

Démarrer avec Python

Alison PATOU

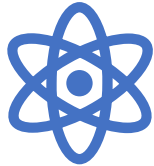
Patou.alison@gmail.com



Programme



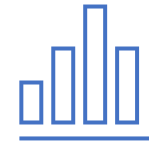
Introduction



Jupyter
Notebook



Langage Python



Machine
Learning

1

Introduction

Le langage Python

- Développé en 1989 par Guido van Rossum
- **Gratuit** : mais on peut l'utiliser sans restriction dans des projets commerciaux. Il est facilement téléchargeable sur www.python.org
- **Syntaxe aisée** : la syntaxe de Python est très simple et, combinée à de nombreux types de données évolués (comme les listes, dictionnaires, tuples...)
- **Portable** : on peut utiliser le même programme sur presque tous les systèmes d'exploitation (Linux/UNIX, Windows, Mac, OS/2, ...)
- **Orienté objet** : Il supporte l'héritage multiple et la surcharge des opérateurs
- **Extensible** : possède une multitude de bibliothèques dans différents domaines (BDD, Web, calcul scientifique, ...), qui peuvent être adaptées ou complétées afin d'être plus productifs
- **Communauté** : soutenu par une communauté d'utilisateurs enthousiastes, vous trouverez de nombreuses résolutions en ligne.

Écriture d'un programme : syntaxe

→ Instructions : des commandes impliquant l'emploi de mots-clés

➤ Entrées

- Fonction **input()**

```
>>> nom = input()
```

l'utilisateur tape son nom, qui stocké dans la variable « nom »

➤ Sorties

- Fonction **print()**

```
>>> print(nom)
```

Le nom de l'utilisateur s'affiche à l'écran

➤ ... autres commandes dans la suite du cours

→ Expressions : elles n'utilisent pas de mots-clés

➤ Equations arithmétiques

➤ Appel de fonctions

Écriture d'un programme : syntaxe

→ Quelques exemples

- Afficher « hello world » à l'écran

```
>>> print("hello world")
```

- Demander le nom de l'utilisateur et afficher le à l'écran

```
>>> nom = input("Quel est votre nom :")
```

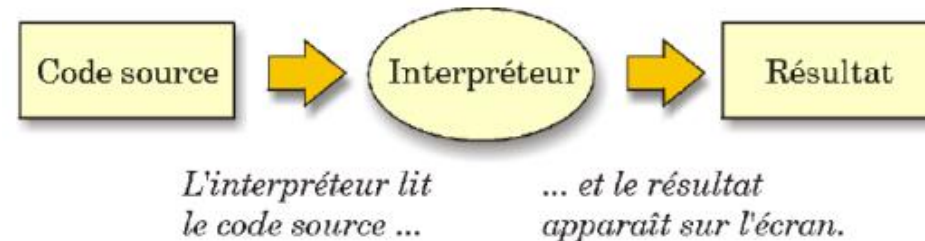
```
>>> print(nom)
```

Compilation et exécution du programme

Deux techniques principales pour faire la traduction d'un code source (programme) en langage machine :

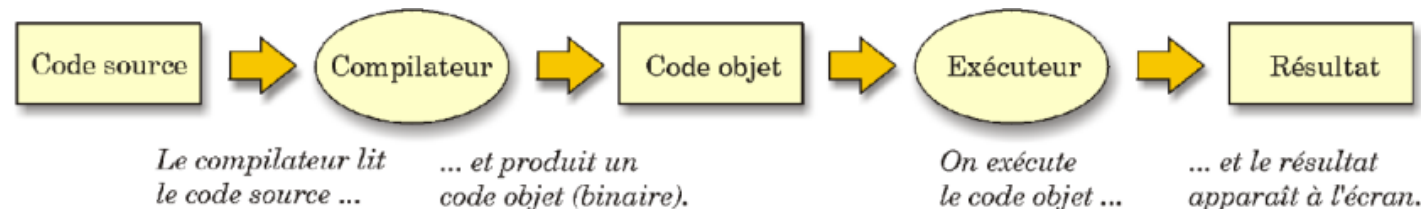
→ Compilation

- Le compilateur, transforme le code source en code machine directement exécutable par le processeur



→ Interprétation

- L'interpréteur, lit chaque ligne du code source et l'interprète en terme de code à faire exécuter par le processeur

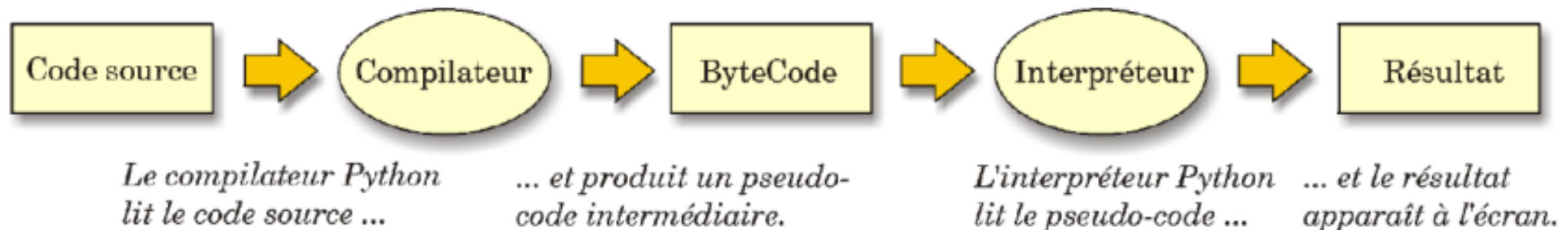


Compilation et exécution du programme

En Python

→ Compilation / Interprétation

- Le code source est compilé en code intermédiaire pour une machine virtuelle, lui-même interprété par le programme simulant cette machine virtuelle sur la machine physique



2

Jupyter Notebook

Notebook

Qu'est ce que c'est ?

Un notebook contient le code du développeur mais aussi les différentes étapes d'analyses, les visualisations, les commentaires et des découpages grâce à des titres et sous-titres pour une meilleure lisibilité.

Cela permet notamment une meilleure :

- Efficacité
- Interactivité
- Collaborativité
- Reproductibilité
- Automatisation
- ...

