TP6_Sujet

February 19, 2024

1 Calcul scientifique – TP6 - Traitement de données avec `numpy` et `matplotlib`

Pour commencer, on inclut (avec import) dans le bloc de code les librairies `numpy` et `matplotlib` :

[1]:
```
%matplotlib inline
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from pydoc import help
```

1.1 Analyse de données

Le tableau qui suit contient le relevé des températures de 1947 à 2018 à St-Etienne Bouthéon. Il se compose de 12 lignes (les mois) et 72 colonnes (les années).

[2]:
```

```

```
temp = [[-3.6,1.2,-0.5,-1.6,0.8,-3.2,-4.7,-2.1,2.2,0.4,-4.7,-0.3,-1.1,-1.9,-0.
↪2,1.6,-7.2,-3.1,0.1,-1.8,-1.5,-2.5,0.3,0.1,-3.9,0.2,-1.9,1.9,1.7,-1.7,0.9,-0.
↪9,-4.0,-1.7,-3.2,2.4,-0.9,-0.1,-7.3,-0.2,-5.4,3.7,-0.2,-1.4,0.5,-2.9,0.0,0.
↪9,0.9,2.9,-1.0,1.2,1.3,-0.9,2.4,-0.3,-1.0,0.9,-1.2,-1.1,2.2,1.1,-2.5,-2.3,-0.
↪3,1.4,0.1,3.4,0.6,2.8,-3.4,3.4],[-1.8,-1.7,-3.3,2.1,1.0,-2.6,-2.7,-2.9,-0.
↪3,-12.8,3.5,1.5,-2.0,0.5,3.2,-2.4,-4.7,1.8,-4.2,4.2,0.5,1.1,-2.6,0.4,-0.9,2.
↪0,-1.7,0.8,-0.9,0.7,3.4,0.4,1.2,2.4,-3.4,0.6,-2.4,-1.7,-0.5,-3.5,1.0,0.1,0.
↪2,4.6,-1.9,-0.7,-3.1,2.2,3.5,-0.3,2.2,0.2,-0.1,2.2,1.6,3.6,-0.6,-0.7,-2.5,-0.
↪8,3.6,0.7,-0.6,0.1,0.0,-5.5,-2.3,3.1,-0.9,2.3,2.4,-2.2],[3.7,3.0,-1.0,1.3,1.
↪5,3.9,-0.2,2.1,-1.0,0.9,3.0,0.1,3.6,4.0,-0.1,-1.3,2.0,2.5,1.2,-0.7,3.0,1.2,2.
↪5,-0.5,-3.2,1.4,-1.7,2.9,1.2,-1.1,3.4,2.0,3.4,1.3,5.6,0.4,1.9,-0.2,0.9,0.9,1.
↪0,2.3,3.7,2.4,5.5,3.1,0.5,4.7,0.9,1.0,2.9,2.6,3.4,1.7,6.5,3.6,2.5,1.6,2.0,2.
↪8,2.5,2.0,1.1,1.5,2.9,2.7,2.6,2.6,2.8,1.4,4.3,2.2],[5.0,5.0,6.2,3.0,3.8,5.
↪3,4.3,3.0,1.5,2.6,3.4,2.5,5.2,3.2,7.1,3.1,4.9,4.6,3.3,6.0,1.9,3.6,4.0,2.2,4.
↪1,3.1,0.2,2.2,3.2,2.1,3.4,2.4,2.9,2.9,4.4,1.7,4.6,2.0,4.1,2.6,5.0,5.6,4.5,3.
↪2,2.9,4.8,6.2,4.6,5.1,4.7,2.2,5.1,5.3,5.8,4.8,4.4,5.3,5.1,5.2,4.7,7.2,4.9,6.
↪6,5.1,6.5,5.4,5.5,5.5,5.3,5.9,2.4,6.5],[9.2,8.6,6.1,8.3,6.8,6.4,7.8,6.1,6.
↪4,7.4,4.9,9.3,7.2,7.8,6.6,6.0,6.0,8.7,7.0,8.1,7.0,6.8,7.4,6.2,8.1,6.0,7.4,6.
↪0,6.1,6.8,7.5,7.2,7.0,6.2,8.1,7.3,7.4,5.5,7.0,9.7,6.1,9.9,8.8,9.7,5.6,9.0,9.
↪1,9.4,8.2,8.9,8.8,8.5,11.4,10.4,10.5,8.0,10.1,7.2,9.3,9.1,10.7,10.3,10.7,7.
↪8,10.2,9.7,7.0,8.1,9.5,8.8,8.5,9.7],[11.6,11.0,9.1,12.4,11.2,11.3,10.4,10.
