





Expert  
EXPERT

# Python

## pour la Data Science

Analysez vos données par la pratique  
avec **NumPy**, **Pandas**, **Matplotlib** et **Seaborn**

### En téléchargement

-  code source
-  jeux de données

 + QUIZ

Version en ligne  
**OFFERTE !**  
pendant 1 an

Amandine VELT

Les éléments à télécharger sont disponibles à l'adresse suivante :  
**<http://www.editions-eni.fr>**  
Saisissez la référence ENI de l'ouvrage **EIPYTDAT** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.



## Chapitre 1

### Avant-propos et introduction

- 1. Avant-propos ..... 11
- 2. Python et l'analyse de données ..... 12
  - 2.1 L'explosion des données ..... 12
  - 2.2 L'analyse de données ..... 13
  - 2.3 R et Python pour l'analyse de données ..... 14
- 3. Connaître les sources de données libres ..... 15
  - 3.1 Kaggle ..... 15
  - 3.2 Les données gouvernementales ..... 17
- 4. Déroulement du livre ..... 18

## Chapitre 2

### Mise en place de l'environnement de travail

- 1. Introduction : pourquoi utiliser Python pour la Data Science ? .... 19
- 2. Introduction à IPython et Jupyter ..... 20
  - 2.1 Introduction à IPython ..... 20
  - 2.2 Le projet Jupyter ..... 22
- 3. Qu'est-ce qu'Anaconda ? ..... 23
- 4. Installation d'Anaconda ..... 24
  - 4.1 Installation sur Windows ..... 24
  - 4.2 Installation sur MacOS ..... 26
  - 4.3 Installation sur Linux ..... 26

# 2 ————— Python pour la Data Science

Analysez vos données par la pratique

|       |   |    |
|-------|---|----|
| 5.    | Découverte d'Anaconda Navigator . . . . .                               | 28 |
| 5.1   | Applications disponibles dans Anaconda Navigator . . . . .              | 28 |
| 5.2   | Gestion des packages et environnements . . . . .                        | 29 |
| 6.    | Prise en main de Jupyter Notebook. . . . .                              | 32 |
| 6.1   | Tableau de bord de Jupyter Notebook . . . . .                           | 32 |
| 6.2   | Premiers pas avec les notebooks . . . . .                               | 34 |
| 6.3   | Comprendre l'interface des notebooks . . . . .                          | 38 |
| 6.3.1 | La barre de menus . . . . .   | 38 |
| 6.3.2 | La barre d'outils . . . . .   | 40 |
| 6.3.3 | Les cellules . . . . .  | 41 |
| 6.3.4 | Les modes Commande et Edition<br>et les raccourcis-clavier . . . . .    | 42 |
| 6.3.5 | Les bases du langage Markdown<br>pour écrire dans un notebook . . . . . | 45 |
| 6.3.6 | Partager son notebook. . . . .  | 51 |
| 7.    | Les packages Python essentiels pour la Data Science. . . . .            | 52 |
| 7.1   | NumPy . . . . .   | 52 |
| 7.2   | Pandas . . . . .  | 53 |
| 7.3   | Matplotlib. . . . .   | 53 |
| 7.4   | Seaborn . . . . .   | 53 |

## Chapitre 3

### Rappels sur le langage Python

|     |  |    |
|-----|--|----|
| 1.  | Introduction sur le langage de programmation Python. . . . .     | 55 |
| 2.  | Les variables . . . . .  | 56 |
| 3.  | Les différents types de données (int, float, bool, str). . . . . | 58 |
| 3.1 | Les nombres réels et entiers . . . . .                           | 58 |
| 3.2 | Les booléens . . . . .   | 59 |
| 3.3 | Les chaînes de caractères . . . . .                              | 60 |

