le graphe de la slide 'Formatage de la légende 1'.

Quiz 2

3) construire un pie chart

Soit le DataFrame suivant:
tab = {
'categorie': ['leger', 'semi-leger', 'moyen', 'semi-moyen', 'lourd'],
'taille': [155, 169, 198, 182, 178]
}
Construire le pie chart correspondant à ce DataFrame. N'oubliez pas de transformer tab en DataFrame.

Quiz 3

4) custumisation un pie chart

Ajoutez un titre, ainsi que les noms des axes du graphique de la question 3.

5) Outliers

Á quoi correspond les outliers dans un box plot ?

Python for Data Science

Chapitre 4 : Examen de certification

David TCHOUTA

Académie Française du Numérique
www.frenchtechacademie.fr
Tél/Whatsapp : +33 (0)7 49 62 72 49

March 6, 2022

"Vous devez y mettre beaucoup, beaucoup, beaucoup d'infimes efforts que personne ne perçoit avant d'aboutir à quelque chose qui en vaux la peine." Brian Tracy

Table de matières

Projet 1

Soit la liste suivante :

distance = [145, 168, 180, 185, 196, 175]

1) Calculer et afficher la moyenne

2) Calculer et afficher l'écart-type

3) Afficher les valeurs qui se trouvent à + ou - un écart-type

Projet 2

Soit le nd (Numpy array) suivant :

tab = np.array([3200, 4500, 6700, 9800, 8700, 2100, 6100,4400])

1) Calculer et afficher la moyenne

2) Calculer et afficher l'écart-type

3) Afficher le pourcentage des valeurs qui se trouvent à + ou - un écart-type

Attention !

Le pourcentage est obtenu en divisant le nombre de valeurs satisfaisant la condition sur la taille du nd, le tout multiplié par 100.

Projet 3

En utilisant les données de Santé Publique France : *https://www.coronavirus-statistiques.com/corostats/openstats/open_stats_coronavirus.csv*

1) Créer les colonnes month, tauxdeces (nombre de décès divisé par le nombre de cas) et annee

2) Supprimez toutes les saisies correspondates aux années 2021 et 2022 (**autrement, gardez uniquement les saisies de 2020**)

3) Conservez uniquement les saisies de 2020 **en France uniquement**.

Projet 4

En vous appuyant sur l'exercice 3 :

1) calculez dans une nouvelle colonne le nombre de cas journalier (non cumulé) en France en 2020. Faîtes de même pour le nombre décès.

2) Calculez et affichez la ligne correspondante au jour ayant enregistré le taux de décès le plus élévé.