↪0,10.9,9.0,11.0,10.2,10.9,11.4,11.5,7.8,11.4,11.0,11.3,10.7,8.6,9.6,9.2,11.
↪6,10.1,8.1,10.1,8.8,9.4,10.5,10.0,10.0,12.4,10.1,10.0,12.8,12.0,10.0,9.6,11.
↪9,11.4,10.9,9.2,11.5,11.5,11.6,12.7,12.4,10.4,12.3,12.4,12.3,10.7,12.6,11.
↪2,13.7,16.9,12.9,13.6,12.9,13.0,13.0,12.4,12.5,12.7,14.0,11.4,12.5,13.6,13.
↪3,14.4,12.4],[14.1,11.3,11.6,13.8,12.1,12.9,12.2,10.3,12.7,12.1,12.4,12.7,14.
↪0,11.0,11.4,11.1,13.3,12.2,12.2,11.1,13.7,11.6,12.5,11.8,12.3,10.8,11.8,10.
↪6,11.8,13.4,12.2,12.1,11.8,11.5,12.3,14.4,16.1,11.9,13.5,13.3,14.6,12.9,14.
↪2,13.8,14.9,14.6,12.8,16.1,15.7,12.8,13.5,14.3,14.7,12.3,14.6,13.8,15.4,13.
↪8,15.3,17.6,13.9,13.2,14.9,15.6,12.5,13.4,15.2,14.2,16.5,14.6,14.8,14.3],[14.
↪1,13.0,12.5,12.8,12.8,13.2,12.1,11.5,11.5,11.4,11.0,13.3,11.9,12.3,11.3,11.
↪9,11.5,11.9,11.5,11.6,12.4,11.4,11.7,12.8,13.2,11.0,13.3,12.1,13.9,10.8,11.
↪1,10.7,11.7,13.0,12.0,12.9,13.7,12.0,11.6,12.7,13.4,13.5,13.1,14.0,14.5,15.
↪1,13.1,15.2,13.9,13.0,16.5,13.1,14.7,14.7,14.7,14.0,17.8,15.1,12.4,11.9,13.
↪3,13.4,14.7,13.6,14.2,14.9,13.3,13.1,15.1,14.0,13.9,14.4],[10.6,9.0,14.0,9.
↪9,10.6,8.1,10.6,10.7,9.4,11.6,9.2,11.7,12.1,9.5,13.0,9.5,9.9,11.2,9.3,10.2,9.
↪7,9.4,10.1,9.3,7.8,5.1,8.8,10.2,11.2,9.3,7.4,8.3,10.1,10.6,11.4,11.9,11.5,9.
↪3,10.0,10.7,13.4,10.6,10.1,9.3,12.7,10.2,10.0,11.5,9.1,6.9,11.0,11.3,13.3,11.
↪3,8.8,10.7,10.6,11.6,11.5,14.1,9.1,8.8,11.0,9.4,12.3,11.2,11.4,11.5,9.3,12.
↪5,8.5,10.1],[6.2,4.6,8.5,6.6,6.2,6.4,7.5,6.3,3.5,4.8,3.4,5.2,6.1,7.1,7.6,5.
↪5,5.5,5.2,7.7,9.5,9.3,8.0,5.9,5.5,4.5,3.8,3.9,3.1,4.6,7.3,8.7,4.8,9.1,5.5,7.
↪5,7.0,6.0,7.2,5.9,9.1,9.6,8.9,7.2,9.2,6.0,7.0,7.4,8.3,10.9,7.2,7.1,7.5,8.8,8.
↪0,10.4,7.9,6.9,10.3,11.1,11.2,6.5,7.2,6.1,6.8,8.0,8.9,10.8,10.2,6.4,6.0,6.
↪1,6.8],[3.9,0.9,2.0,4.2,4.6,2.0,1.1,3.9,1.0,0.1,2.3,2.2,2.2,4.2,2.2,1.1,5.
↪2,3.3,3.5,0.9,2.6,2.8,1.6,3.5,0.5,2.3,0.0,3.5,2.5,1.6,2.4,-0.1,1.4,0.5,0.2,3.
↪8,2.1,6.1,-0.6,4.3,3.6,1.2,1.9,4.0,3.5,4.8,1.5,6.6,2.5,2.5,5.0,0.0,1.9,5.2,0.
↪9,5.9,4.4,3.5,2.4,5.5,1.9,3.7,6.3,4.5,5.4,4.5,2.7,6.3,4.0,4.1,1.6,4.4],[-0.
↪3,0.1,1.4,-3.1,-0.2,-0.2,4.4,1.4,1.9,1.6,-1.7,0.7,2.7,-0.5,0.0,-4.7,-5.2,-1.
↪6,1.9,1.1,-2.4,-1.1,-4.5,-2.7,-0.7,-1.2,-1.5,2.7,-1.8,-0.5,0.8,2.4,0.5,-3.
```