Notebook

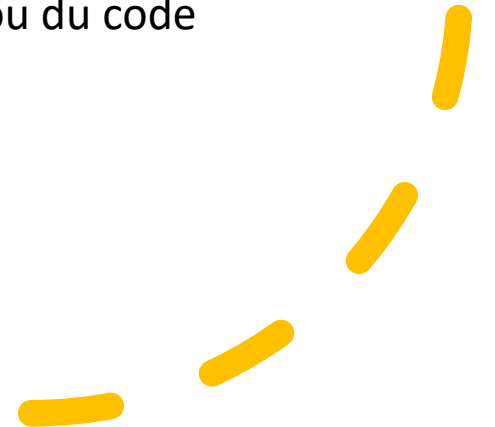
Quand l'utiliser ?

Le notebook s'utilise durant les différentes étapes d'analyse :

- **Nettoyage des données** : faire le tri entre les données importantes et celles qui ne le sont pas dans l'analyse des ensembles de mégadonnées
- **Modélisation statistique** : méthode mathématique permettant d'établir la probabilité de répartition d'une caractéristique particulière
- **Création et mise en œuvre de modèles d'apprentissage automatique** : étude, programmation et apprentissage de modèles
- **Visualisation de données** : représentation graphique de données pour faire apparaître des structures, des tendances, des relations, etc.

Jupyter Notebook

- Les notebooks Jupyter sont des cahiers électroniques qui, dans le même document, peuvent rassembler du texte, des images, des formules mathématiques et du code informatique exécutable. Ils sont manipulables interactivement dans un navigateur web.
- Initialement développés pour les langages de programmation Julia, Python et R (d'où le nom Jupyter), les notebooks Jupyter supportent près de 40 langages différents.
- La cellule est l'élément de base d'un notebook Jupyter. Elle peut contenir du texte formaté au format Markdown ou du code informatique qui pourra être exécuté.

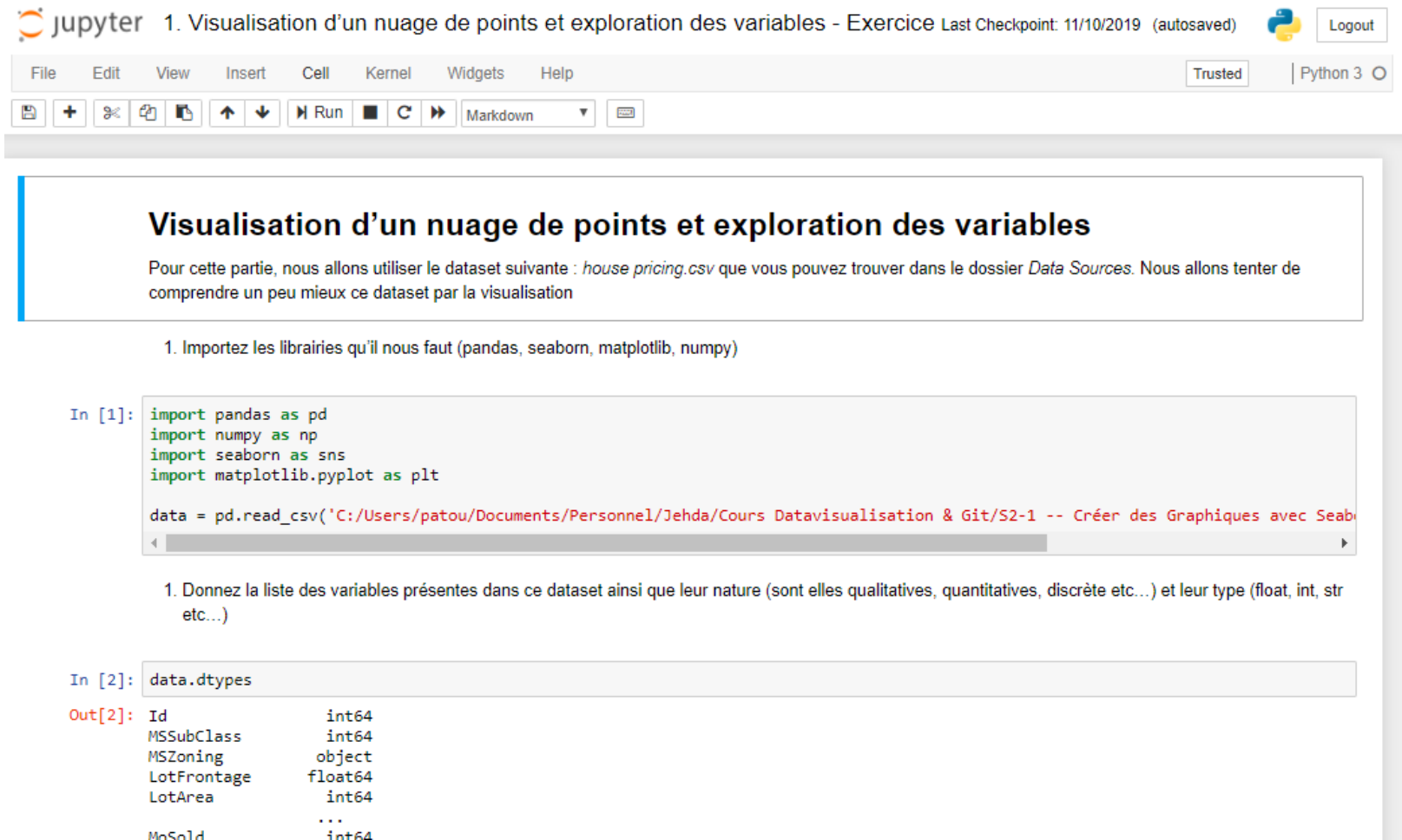


Jupyter Notebook

- Les notebooks Jupyter sont des cahiers électroniques qui, dans le même document, peuvent rassembler du texte, des images, des formules mathématiques et du code informatique exécutable. Ils sont manipulables interactivement dans un navigateur web.
- Initialement développés pour les langages de programmation Julia, Python et R (d'où le nom Jupyter), les notebooks Jupyter supportent près de 40 langages différents.
- La cellule est l'élément de base d'un notebook Jupyter. Elle peut contenir du texte formaté au format Markdown ou du code informatique qui pourra être exécuté.

