



Parcours de formation

# Python pour les données en géosciences

*Adrien Garinet & Robin Guillaume-Castel*

# ATTENTION !

SIGNEZ LA FEUILLE DE PRÉSENCE !!

Disclaimer

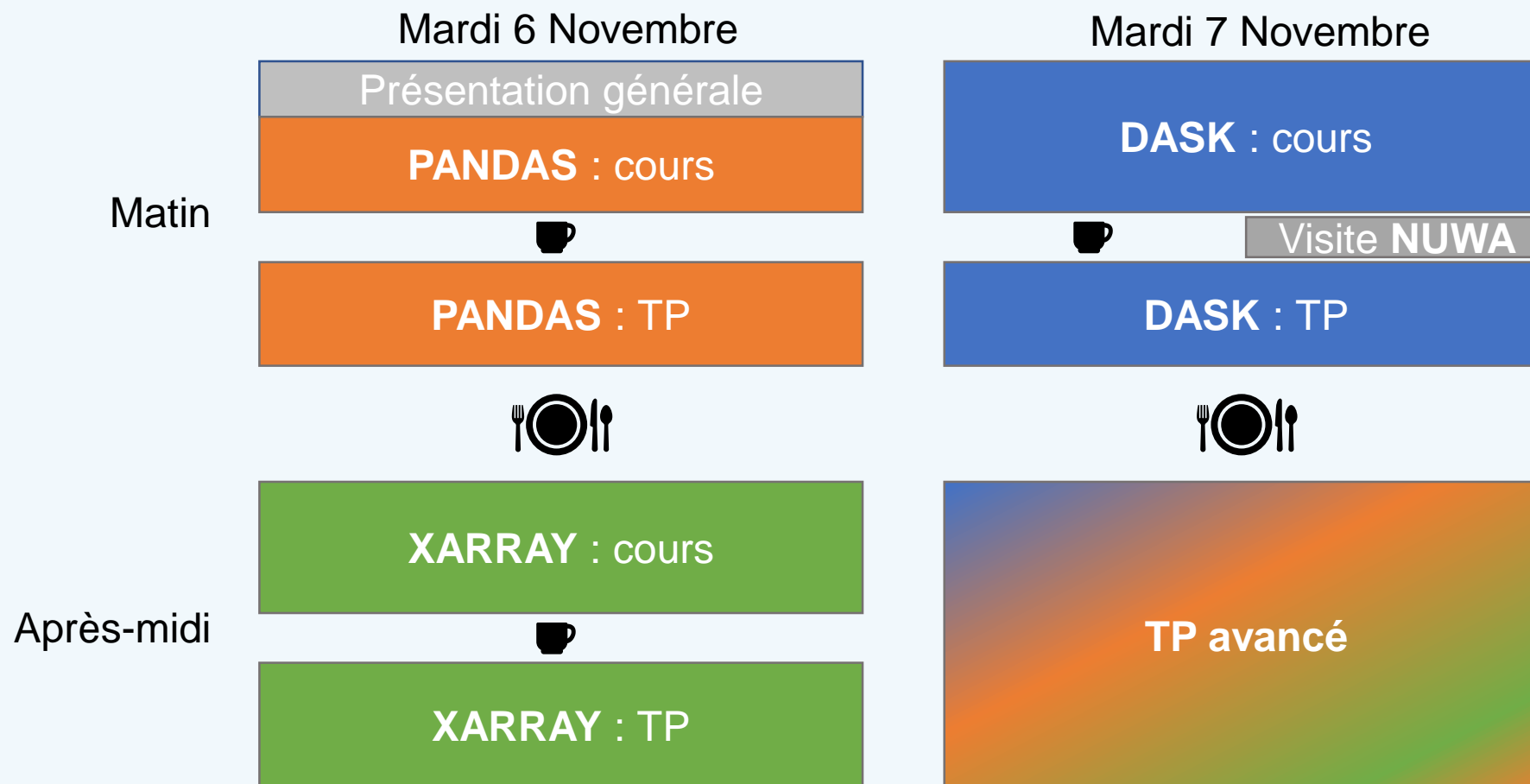
# Objectifs du parcours de formation

**Se familiariser avec les fonctions principales de pandas/xarray/dask**

**Analyser des données de géosciences de manière efficace et rapide avec python**

**Effectuer de grandes quantités de calculs avec une mémoire limitée/un CPU limité**

# Déroulement de la formation





Formation Mardi

# Analyse de données avec *pandas et xarray*

*Adrien Garinet & Robin Guillaume-Castel*

# Objectif de la journée

Connaître l'intérêt et les fonctions générales de pandas et xarray



# Présentation Générale

Pourquoi utiliser pandas et xarray?