- Créer, à partir due la liste python `temp` défini ci-dessus, une variable numpy de type `array`, nommée `t` contenant les valeurs du tableau, puis afficher les dimensions de `t` pour vérifier que vous avez bien 12 lignes et 72 colonnes.

[3]:
```
# A faire ...
```

```
(12, 72)
```

1.1.1 Extraction et affichage des statistiques simples

Remarque: pour cette partie, il n'y a aucune boucle à écrire.

- Quelle est l'année pour laquelle la température de janvier a été la plus froide ? Quelle était cette température ?

[4]:
```
# A faire ...
```

```
Température la plus froide observée en janvier :  -7.3
C'était en :  1985
```

- Calculer la température moyenne sur toutes les années et tous les mois. Afficher la valeur approchée avec 1 chiffre significatif.

[5]:
```
# A faire ...
```

```
Température moyenne :  6.0
```

- Calculer le tableau `ta` contenant les valeurs moyennes des températures de chaque année.
- Calculer le tableau `tm`contenant les temparatures moyenne de chaque mois, sur l'ensemble des années.
- Afficher ces deux tableaux (approximation à 1 chiffre significatif).

[6]:
```
# A faire ...
```

```
Temp. annuelles :  [6. 6. 6. 6. 6. 5. 5. 5. 5. 4. 5. 6. 6. 6. 6. 4. 4. 6. 5. 6.
5. 5. 5. 5.
 4. 4. 4. 5. 5. 5. 6. 5. 6. 5. 5. 6. 6. 5. 5. 6. 6. 7. 6. 7. 6. 7. 6. 8.
 7. 6. 7. 6. 7. 7. 7. 8. 7. 7. 6. 7. 7. 7. 7. 6. 7. 7. 6. 8. 7. 7. 6. 7.]
Temp. mensuelles :  [-1. -0.  2.  4.  8. 11. 13. 13. 10.  7.  3.  0.]
```

- Calculer la température minimale (`tmin`), maximale (`tmax`) et médiane (`tmed`) pour chaque mois de l'année.