Jupyter Notebook

Voici à quoi ressemble un notebook Jupyter :



The screenshot displays a Jupyter Notebook interface. At the top, the Jupyter logo is followed by the title "1. Visualisation d'un nuage de points et exploration des variables - Exercice". To the right of the title, it says "Last Checkpoint: 11/10/2019 (autosaved)" and a "Logout" button. Below the title bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. To the right of the menu bar are "Trusted" and "Python 3" indicators. Below the menu bar is a toolbar with icons for saving, adding cells, zooming, and running code. The main content area has a title "Visualisation d'un nuage de points et exploration des variables" and a paragraph: "Pour cette partie, nous allons utiliser le dataset suivante : *house pricing.csv* que vous pouvez trouver dans le dossier *Data Sources*. Nous allons tenter de comprendre un peu mieux ce dataset par la visualisation". Below this is a numbered instruction: "1. Importez les librairies qu'il nous faut (pandas, seaborn, matplotlib, numpy)". This is followed by a code cell with the following code:

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv('C:/Users/patou/Documents/Personnel/Jehda/Cours Datavisualisation & Git/S2-1 -- Créer des Graphiques avec Seaborn/house pricing.csv')
```

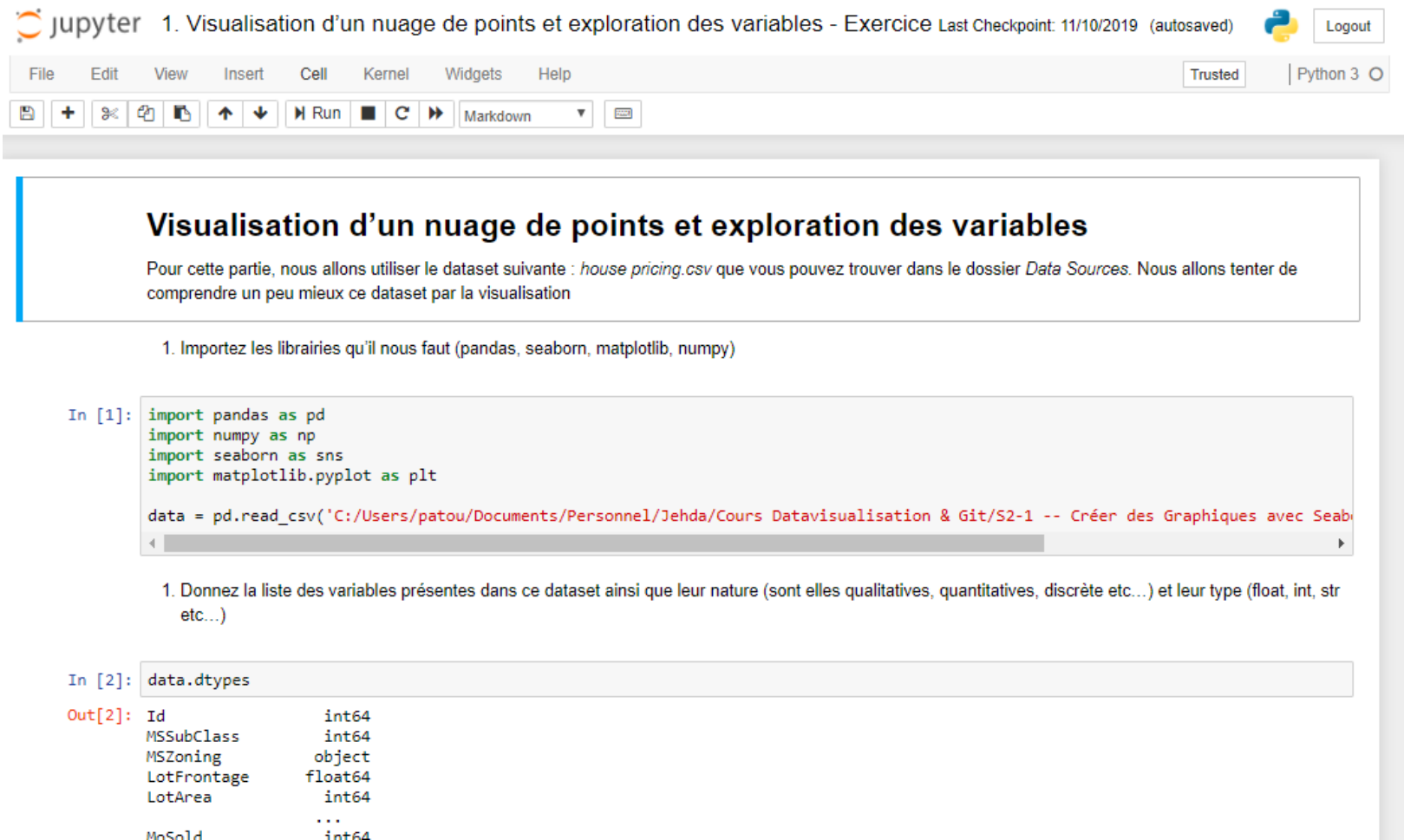
 Below the code cell is another numbered instruction: "1. Donnez la liste des variables présentes dans ce dataset ainsi que leur nature (sont elles qualitatives, quantitatives, discrète etc...) et leur type (float, int, str etc...)". This is followed by another code cell:

```
In [2]: data.dtypes
```

 The output of this cell is shown below:

```
Out[2]: Id                int64
MSSubClass              int64
MSZoning                object
LotFrontage             float64
LotArea                 int64
...
MoSold                  int64
```

Jupyter Notebook



The screenshot shows a Jupyter Notebook interface. At the top, there's a header bar with the Jupyter logo, the title "1. Visualisation d'un nuage de points et exploration des variables - Exercice", the last checkpoint "11/10/2019 (autosaved)", and a "Logout" button. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Trusted" and "Python 3" indicators. Below the menu bar is a toolbar with icons for saving, adding, deleting, and running cells, as well as a dropdown menu for "Markdown".

The notebook content starts with a title "Visualisation d'un nuage de points et exploration des variables" in bold. Below the title is a paragraph of text: "Pour cette partie, nous allons utiliser le dataset suivante : *house pricing.csv* que vous pouvez trouver dans le dossier *Data Sources*. Nous allons tenter de comprendre un peu mieux ce dataset par la visualisation".

Below the text is a numbered list item: "1. Importez les librairies qu'il nous faut (pandas, seaborn, matplotlib, numpy)".

Below the list item is a code cell. The input is "In [1]:". The code is:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv('C:/Users/patou/Documents/Personnel/Jehda/Cours Datavisualisation & Git/S2-1 -- Créer des Graphiques avec Seaborn/house pricing.csv')
```

Below the code cell is another numbered list item: "1. Donnez la liste des variables présentes dans ce dataset ainsi que leur nature (sont elles qualitatives, quantitatives, discrète etc...) et leur type (float, int, str etc...)".