|  |    |
|--|----|
| 4. Les structures de données basiques (listes, tuples et dictionnaires) .                                | 62 |
| 4.1 Les listes . . . . .   | 62 |
| 4.1.1 Créer une liste . . . . .  | 62 |
| 4.1.2 Accéder aux éléments d'une liste . . . . .   | 63 |
| 4.1.3 Ajouter et supprimer des éléments à une liste. . . . .   | 66 |
| 4.2 Les tuples. . . . .  | 70 |
| 4.3 Les dictionnaires . . . . .  | 72 |
| 4.3.1 Introduction aux dictionnaires. . . . .  | 72 |
| 4.3.2 Ajouter, modifier et supprimer<br>des éléments d'un dictionnaire . . . . .                         | 74 |
| 4.3.3 Parcourir un dictionnaire . . . . .  | 76 |
| 5. Les opérateurs arithmétiques, relationnels et logiques . . . . .                                      | 78 |
| 5.1 Les opérateurs arithmétiques. . . . .  | 78 |
| 5.2 Les opérateurs relationnels et logiques . . . . .  | 80 |
| 5.2.1 Les opérateurs relationnels . . . . .  | 80 |
| 5.2.2 Les opérateurs logiques . . . . .  | 81 |
| 6. Vocabulaire en Python : fonctions, méthodes,<br>attributs, modules et librairies (packages) . . . . . | 82 |
| 6.1 Fonctions. . . . .   | 82 |
| 6.2 Méthodes. . . . .  | 84 |
| 6.3 Attributs . . . . .  | 85 |
| 6.4 Modules. . . . .   | 85 |
| 6.5 Librairies (packages) . . . . .  | 87 |
| 7. Instructions de condition if et boucles for . . . . .   | 88 |
| 7.1 Instruction de condition if. . . . .   | 88 |
| 7.2 Boucle for . . . . .   | 90 |

# 4 Python pour la Data Science

Analysez vos données par la pratique

## Chapitre 4 Maîtriser la librairie NumPy

|  |     |
|--|-----|
| 1. Introduction à NumPy . . . . .                                | 93  |
| 2. Les tableaux NumPy . . . . .                                  | 94  |
| 2.1 Créer un ndarray . . . . .                                   | 94  |
| 2.1.1 Créer un ndarray à partir de listes . . . . .              | 94  |
| 2.1.2 Créer un ndarray grâce à des fonctions NumPy . . . . .     | 97  |
| 2.1.3 Créer un ndarray à partir d'un fichier . . . . .           | 100 |
| 2.2 Indexation . . . . .   | 103 |
| 2.2.1 Indexation simple . . . . .                                | 103 |
| 2.2.2 Indexation booléenne . . . . .                             | 105 |
| 2.2.3 Fancy indexing . . . . .                                   | 108 |
| 2.3 Accéder aux éléments par tranche (slicing) . . . . .         | 110 |
| 2.3.1 Slicing sur un tableau NumPy à 1 dimension . . . . .       | 110 |
| 2.3.2 Slicing sur un tableau NumPy à 2 dimensions . . . . .      | 112 |
| 2.4 Notion de vue et copie . . . . .                             | 114 |
| 3. Les opérations mathématiques avec NumPy . . . . .             | 116 |
| 3.1 Les opérations arithmétiques . . . . .                       | 116 |
| 3.2 Les fonctions d'agrégations . . . . .                        | 119 |
| 4. Inspecter un tableau grâce aux attributs de NumPy . . . . .   | 122 |
| 5. Manipuler des tableaux NumPy . . . . .                        | 124 |
| 5.1 Ajouter et supprimer des éléments dans un tableau . . . . .  | 124 |
| 5.1.1 Ajouter des éléments dans un tableau . . . . .             | 124 |
| 5.1.2 Supprimer des éléments d'un tableau . . . . .              | 127 |
| 5.2 Diviser un tableau NumPy (split, hsplit et vsplit) . . . . . | 128 |
| 5.2.1 Sur un tableau à une dimension . . . . .                   | 129 |
| 5.2.2 Sur un tableau à deux dimensions . . . . .                 | 130 |
| 5.3 Concaténer/combiner des tableaux . . . . .                   | 131 |
| 5.3.1 La fonction concatenate() . . . . .                        | 131 |
| 5.3.2 Les fonctions vstack() et hstack() . . . . .               | 133 |
| 6. Introduction aux matrices avec NumPy . . . . .                | 135 |