# Retour à la base: numpy.array

- Tableaux de valeurs à n dimensions

```
array([27, 90, 55, ..., 97, 76, 87])
```

```
array([[ 5, 89, 40, ..., 78, 99, 70],  
       [ 2, 43, 13, ..., 91, 73, 34],  
       [27,  9, 91, ..., 68, 84, 62],  
       ...,  
       [44, 55, 63, ..., 65, 59, 37],  
       [20,  8, 14, ..., 94, 49, 16],  
       [45, 37, 89, ..., 53, 65, 84]])
```

- Opérations « terme à terme » vectorielles
  - array1 + array2
- Opérations statistiques
  - np.mean(array)





# Retour à la base: numpy.array

- Selection de données:
    - `array[12]`: accès par la position
    - Cas de n dimensions: `array[3, :, 4]`
  - Accès aux données pas très clair
- À quoi correspondent les données?
  - À quoi correspondent les axes?

# Principe commun 1: l'indexation

Enrobage pandas/xarray

Numpy array

Noms de variables

Indice des valeurs	Taille (cm)	Poids (kg)	Durée de vie (ans)
Elfe	200	80	1000
Nain	120	120	300
Hobbit	110	40	120

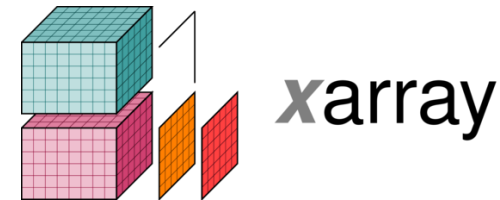
- Sélection par les indices
- Accès aux variables/dimensions par leur nom

# Principe commun 2: Un large panel de format d'entrées/sorties

- Permet de lire et écrire de nombreux types de fichiers



- csv
- excel (xls,xlsx)
- txt
- json
- ...

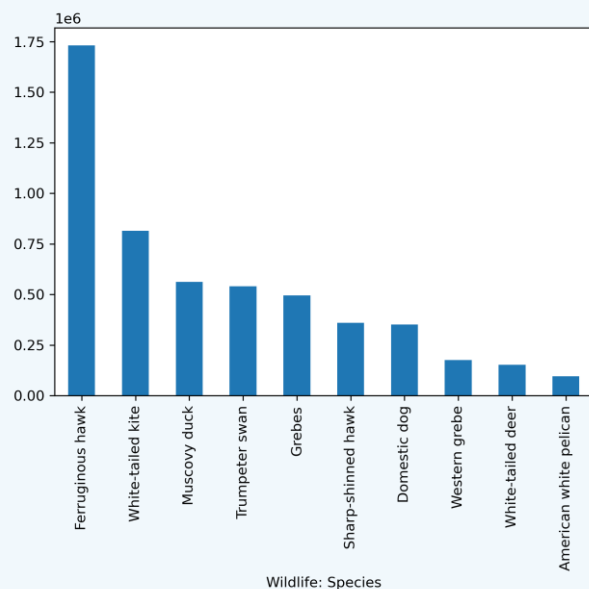


- Netcdf
- GRIB
- geoTIFF
- zarr
- ...