Below the list item is another code cell. The input is "In [2]:". The code is:

```
data.dtypes
```

Below the code cell is the output. The output is "Out[2]:". The output is a table showing the data types for each variable in the dataset:

Variable	Type
Id	int64
MSSubClass	int64
MSZoning	object
LotFrontage	float64
LotArea	int64
...	...
MoSold	int64

Titre, explications

Partie de code en Python

Affichage de résultats

Jupyter Notebook

The image shows a Jupyter Notebook interface with several annotations in blue boxes and lines:

- Ajoute un input**: Points to the '+' icon in the toolbar.
- Exécute le code**: Points to the 'Run' button in the toolbar.
- Passe de l'écriture texte au code**: Points to the 'Markdown' dropdown menu in the toolbar.

The notebook content includes:

- Header**: "1. Visualisation d'un nuage de points et exploration des variables - Exercice" with a "Last Checkpoint: 11/10/2019 (autosaved)" and "Logout" button.
- Toolbar**: Contains icons for file operations, a '+' button, a 'Run' button, and a 'Markdown' dropdown.
- Section Title**: "Visualisation d'un nuage de points et exploration des variables".
- Text**: "Pour cette partie, nous allons utiliser le dataset suivante : *house pricing.csv* que vous pouvez trouver dans le dossier *Data Sources*. Nous allons tenter de comprendre un peu mieux ce dataset par la visualisation".
- Code Cell 1**:

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv('C:/Users/patou/Documents/Personnel/Jehda/Cours Datavisualisation & Git/S2-1 -- Créer des Graphiques avec Seaborn/house pricing.csv')
```
- Text**: "1. Donnez la liste des variables présentes dans ce dataset ainsi que leur nature (sont elles qualitatives, quantitatives, discrète etc...) et leur type (float, int, str etc...)"
- Code Cell 2**:

```
In [2]: data.dtypes
```
- Output**:

```
Out[2]: Id                int64
MSSubClass             int64
MSZoning                object
LotFrontage            float64
LotArea                int64
...
MoSold                 int64
```

On the left side, there are labels for the input and output sections:

- INPUT**: A bracket grouping the first code cell and its associated text.
- OUTPUT**: A bracket grouping the second code cell and its output.

3

Le langage Python

Qu'est-ce qu'une variable ?

- Une **variable** est une zone de la mémoire dans laquelle une **valeur** est stockée.
- Pour les programmeurs, la variable est définie par un **nom** et elle permet de mémoriser une donnée.
- Pour les ordinateurs, il s'agit d'une **référence** désignant une **adresse mémoire**, c'est-à-dire un emplacement précis dans la mémoire vive.
- Une variable possède par définition :
 - Un **nom** (une étiquette)
 - Un **type** (int, float, str, boolean, ...)
 - Une **valeur** (l'information)



Les types primitifs

→ Le type d'une variable correspond à la nature de celle-ci.

→ Les types principaux :

- Entiers → *int* Exp : unNombreEntier= 16
- Réels → *floats* Exp : unNombreFlottant= 21.3462
- Chaines de caractères → *str* Exp. : uneChaine= « bonjour »
- Booléen → *bool* (True ou False) Exp : unBooleen = True | unAutre = not(unBooleen)

→ Il existe de nombreux autres types en Python (nombres complexes, listes, etc.).

- Pour plus de détails : <https://docs.python.org/fr/3.7/library/stdtypes.html>

→ En python, le typage est dynamique.

- En affectant une valeur à une variable, elle sera automatiquement créée avec le type correspond au mieux à la valeur fournie.

Les types primitifs

→ La fonction « type » :

- Python comprend automatiquement de quel type est une variable et cela lors de son affectation. Mais il est pratique de pouvoir savoir de quel type est une variable.
- Syntaxe : ***type(nom_de_la_variable)***
- La fonction renvoie le type de la variable passée en paramètre.
- **Exp :**

```
>>> a = 3  
>>> type(a)
```

Le résultat est ?

Les types primitifs

→ La fonction « type » :

- Python comprend automatiquement de quel type est une variable et cela lors de son affectation. Mais il est pratique de pouvoir savoir de quel type est une variable.
- Syntaxe : ***type(nom_de_la_variable)***
- La fonction renvoie le type de la variable passée en paramètre.
- **Exp :**

```
>>> a = 3  
>>> type(a)
```

Le résultat est :

<class 'int'>

Python vous indique donc que la variable a appartient à la classe des entiers.

- **Autres Exps :**

```
>>> type(3.4)  
<class 'float'>  
>>> type("un essai")  
<class 'str'>  
>>>
```

Les constantes.

- En Python, la déclaration d'une variable et son initialisation (c'est-à-dire la première valeur que l'on va stocker dedans) se font en même temps.
 - **Exp :** `age = 20` → Déclarer et initialiser la variable `x` avec la valeur 20
- En réalité, dans cet exemple, il s'est passé trois étapes en une fois :
 - Python a « deviné » que la variable était un entier. On a dit que Python est un langage au **typage dynamique**.
 - Python a alloué (réservé) l'espace en mémoire pour y accueillir un entier. Chaque type de variable prend plus ou moins d'espace en mémoire. Python a aussi fait en sorte qu'on puisse retrouver la variable sous le nom **age**.
 - Enfin, Python a assigné la valeur 20 à la variable **age**.
- Une **constante** est une variable dont la valeur ne varie pas.
 - **Exp :** `PI = 3.141` → Déclarer et affecter 3.141 à la constante `PI`

Saisie, affichage, affectation, conversion de type.

→ Saisie : fonction `input()`

- Elle provoque une interruption dans le programme courant où l'utilisateur est invité à entrer des données au clavier et à terminer avec **<Enter>**.
- L'exécution du programme se poursuit alors et la fonction fournit en retour les valeurs entrées par l'utilisateur.
- Elle est soit, sans paramètre ou soit, avec un seul paramètre, une chaîne de caractères, lequel est un message explicatif destiné à l'utilisateur.