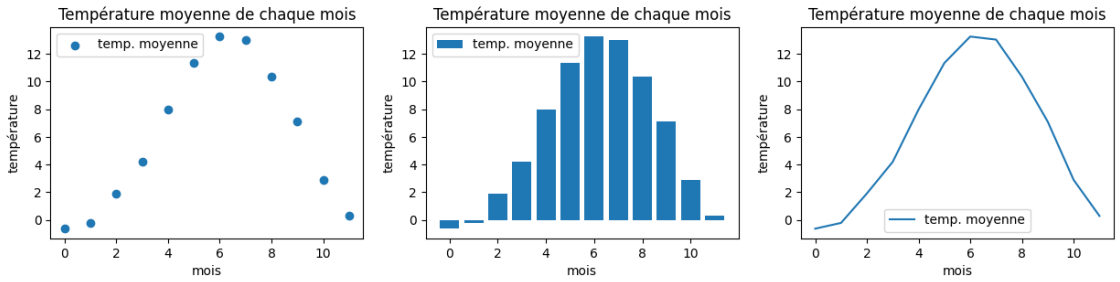
[7]:
```
# A faire ...
```

```
tmin =  [ -7.3 -12.8  -3.2   0.2   4.9   7.8  10.3  10.7   5.1   3.1  -0.6
-5.2]
tmax =  [ 3.7  4.6  6.5  7.2 11.4 16.9 17.6 17.8 14.1 11.2  6.6  4.9]
tmed =  [-0.3   0.05  2.    4.55  7.9  11.4  13.25 13.05 10.2   7.05  2.55  0.4
]
```

- Avec matplotlib, afficher la température moyenne des 12 mois de l'année: on fera une figure contenant 3 sous-figures représentant ces températures de 3 manières différentes (scatter, bar et plot), à l'image de la figure qui suit. On fera attention à la place de la légende.

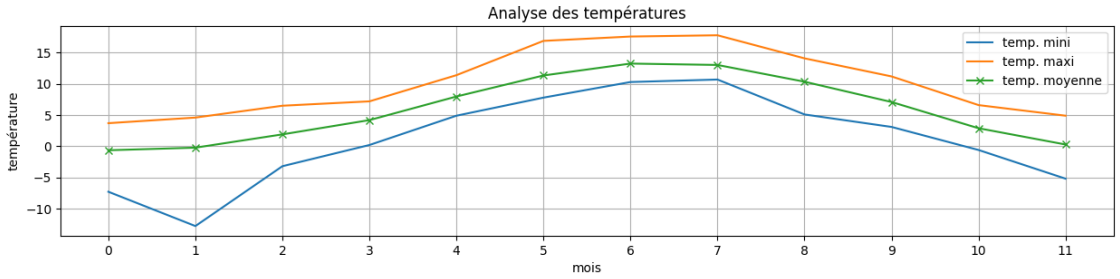
[8]:
```
# A faire ...
```

[8]:
```
<matplotlib.legend.Legend at 0x7f99adad3b20>
```



- Faire un graphique où figurent les températures moyennes, min et max comme sur le shéma ci-dessous (avec légende et ticks)

[22]:
```
# A faire ...
```



1.1.2 Calcul d'histogrammes

- Calculer l'histogramme des température sur l'ensemble de la période considérée au moyen de la fonction `histogram`de `numpy`.

Regarder dans la documentation (avec help dans la cellule python) quelles sont les paramètres de cette fonction et ce qu'elle retourne.

On souhaite calculer la probabilité (c'est à dire avec density=True) plutôt que la fréquence (density=False) et on choisira bins=20 (nombre+1 de bins également espacés).

[10]:
```
# A faire ...
```

```
Help on function histogram in module numpy:
```

```
histogram(a, bins=10, range=None, density=None, weights=None)
    Compute the histogram of a dataset.

    Parameters
    ----------
    a : array_like
        Input data. The histogram is computed over the flattened array.
    bins : int or sequence of scalars or str, optional
        If `bins` is an int, it defines the number of equal-width
        bins in the given range (10, by default). If `bins` is a
        sequence, it defines a monotonically increasing array of bin edges,
        including the rightmost edge, allowing for non-uniform bin widths.

        .. versionadded:: 1.11.0

        If `bins` is a string, it defines the method used to calculate the
        optimal bin width, as defined by `histogram_bin_edges`.

    range : (float, float), optional
        The lower and upper range of the bins.  If not provided, range
        is simply ``(a.min(), a.max())``.  Values outside the range are
        ignored. The first element of the range must be less than or
        equal to the second. `range` affects the automatic bin
        computation as well. While bin width is computed to be optimal
        based on the actual data within `range`, the bin count will fill
        the entire range including portions containing no data.
    weights : array_like, optional
        An array of weights, of the same shape as `a`.  Each value in
        `a` only contributes its associated weight towards the bin count
        (instead of 1). If `density` is True, the weights are
        normalized, so that the integral of the density over the range
        remains 1.
    density : bool, optional
        If ``False``, the result will contain the number of samples in
        each bin. If ``True``, the result is the value of the
        probability *density* function at the bin, normalized such that
        the *integral* over the range is 1. Note that the sum of the
        histogram values will not be equal to 1 unless bins of unity
        width are chosen; it is not a probability *mass* function.

    Returns
    -------
    hist : array
        The values of the histogram. See `density` and `weights` for a
        description of the possible semantics.
    bin_edges : array of dtype float
        Return the bin edges ``(length(hist)+1)``.
```