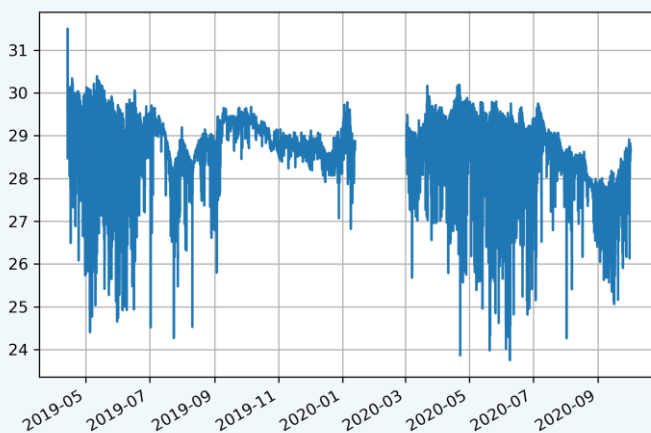
# Principe commun 3: Intégration des figures

- Intégration de matplotlib (et d'autres librairies comme hvplot, plotly, ...)

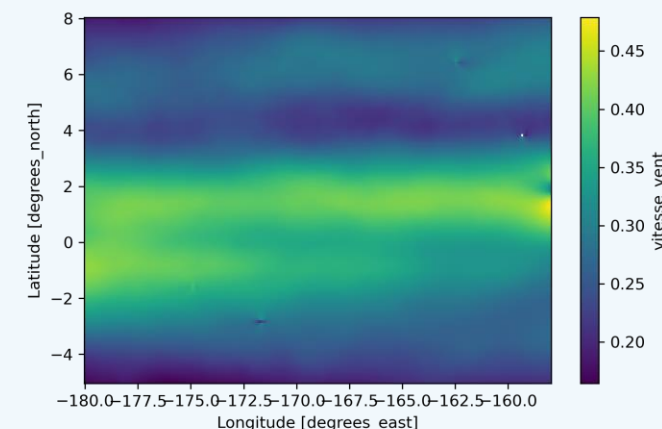
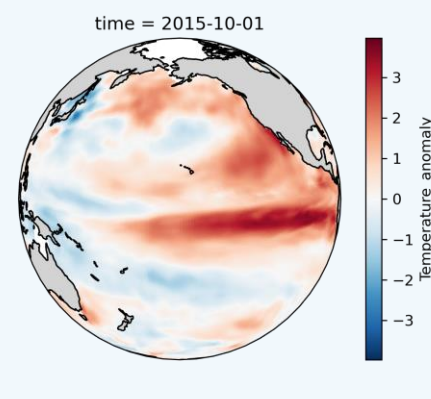
## Plots catégoriques



## Séries temporelles



## Cartes



# Quelques différences et spécificités



- Données plutôt unidimensionnelles
- Plusieurs types possibles (string, datetime, int, float, ...)
- Plus complet en termes de fonctionnalités

- Séries temporelles 1d
- Données catégoriques
- Données statistiques



- Généralisation de pandas à plusieurs dimensions
  - Parfait pour données géographiques 3d-4d
- 
- Données grillées climat/océan/atmosphère

# Quelques ressources

- <https://pandas.pydata.org/>
- <https://docs.xarray.dev/en/stable/>
- <https://gallery.pangeo.io/>
- <https://www.dask.org/>



# Des questions?

# Introduction (très) rapide à Jupyter Notebook

- Cellules
- Code
- Markdown
- Sorties

The screenshot shows a Jupyter Notebook titled "Pandas tuto" with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The notebook contains several cells:

- Entrée [4]:** A code cell with `import numpy as np` and `import matplotlib.pyplot as plt`. An arrow from "Code" points to this cell.
- Bienvenue à l'atelier numérique de l'OMP**: A markdown cell with the text "Ici je peux écrire du texte en markdown". An arrow from "Markdown" points to this cell.
- Entrée [5]:** A code cell with `x = np.arange(10)` and `y = 3*x**2 + 2`. An arrow from "Code" points to this cell.
- Entrée [6]:** A code cell with `plt.plot(x,y)`. An arrow from "Code" points to this cell.
- Out[6]:**: The output of the previous cell, showing a plot of a parabola. An arrow from "Sorties" points to this output.
- Encore un peu de markdown**: A markdown cell. An arrow from "Markdown" points to this cell.
- Entrée [7]:** A code cell with `print(x**2)`. An arrow from "Code" points to this cell.
- [ 0 1 4 9 16 25 36 49 64 81]**: The output of the previous cell. An arrow from "Sorties" points to this output.