```
>>> nom = input("Entrez votre nom : ")  
>>> print(nom)
```

→ Affichage : fonction `print()`

- Elle permet d'afficher la valeur d'une ou plusieurs variables.
- Le premier print affiche la valeur de la variable a, c'est-à-dire « 3 ».
- Le second print affiche : a = 6 et b = 4

```
>>> a = 3  
>>> print(a)  
>>> a = a + 3  
>>> b = a - 2  
>>> print("a =", a, "et b =", b)
```

Saisie, affichage, affectation, conversion de type.

→ Affectation :

- Elle désigne l'opération par laquelle on établit un lien entre le nom de la variable et sa valeur (son contenu).
- Les termes « affecter une valeur » ou « assigner une valeur » à une variable sont équivalents.
- En Python comme dans de nombreux autres langages, l'opération d'affectation est représentée par le signe *égale*.

▪ **Exp :** `note = 13` → relier un contenu (la valeur 13) à un contenant (la variable `note`)

➤ Affectation multiple :

- On peut attribuer la même valeur à plusieurs variables simultanément.

```
>>> x = y = 7
>>> x
>>> y
```

➤ Affectation parallèle :

- On peut aussi effectuer des affectations parallèles à l'aide d'un seul opérateur.

```
>>> a, b = 4, 8.33
>>> a
>>> b
```


Saisie, affichage, affectation, conversion de type.

→ Conversion de type :

- Passer d'un type de données vers un autre.
- Ce changement de type est appelé **transtypage** ou conversion de type, ou **cast** en anglais.
- Principe : Utilisation du mot-clé désignant le type

nouveau_type(objet)

- Certains transtypes ne sont pas autorisés (la valeur affectée à la variable ne peut pas être convertie vers le type désiré) :

```
>>> float(3)
>>> int(3.7)
>>> str(3.4)
>>> float("3.4")
>>> bool(0)
>>> bool("TRUE")
>>> int(True)
```

```
>>> int("bonjour")
ValueError: invalid literal for int() with base 10: 'bonjour'
```

Qu'est-ce qu'une librairie ? Son rôle, son usage.

- Une bibliothèque ou librairie logicielle est un ensemble de fonctions utilitaires, regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire.
 - Les fonctions sont regroupées de par leur appartenance à un même domaine conceptuel (mathématique, graphique, tris, etc)
- Intérêt : elles contiennent du code utile et réutilisable pour les nouveaux développements.
 - Favorisent le facteur d'extensibilité des langages de programmation
 - Accélèrent la productivité des programmeurs et permettent un gain de temps considérable

Qu'est-ce qu'une librairie ? Son rôle, son usage.

- **Pour faire du Machine Learning, Deep Learning ou tout simplement de la manipulation de données, les bibliothèques les plus utilisées sont :**
 - Pandas
 - Numpy
 - Keras
 - Tensorflow
- **Pour la partie visualisation des données, on utilise souvent :**
 - Matplotlib
 - Seaborn

Qu'est-ce qu'une librairie ? Son rôle, son usage.

- ❑ Télécharger une librairie :

```
pip install seaborn
```

- ❑ Charger la librairie (doit se faire à chaque utilisation de fonctions de cette librairie)

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Import de données

- ❑ Il existe une multitude de fonction pouvant importer des données, selon le type de source de données en entrée (SGBD, BDD dans le cloud, fichiers plats ...) et selon la fonction/librairie utilisée.

```
data = pd.read_csv('C:/Users/patou/Documents/Personnel/EPXI/Cours/Big Data et IA/Démarrer en Python\house_pricing.csv')
```

Attention à bien remplacer les « \ » par des « / » dans votre chemin d'accès au fichier

Vous pouvez récupérer facilement le chemin d'accès d'un fichier en cliquant sur CTRL + ALT + CLIQUE DROIT : « Copier le chemin d'accès »

Les différents opérateurs

Opérateur	Signification
>	Supérieur strict
<	Inférieur strict
>=	Supérieur ou égal
<=	Inférieur ou égal
==	Égal à(Attention à bien mettre le double égal sinon c'est comme si vous assigniez une nouvelle valeur à une variable)
!= (ou <>)	Différent de

Sélection

Vous allez souvent avoir besoin de sélectionner une partie de vos données, vecteurs ou autre.
Gardez donc en tête la structure suivante :

```
In [35]: dataset_exemple = [1,2,3,4,5,6,7,8,9,10]
trois_derniers = dataset_exemple[7:10]
trois_derniers
```

```
Out[35]: [8, 9, 10]
```



Je connais mes « bornes »
de sélection

```
In [36]: dataset_exemple = [1,2,3,4,5,6,7,8,9,10]
trois_derniers_bis = dataset_exemple[7:]
trois_derniers_bis
```

```
Out[36]: [8, 9, 10]
```



Je connais ma « borne »
inférieure et je veux sélectionner
jusqu'à mon item max

```
In [37]: dataset_exemple = [1,2,3,4,5,6,7,8,9,10]
quatre_derniers = dataset_exemple[-4:]
quatre_derniers
```