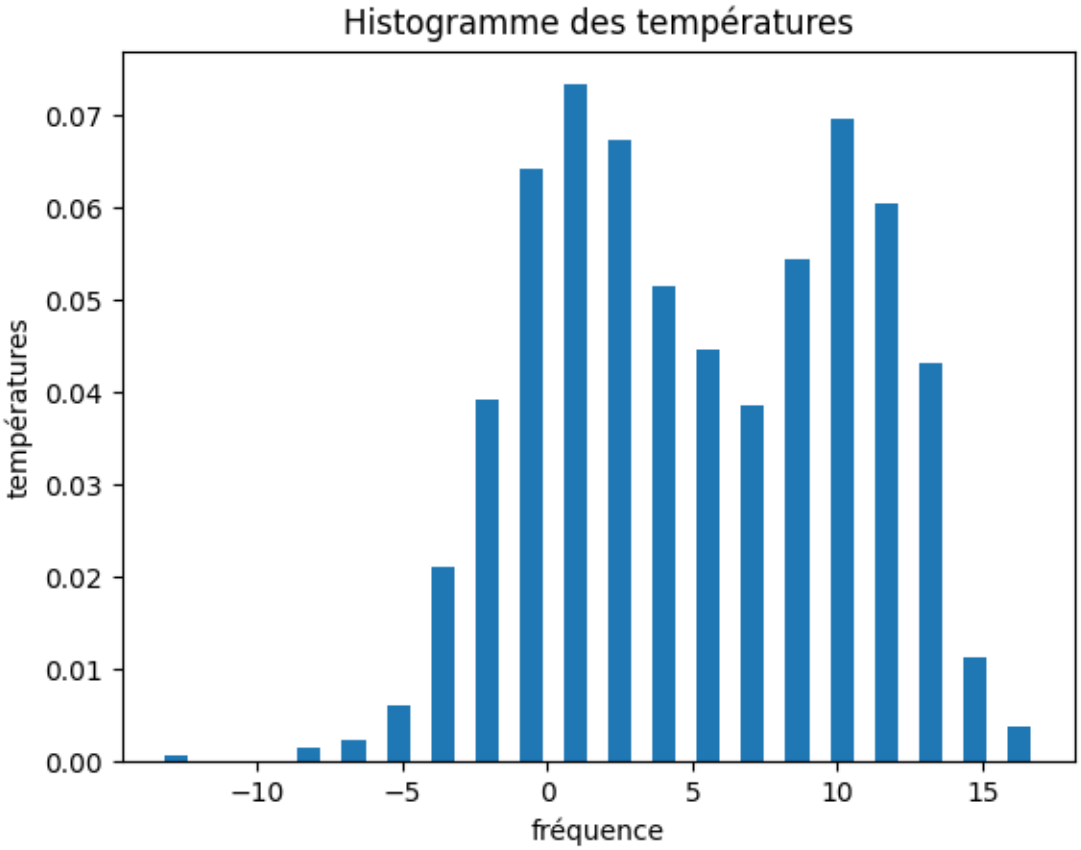
```
See Also
--------
histogramdd, bincount, searchsorted, digitize, histogram_bin_edges

Notes
-----
All but the last (righthand-most) bin is half-open.  In other words,
if `bins` is::

  [1, 2, 3, 4]

then the first bin is ``[1, 2)`` (including 1, but excluding 2) and
the second ``[2, 3)``.  The last bin, however, is ``[3, 4]``, which
*includes* 4.


Examples
--------
>>> np.histogram([1, 2, 1], bins=[0, 1, 2, 3])
(array([0, 2, 1]), array([0, 1, 2, 3]))
>>> np.histogram(np.arange(4), bins=np.arange(5), density=True)
(array([0.25, 0.25, 0.25, 0.25]), array([0, 1, 2, 3, 4]))
>>> np.histogram([[1, 2, 1], [1, 0, 1]], bins=[0,1,2,3])
(array([1, 4, 1]), array([0, 1, 2, 3]))

>>> a = np.arange(5)
>>> hist, bin_edges = np.histogram(a, density=True)
>>> hist
array([0.5, 0. , 0.5, 0. , 0. , 0.5, 0. , 0.5, 0. , 0.5])
>>> hist.sum()
2.4999999999999996
>>> np.sum(hist * np.diff(bin_edges))
1.0

.. versionadded:: 1.11.0

Automated Bin Selection Methods example, using 2 peak random data
with 2000 points:

>>> import matplotlib.pyplot as plt
>>> rng = np.random.RandomState(10)  # deterministic random data
>>> a = np.hstack((rng.normal(size=1000),
…                rng.normal(loc=5, scale=2, size=1000)))
>>> _ = plt.hist(a, bins='auto')  # arguments are passed to np.histogram
>>> plt.title("Histogram with 'auto' bins")
Text(0.5, 1.0, "Histogram with 'auto' bins")
>>> plt.show()
```