**Chapitre 5**  
**Maîtriser la librairie Pandas**

- 1. Introduction ..... 137
  - 1.1 Introduction à la librairie Pandas. .... 137
  - 1.2 Introduction au jeu de données utilisé pour les exemples . . . 140
- 2. Lire et écrire des fichiers avec Pandas ..... 143
  - 2.1 Lecture de fichiers texte (CSV ou TXT) ..... 143
    - 2.1.1 Lecture basique d'un fichier ..... 143
    - 2.1.2 Gestion de l'en-tête ..... 146
    - 2.1.3 Gestion des index. .... 148
    - 2.1.4 Création d'un tableau à une dimension  
à partir du fichier. .... 150
    - 2.1.5 Filtrage des colonnes lors de la lecture du fichier . . . . 150
    - 2.1.6 Les types des différentes colonnes ..... 152
    - 2.1.7 Gestion des dates lors de la lecture du fichier . . . . . 152
  - 2.2 Lecture de fichiers Excel. .... 153
  - 2.3 Importation des données à partir d'une base de données. . . . 156
  - 2.4 Lecture de fichiers au format JSON. .... 158
  - 2.5 Écriture de fichiers ou exportation de données. .... 160
- 3. Structure de données Pandas : les Series (Séries) ..... 162
  - 3.1 Introduction ..... 162
  - 3.2 Créer des séries ..... 163
    - 3.2.1 À partir de valeurs aléatoires. .... 163
    - 3.2.2 À partir d'une liste Python ..... 165
    - 3.2.3 À partir d'un tableau NumPy (ndarray)..... 168
    - 3.2.4 À partir d'un fichier texte ..... 168
  - 3.3 Choisir l'index d'une série. .... 169
  - 3.4 Accéder aux valeurs d'une série ..... 171
    - 3.4.1 Indexing via la position des valeurs ..... 171
    - 3.4.2 Indexing via l'étiquette des valeurs ..... 172
    - 3.4.3 Les indexeurs loc et iloc. .... 173
    - 3.4.4 Indexing via une expression booléenne ..... 175

# 6 — Python pour la Data Science

Analysez vos données par la pratique

|       |  |     |
|-------|--|-----|
| 3.4.5 | Slicing : découpage de valeurs successives . . . . .   | 178 |
| 3.5   | Les attributs et les méthodes des objets de classe Series . . . . .  | 184 |
| 3.5.1 | Les attributs des objets de classe Series . . . . .  | 184 |
| 3.5.2 | Les méthodes des objets de classe Series . . . . .   | 185 |
| 3.6   | Ajouter, supprimer et modifier les valeurs d'une série . . . . .   | 187 |
| 3.6.1 | Ajouter des valeurs à une série . . . . .  | 187 |
| 3.6.2 | Supprimer une valeur d'une série . . . . .   | 188 |
| 3.6.3 | Modifier les valeurs d'une série . . . . .   | 189 |
| 4.    | Structure de données Pandas : les objets de type DataFrame . . . . .   | 191 |
| 4.1   | Introduction . . . . .   | 191 |
| 4.2   | Indexing : sélectionner des valeurs d'un dataframe . . . . .   | 193 |
| 4.2.1 | Indexing et slicing avec l'attribut loc . . . . .  | 194 |
| 4.2.2 | Indexing et slicing avec l'attribut iloc . . . . .   | 198 |
| 4.2.3 | Indexing avec une expression booléenne . . . . .   | 199 |
| 4.3   | Ajout, suppression et modification sur un dataframe . . . . .  | 201 |
| 4.3.1 | Ajouter une ou plusieurs colonnes à un dataframe . . . . .   | 201 |
| 4.3.2 | Ajouter une ligne à un dataframe . . . . .   | 202 |
| 4.3.3 | Supprimer des lignes ou colonnes d'un dataframe . . . . .  | 205 |
| 4.3.4 | Modifier des valeurs dans un dataframe . . . . .   | 207 |
| 4.4   | Nettoyage et préparation des données avec Pandas . . . . .   | 209 |
| 4.4.1 | Gestion des données manquantes . . . . .   | 210 |
| 4.4.2 | Gestion des données dupliquées . . . . .   | 215 |
| 4.5   | Exploration préliminaire d'un dataframe . . . . .  | 219 |
| 4.5.1 | Principaux attributs . . . . .   | 219 |
| 4.5.2 | Définition des termes variable, variable quantitative<br>et variable qualitative et découverte<br>de la méthode describe() . . . . . | 222 |
| 4.5.3 | Méthodes de tri d'un dataframe . . . . .   | 225 |
| 5.    | Structure de données Pandas : les panels . . . . .   | 227 |
| 6.    | Manipulation avancée des données avec Pandas . . . . .   | 228 |
| 6.1   | Les opérations groupby . . . . .   | 228 |
| 6.1.1 | groupby sur une colonne . . . . .  | 228 |
| 6.1.2 | groupby sur plusieurs colonnes . . . . .   | 231 |