```
Out[37]: [7, 8, 9, 10]
```



Je veux les derniers indices

4

Data Visualisation en Python

Variables qualitatives

- Les caractères qualitatifs sont ceux dont les modalités ne peuvent pas être ordonnées, c'est-à-dire que si l'on considère deux caractères pris au hasard, on ne peut pas dire de l'un des caractères qu'il est inférieur ou égal à l'autre.
- Exemple : La région, le pays, couleurs sont des variables qualitatives

Variables quantitatives

- Les caractères quantitatifs sont des caractères dont les modalités peuvent être ordonnées.
- Exemple : l'âge, la taille de vie ou le salaire d'un individu sont des caractères quantitatifs

Effectif

L'effectif de la valeur x_i est le nombre d'individus de la population ayant cette valeur ou appartenant à cette classe : on le note n_i .

L'effectif total N est la somme de tous les effectifs : $N = n_1 + n_2 + \dots + n_k$.

En rangeant les valeurs du caractère dans l'ordre croissant, on peut calculer l'effectif cumulé croissant en faisant la somme des effectifs de cette valeur et de tous ceux qui la précèdent.

Exemple

Note	19	11	8	12	10	17	8	10	12
Note	8	10	11	12	17	19	Effectif		
Effectif	2	2	1	1	1	1			
Note	8	10	11	12	17	19			
Effectif	2	2	1	1	1	1			
Eff. cumulé croissant	2	4	5	6	7	8			

Fréquence

La fréquence d'une valeur est le quotient de l'effectif de la valeur par l'effectif total.

En rangeant les valeurs du caractère dans l'ordre croissant, on peut calculer les fréquences cumulées croissantes en faisant la somme des fréquences de cette valeur et de tous ceux qui la précèdent.

La fréquence est comprise entre 0 et 1

Exemple

Note	8	10	11	12	17	19
Effectif	2	2	1	1	1	1
Eff. cumulé croissant	2	4	5	6	7	8
Fréquence	0,25	0,25	0,125	0,125	0,125	0,125
Fréq. Cumulée croissant	0,25	0,5	0,625	0,75	0,875	1

Dataset utilisé

Le dataset utilisé pour illustrer les différentes commandes nécessaires en Python est le dataset HousePricing :

Out[10]:

tArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2008	WD	Normal	208500
9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	5	2007	WD	Normal	181500
11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	9	2008	WD	Normal	223500
9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2006	WD	Abnorml	140000
14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	12	2008	WD	Normal	250000
...
7917	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	8	2007	WD	Normal	175000
13175	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0	2	2010	WD	Normal	210000
9042	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	GdPrv	Shed	2500	5	2010	WD	Normal	266500
9717	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	4	2010	WD	Normal	142125
9937	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	6	2008	WD	Normal	147500

Ce dataset regroupe plusieurs variables et indicateurs sur les ventes de biens immobiliers.

Fonctions utiles

- **Value_counts()** permet de réaliser des calculs sur un vecteur, conditionnellement aux valeurs prises par un ou plusieurs facteurs

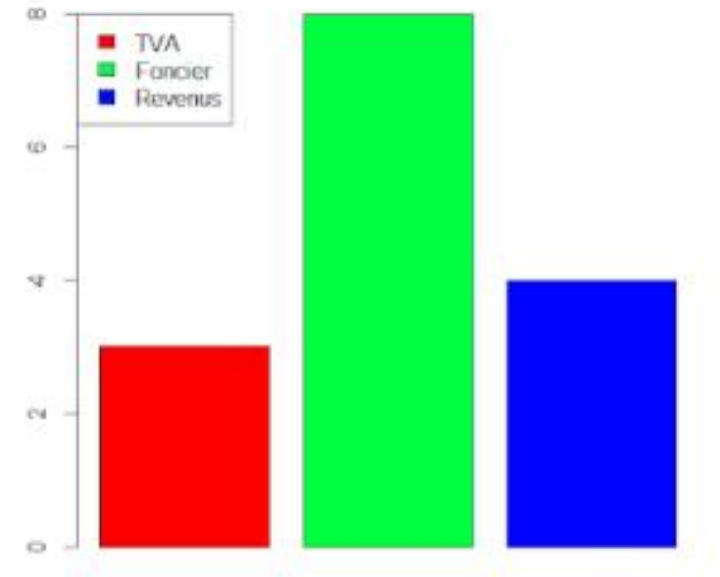
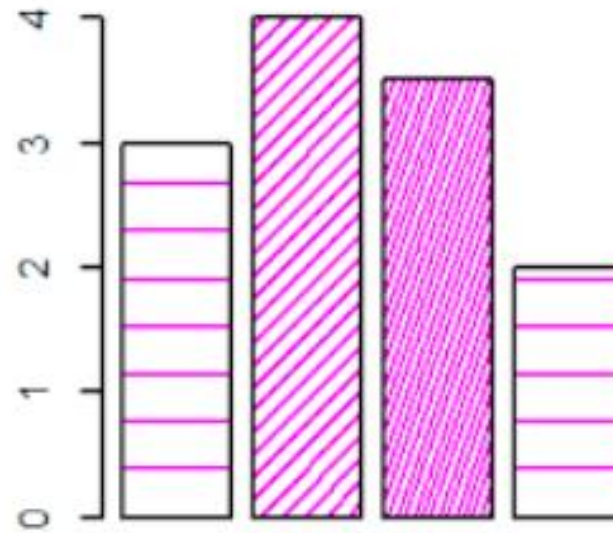
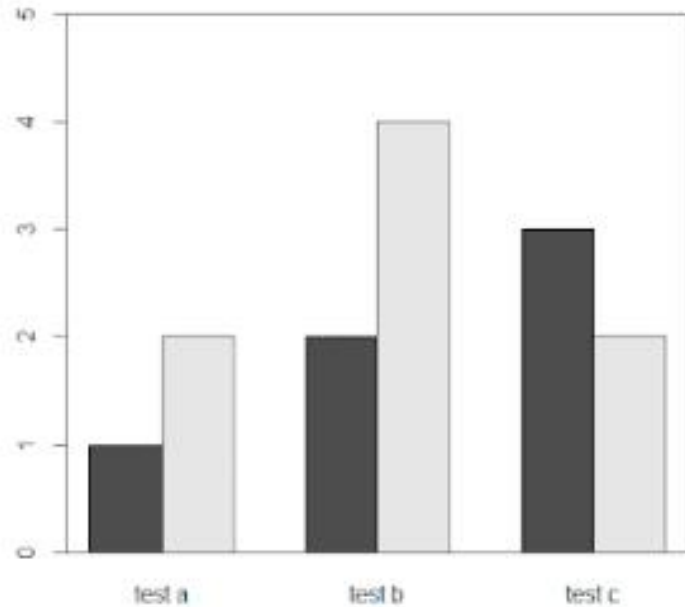
```
In [13]: data["SaleCondition"].value_counts()
```