[11]:
```
# A faire ...
```

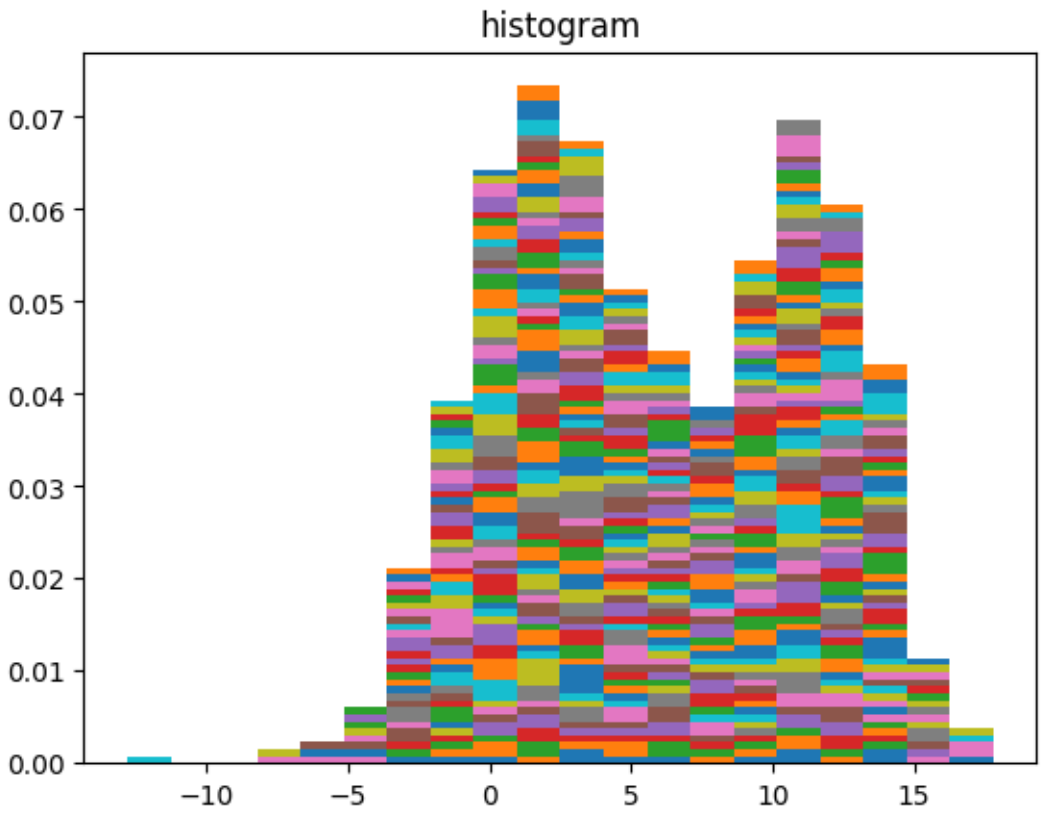
```
[0.00075648 0.         0.         0.00151295 0.00226943 0.0060518
 0.02118131 0.03933672 0.06430041 0.07337812 0.06732631 0.05144033
 0.04463205 0.03858025 0.05446623 0.06959574 0.06051803 0.0431191
 0.01134713 0.00378238]
[-12.8  -11.27  -9.74  -8.21  -6.68  -5.15  -3.62  -2.09  -0.56   0.97
   2.5    4.03   5.56   7.09   8.62  10.15  11.68  13.21  14.74  16.27
  17.8 ]
```