|  |     |
|--|-----|
| 6.1.3 Appliquer plusieurs fonctions<br>avec la méthode groupby et la méthode aggregate . . . . | 232 |
| 6.2 Appliquer une fonction à un dataframe<br>avec la méthode apply . . . . .                   | 234 |
| 6.3 Remodeler/réorganiser des dataframes . . . . .   | 236 |
| 6.3.1 Pivotage : la méthode pivot_table . . . . .  | 236 |
| 6.3.2 Les méthodes stack (empiler) et unstack (désempiler) .                                   | 238 |

## Chapitre 6

### Maîtriser la librairie Matplotlib

|  |     |
|--|-----|
| 1. Introduction . . . . .  | 241 |
| 2. Le fonctionnement de Matplotlib . . . . .   | 242 |
| 2.1 Architecture de Matplotlib . . . . .   | 242 |
| 2.2 Organisation des figures avec Matplotlib . . . . .                                       | 244 |
| 3. La création d'un premier graphique simple . . . . .                                       | 246 |
| 3.1 Préparer son jeu de données . . . . .  | 246 |
| 3.2 Créer un nuage de points . . . . .   | 250 |
| 3.3 Ajouter un titre principal et des labels<br>aux axes du nuage de points . . . . .        | 254 |
| 3.4 Enregistrer son graphique . . . . .  | 256 |
| 3.5 Changer la taille de la fenêtre graphique<br>et la résolution de son graphique . . . . . | 258 |
| 3.6 Tracer plusieurs courbes sur un même graphique<br>(sur un même objet axes) . . . . .     | 260 |
| 3.7 Ajouter une légende à son graphique . . . . .  | 263 |
| 3.8 Annoter son graphique avec du texte . . . . .  | 265 |
| 3.9 Combiner plusieurs graphiques grâce à subplot et subplots . .                            | 268 |
| 3.9.1 Tracer des sous-graphiques (subplot)<br>sur une ligne ou une colonne . . . . .         | 268 |
| 3.9.2 Tracer des sous-graphiques sur plusieurs lignes<br>et plusieurs colonnes . . . . .     | 271 |
| 3.9.3 Incruster un objet axes dans un autre . . . . .  | 274 |



# 8 \_\_\_\_\_ Python pour la Data Science

Analysez vos données par la pratique

|   |     |
|---|-----|
| 4. Les différents types de graphes .....  | 278 |
| 4.1 Types de graphiques selon les types de variables<br>(quantitatives et qualitatives) ..... | 278 |
| 4.2 Scatterplot .....   | 279 |
| 4.3 Graphique à barres (bargraph) .....   | 282 |
| 4.3.1 Graphique à barres simple .....   | 282 |
| 4.3.2 Graphique à barres groupées .....   | 287 |
| 4.3.3 Graphique à barres empilées .....   | 294 |
| 4.4 Boxplots .....  | 296 |

## Chapitre 7

### Maîtriser la librairie Seaborn

|   |     |
|---|-----|
| 1. Introduction .....   | 301 |
| 2. L'esthétique des figures avec Seaborn (Aesthetic) .....          | 303 |
| 2.1 Paramétrer les styles Seaborn (thèmes) .....                    | 304 |
| 2.2 Supprimer les axes .....  | 306 |
| 2.3 Paramétrer les contextes avec Seaborn .....                     | 309 |
| 2.4 Les palettes de couleur avec Seaborn .....                      | 314 |
| 2.4.1 Choisir une palette de couleurs existante .....               | 314 |
| 2.4.2 Créer sa propre palette de couleurs .....                     | 317 |
| 3. Les différents types de graphiques .....                         | 318 |
| 3.1 Préparation du jeu de données .....                             | 318 |
| 3.2 Nuage de points (scatterplot) .....                             | 318 |
| 3.3 Graphiques de régression .....                                  | 323 |
| 3.4 Pointplot .....   | 327 |
| 3.5 Nuage de points avec une variable qualitative : stripplot ..... | 331 |
| 3.6 Boxplots .....  | 334 |
| 3.7 Graphique à barres : countplot .....                            | 336 |
| 3.8 Histogrammes .....  | 339 |
| 3.9 Jointplot .....   | 344 |
| 3.10 Pairplot .....   | 346 |
| 3.11 Heatmap .....  | 349 |