```
Out[13]: Normal      1198  
         Partial      125  
         Abnorml     101  
         Family        20  
         Alloca        12  
         AdjLand         4  
         Name: SaleCondition, dtype: int64
```

J'effectue mon calcul sur ma colonne SaleCondition de mon dataset nommé data.

In fine je me retrouve avec 6 catégories (Normal, Partial, Abnorm, Family, Alloca, AdjLand)

2. Diagrammes en barre

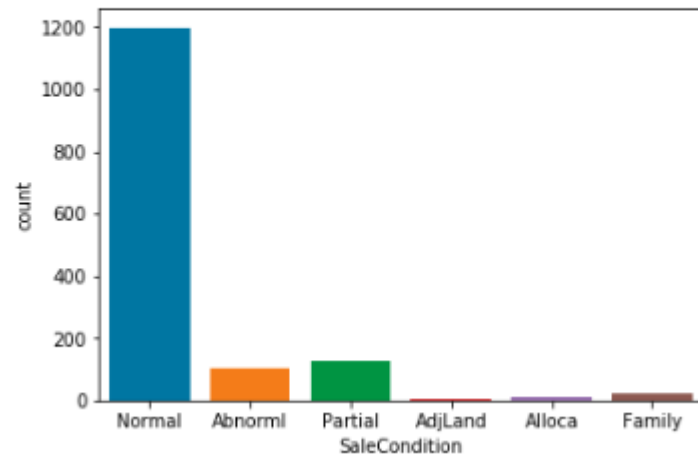


Fonctions utiles

- **countplot()** permet de visualiser à l'aide d'un diagramme en barre la repartition selon les différents facteurs d'une variable

```
In [19]: sns.countplot(data["SaleCondition"])
```

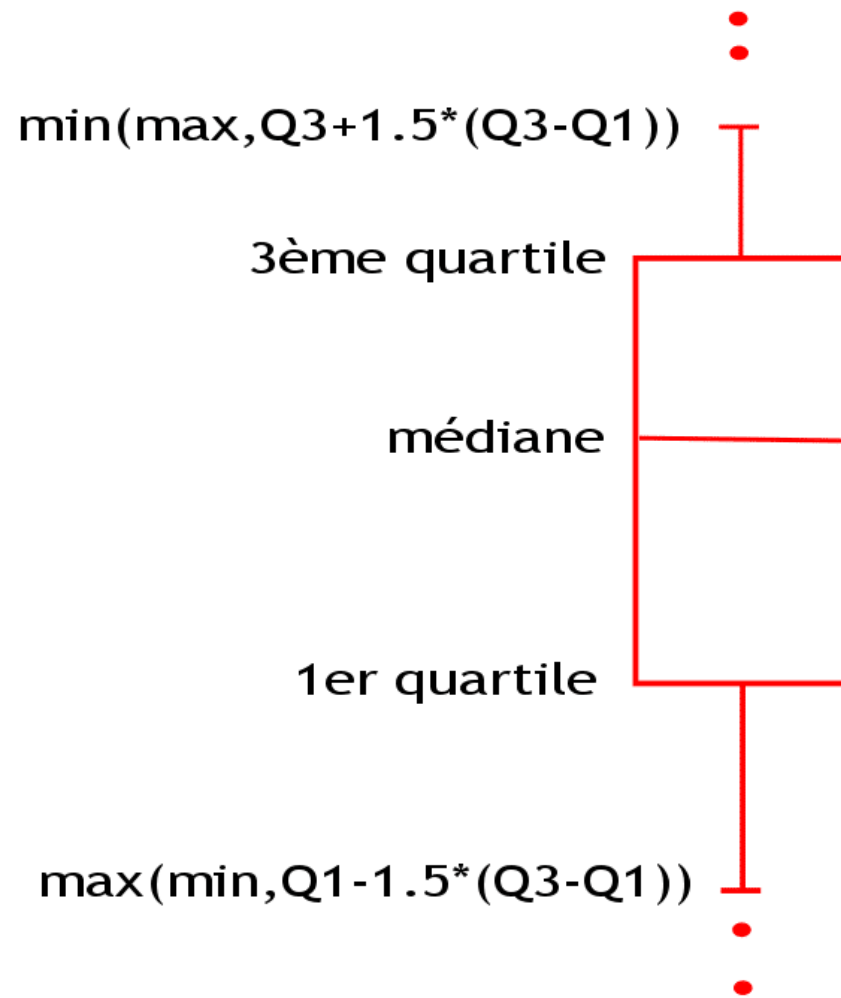
```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x17fb769ef60>
```



Il s'agit du format visuel de la table obtenue avec `valuecount()`

La fonction `countplot()` est une fonction du package `seaborn`

Boxplot – Boite à moustaches

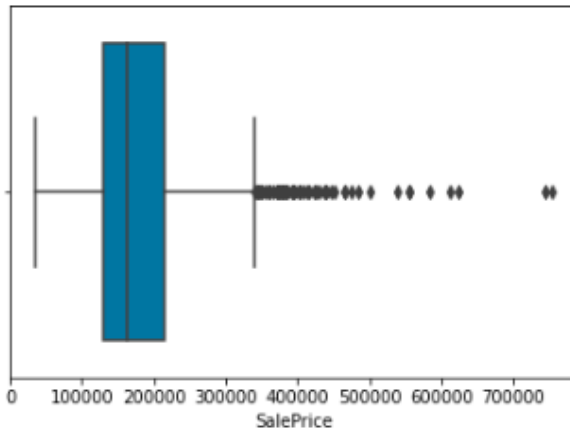


Fonctions utiles

- **boxplot()** permet de visualiser quelques indicateurs statistiques : quartiles, mediane, valeurs atypiques, ...

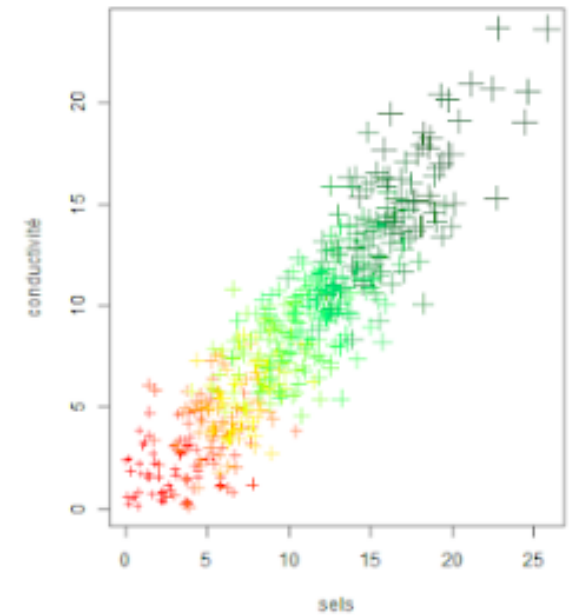
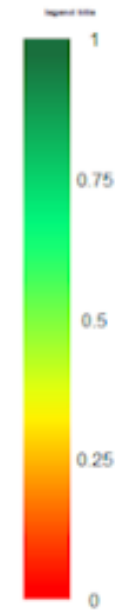
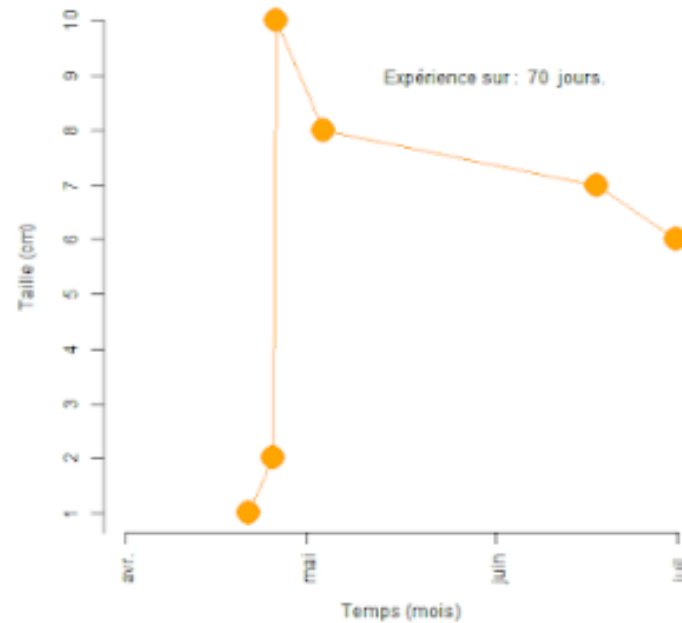
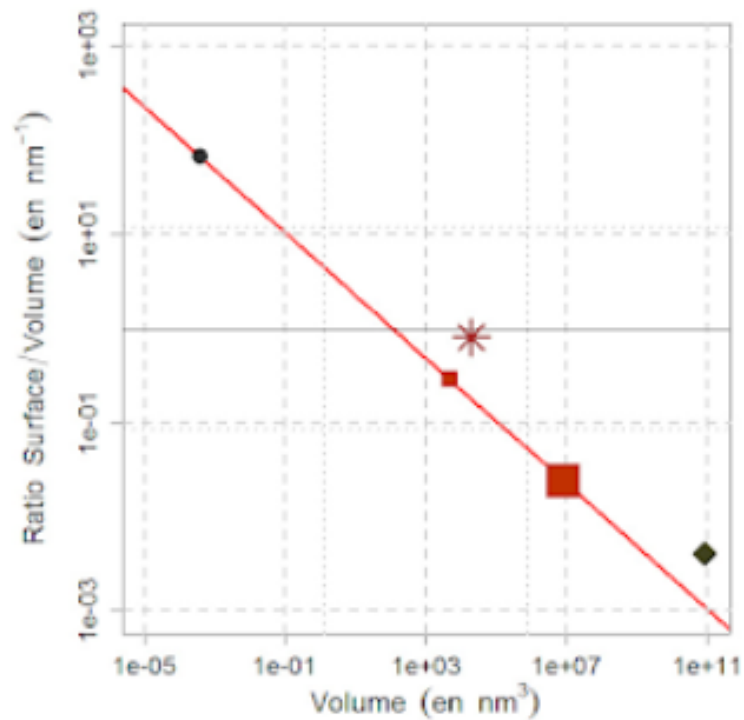
```
In [20]: sns.boxplot(data["SalePrice"])
```