- Afficher cet histogramme avec `matplotlib`.

[12]:
```
# A faire ...
```



[13]:
```
# A faire ...
```



1.1.3 Mesures de dispersion des températures.

Nous souhaitons désormais mesurer la dispersion des températures, c'est-à-dire regarder comment les valeurs se répartissent autour de la moyenne. Les valeurs min et max ne donnent pas une bonne indication de la dispersion. La variance et l'écart type sont liés à la manière dont les points diffèrent de la moyenne. Plus la variance est grande, plus la dispersion est forte. La variance se calcule avec la formule :

$$variance = \frac{\sum_i (x_i - x_{moy})^2}{n}$$

L'écart-type est la racine carré de la variance :

$$ect = \sqrt{variance}$$

- On commence par calculer la variance et l'écart type des températures de chaque mois de l'année. Il y aura donc 12 variances/écarts types. On fera les calculs en utilisant les opérateurs de base de numpy (np.sum(),-,/,**2), sans faire de boucle.
- Afficher les résultats obtenus.

[14]:
```
# A faire ...
```

```
Ecart type des températures de chaque mois [2.33653055 2.71891487 1.74749324
1.4845806  1.50777819 1.59295323
 1.54530289 1.38848814 1.59347568 1.91948263 1.75914717 2.02943188]
```

- Utiliser ensuite les fonctions dédiées de numpy (var et std) et vérifier que les résultats sont identiques.

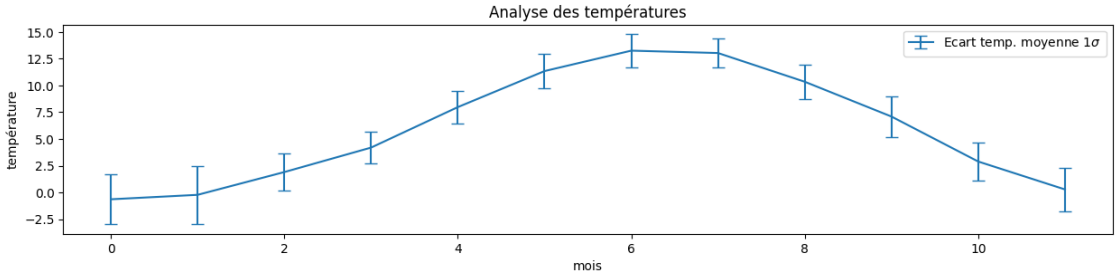
[15]:
```
# A faire ...
```

```
Ecart type des températures de chaque mois [2.33653055 2.71891487 1.74749324
1.4845806  1.50777819 1.59295323
 1.54530289 1.38848814 1.59347568 1.91948263 1.75914717 2.02943188]
```

- Tracer un graphique contenant les températures mensuelles ainsi que des barres d'erreurs indiquant la dispersion à $[-\sigma, \sigma]$ autour des moyennes où σ désigne l'écart-type.

[16]:
```
# A faire ...
```

[16]:
```
<matplotlib.legend.Legend at 0x7f99aa96ff40>
```



- Faire le même graphique donnant la dispersion des températures pour chaque année.

[17]:
```
# A faire ...
```

[17]:
```
<matplotlib.legend.Legend at 0x7f99aa7bde50>
```