- 4. Les graphiques multi-grilles ..... 352
  - 4.1 FacetGrid ..... 352
  - 4.2 PairGrid ..... 354
  - 4.3 JointGrid ..... 359
- 5. Conclusion ..... 361

**Chapitre 8**

**Exercice complet sur jeu de données réel**

- 1. Introduction ..... 363
- 2. Présentation du jeu de données ..... 365
- 3. Énoncé de l'exercice ..... 366
  - 3.1 Lire le fichier ..... 366
  - 3.2 Afficher les dimensions du dataframe ..... 367
  - 3.3 Compter les films et les séries ..... 367
  - 3.4 Générer le résumé statistique du dataframe ..... 367
  - 3.5 Compter les valeurs manquantes ..... 368
  - 3.6 Explorer les valeurs manquantes ..... 368
    - 3.6.1 Sur la colonne des directeurs de production ..... 368
    - 3.6.2 Sur la colonne des acteurs ..... 368
  - 3.7 Supprimer les lignes dupliquées ..... 369
  - 3.8 Compter les films/séries produits  
par les États-Unis et par la France ..... 369
  - 3.9 Afficher le contenu le plus vieux disponible sur Netflix ..... 370
  - 3.10 Afficher le film avec la durée la plus longue sur Netflix ..... 370
    - 3.10.1 Nouvelle notion : les méthodes str ..... 370
    - 3.10.2 Énoncé ..... 371
  - 3.11 Étudier les catégories avec le plus de contenu ..... 372
  - 3.12 Afficher les directeurs qui ont produit  
le plus de films/séries disponibles sur Netflix ..... 375
  - 3.13 Voir si Jan Suter travaille souvent avec les mêmes acteurs ... 375

# 10 \_\_\_\_\_ Python pour la Data Science

Analysez vos données par la pratique

|  |     |
|--|-----|
| 3.14 Représenter les dix pays qui ont produit le plus de contenus disponibles sur Netflix, avec le nombre de contenus par pays . . . . . | 376 |
| 3.15 Tracer un graphe à barres du nombre de films/séries par classement de contenu (rating) . . . . .                                    | 377 |
| 3.16 Afficher l'évolution du nombre de films/séries disponibles sur Netflix au cours du temps . . . . .                                  | 377 |
| 3.16.1 Notions supplémentaires sur les dates . . . . .   | 377 |
| 3.16.2 Énoncé . . . . .  | 377 |
| 3.17 Afficher la distribution de la durée des films disponibles sur Netflix . . . . .  | 378 |
| 3.18 Tracer un graphique représentant le nombre de séries par modalité de nombre de saisons . . . . .                                    | 379 |
| <br>   |     |
| Index . . . . .  | 381 |

## Chapitre 4

# Maîtriser la librairie NumPy

### 1. Introduction à NumPy

NumPy est la librairie Python dédiée au calcul scientifique fournissant des fonctions très performantes de calcul, mais aussi des structures de données, tout aussi performantes.

En Data Science, il est essentiel d'avoir des structures adaptées pour stocker et manipuler de grandes quantités de données. C'est là qu'intervient NumPy, qui intègre une nouvelle structure de données en Python, les `ndarrays` (tableaux à N dimensions, en français, N représentant un chiffre), qui sont des tableaux multidimensionnels ou matrices. Il est important de noter que cette structure de données permet de stocker des données uniquement de même type (uniquement des nombres entiers par exemple).

Les `ndarrays` sont optimisés pour le stockage de données, mais aussi pour leur manipulation et plus encore pour les calculs, car ceux-ci peuvent être vectorisés. NumPy peut gérer de très gros tableaux et est très performante en temps de calcul sur ces tableaux : les `ndarrays` prennent en effet moins de place mémoire que d'autres objets Python, comme par exemple les listes. C'est pour cela que cette librairie a été développée et qu'elle est si utilisée : elle est très performante. C'est également pour cette raison que de nombreuses librairies ont été développées au-dessus de celle-ci, telle que Pandas, que nous verrons plus tard dans ce livre.