```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x17fb769eef0>
```



- **Catplot()** avec le parameter kind="box" permet aussi de faire des boxplot plus poussé

1. Courbes et nuages de point

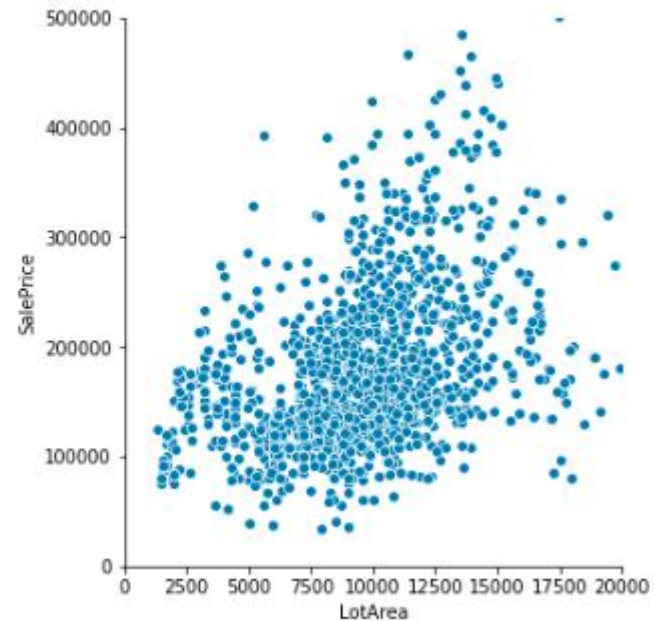


Fonctions utiles

- **relplot()** permet de visualiser un nuage de point

```
In [24]: sns.relplot(x="LotArea", y="SalePrice", data=data).set(xlim=(0,20000), ylim=(0,500000))
```

```
Out[24]: <seaborn.axisgrid.FacetGrid at 0x17fb7b8ebe0>
```



Fonctions utiles

- **relplot()** permet de visualiser une courbe à partir du moment où le paramètre `kind = "line"` est spécifié.

```
In [32]: sns.relplot(x="YrSold", y="LotArea", kind="line", data=data)
```

```
Out[32]: <seaborn.axisgrid.FacetGrid at 0x17fb8e5c4a8>
```