# 94 \_\_\_\_\_ Python pour la Data Science

Analysez vos données par la pratique

Les `ndarrays` de NumPy peuvent être unidimensionnels (aussi appelés 1D array, ce qu'on peut voir comme une liste), bidimensionnels (2D array) donc un tableau avec des lignes et des colonnes, ou encore des tableaux à plus de deux dimensions (3D array, 4D array, 5D array...), que nous ne verrons pas dans ce livre. Nous travaillerons exclusivement avec les `ndarrays` à deux dimensions dans ce chapitre, qui est la structure de données la plus utilisées en Data Science pour manipuler de grands jeux de données.

Lorsque vous souhaitez effectuer des opérations mathématiques ou logiques rapides sur un grand jeu de données, NumPy est votre allié. Pour pouvoir utiliser NumPy sous Python, il suffit de charger la librairie sous Jupyter, celle-ci étant déjà installée dans la distribution Anaconda.

## Syntaxe

```
import numpy as np
```

### ■ Remarque

*Il est courant d'utiliser l'alias "np" pour la librairie NumPy.*

### ■ Remarque

*Pour la suite de ce chapitre, n'hésitez pas à tester les codes que nous proposerons directement dans le notebook lié à ce chapitre et disponible en téléchargement.*

## 2. Les tableaux NumPy

### 2.1 Créer un `ndarray`

#### 2.1.1 Créer un `ndarray` à partir de listes

On peut créer un tableau NumPy à partir d'une liste en utilisant la fonction `array()` afin de convertir cette liste en tableau.

## Syntaxe

```
import numpy as np
notre_tableau=np.array([élément1,élément2,élément3,élément4])
```

## Exemple de code

```
import numpy as np
notre_tableau=np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
18,19,20])
print(notre_tableau)
type(notre_tableau)
```

## Résultat

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
numpy.ndarray
```

Ici, on importe la librairie NumPy, on crée un tableau à partir d'une liste, on affiche ce tableau avec `print()` et on affiche le type de l'objet `notre_tableau`. En résultat, on a notre premier array NumPy, qui est un tableau à une dimension ici, qu'on pourrait considérer comme une liste, mais qui est bien de type `ndarray`.

Pour créer un tableau bidimensionnel, il faut créer une liste contenant des listes, ce qui permet de représenter un tableau à deux entrées : les lignes et les colonnes. Pour cela, il suffit de considérer que la liste qu'on donne à la fonction `array()` est une liste de lignes et que les listes contenues dans cette liste représentent les colonnes. Ainsi, chaque ligne contient une liste qui correspond aux colonnes de cette ligne.

## Syntaxe

```
mon_array_bidimensionnel=np.array([[élément ligne 1 colonne 1,
élément ligne 1 colonne 2,élément ligne 1 colonne 3],
[élément ligne 2 colonne 1, élément ligne 2 colonne 2,
élément ligne 2 colonne 3]])
```

## Code

```
import numpy as np
mon_array_bidimensionnel=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(mon_array_bidimensionnel)
```

## Résultat

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

# 96 Python pour la Data Science

Analysez vos données par la pratique

On remarque tout de suite que, visuellement, cela ressemble à un tableau à deux dimensions, dont la première ligne contient trois colonnes avec les chiffres 1,2,3, la deuxième ligne (la deuxième liste de la liste principale) également trois colonnes avec les chiffres 4, 5, 6, etc.

Il s'agit d'un tableau d'entiers ici, et on ne pourrait pas changer une valeur par un type caractère par exemple, cela générerait une erreur.

## Code

```
mon_array_bidimensionnel[0,0]="texte"
```

## Résultat

```
ValueError: invalid literal for int() with base 10: 'texte'
```

Ici, nous disons à Python que nous souhaitons modifier la valeur de la première ligne à la première colonne par la chaîne de caractères "texte". Python nous retourne une erreur en nous disant que "texte" n'est pas de type `int` (entier), donc ça ne fonctionne pas. Comme nous le disions, un `ndarray` ne peut contenir qu'un seul type de données. Pour connaître le type des données contenues dans un `ndarray`, il suffit d'utiliser l'attribut `dtype`.

## Syntaxe

```
mon_array.dtype
```

Regardons le type de données de notre tableau d'exemple.

## Exemple de code

```
mon_array_bidimensionnel.dtype
```

## Résultat

```
dtype('int32')
```

Ici, le type est `int32`, notre tableau ne peut donc contenir que des nombres entiers, d'où l'erreur précédente.

## Remarque

*Il existe deux types `int` : `int32` et `int64`. La différence entre `int32` et `int64`, c'est globalement la capacité de stockage du type. `int32` peut stocker des nombres entiers entre -2 147 483 648 et 2 147 483 647 et `int64` peut stocker des nombres entiers entre -9 223 372 036 854 775 808 et +9 223 372 036 854 775 808.*

Il est possible de spécifier le type des données du tableau qu'on souhaite créer, avec l'option `dtype` dans la fonction `array()` de NumPy.

## Code

```
mon_array_bidimensionnel=np.array([[1.5,2,3],[4,5.2,6],[7,8,9.1]],  
dtype = float)  
print(mon_array_bidimensionnel)  
mon_array_bidimensionnel.dtype
```

Ici, on crée un tableau en spécifiant le type, `float`, puis on affiche le contenu de ce tableau. Enfin, on affiche le type des données contenues dans ce tableau.

## Résultat

```
[[1.5 2.  3. ]  
 [4.  5.2 6. ]  
 [7.  8.  9.1]]  
Out[9]:  
dtype('float64')
```

Il s'agit bien d'un tableau contenant des données de type `float`, donc des nombres à virgule. Tous les entiers qu'on a donnés lors de la création du tableau (par exemple 2, 3, 4...) sont transformés en réels, `float`, et sont donc suivis de ".0".

### 2.1.2 Créer un `ndarray` grâce à des fonctions NumPy

Il arrive que vous sachiez d'avance la taille de votre `ndarray` mais pas encore son contenu, car vous souhaitez remplir ce tableau au fur et à mesure de votre code. NumPy fournit des fonctions permettant de créer et initialiser des `ndarrays` de taille connue et de contenu inconnu.

La première fonction est `np.zeros()`. Cette fonction permet de créer un `ndarray` dont l'ensemble des éléments correspondent à la valeur zéro.



# 98 Python pour la Data Science

Analysez vos données par la pratique

## Syntaxe

```
np.zeros((nombre de lignes, nombre de colonnes))
```

Il suffit de donner le nombre de lignes et de colonnes que vous souhaitez voir apparaître dans votre tableau.

## Code

```
import numpy as np
mon_array_zeros=np.zeros((4,6))
print(mon_array_zeros)
```

## Résultat

```
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
```

On obtient un tableau de type `float`, par défaut, rempli de zéros et de dimensions quatre lignes et six colonnes. Si on veut, on peut spécifier que le tableau est de type `int`.

## Code

```
import numpy as np
mon_array_zeros=np.zeros((4,6), dtype=int)
print(mon_array_zeros)
```

## Résultat

```
[[0 0 0 0 0 0]
 [0 0 0 0 0 0]
 [0 0 0 0 0 0]
 [0 0 0 0 0 0]]
```

## Remarque

*Il est aussi possible de créer un `ndarray` à une dimension contenant uniquement des zéros. Pour cela, plutôt que de donner un tuple contenant le nombre de lignes et de colonnes, il suffit de donner un seul nombre à la fonction `np.zeros` : `np.zeros(5)` créera un tableau à une dimension contenant 5 valeurs 0.*

Il existe ensuite la fonction `np.ones()`. Cette fonction permet de créer un `ndarray` dont l'ensemble des éléments correspondent au chiffre 1.

### Syntaxe

```
np.ones((nombre de lignes, nombre de colonnes))
```

### Exemple de code

```
import numpy as np
mon_array_ones=np.ones((4,6), dtype=int)
print(mon_array_ones)
```

### Résultat

```
[[1 1 1 1 1 1]
 [1 1 1 1 1 1]
 [1 1 1 1 1 1]
 [1 1 1 1 1 1]]
```

Enfin, il existe la fonction `np.empty()`. Cette fonction permet de créer un `ndarray` dont les valeurs des éléments sont aléatoires.

### Syntaxe

```
np.empty((nombre de lignes, nombre de colonnes))
```

### Exemple de code

```
import numpy as np
mon_array_empty=np.empty((4,6), dtype=int)
print(mon_array_empty)
```

### Résultat

```
[[-1805877096          611           64           0           0           0]
 [           0           0           0           0  929446194  912471142]
 [ 1667510839  909654373  909193776 1664104498 1630941233  929260599]
 [  959657314  878982705 1650615603  929457204 1697985846  842152292]]
```