NSI : TP 3, Statistiques avec des tuples.

1. Déterminer différents indicateurs statistiques d'une série de données quantitatives.

En Python, on utilise souvent les tuples pour stocker des données que l'on ne veut pas modifier. On retrouve sur les tuples les méthodes propres aux chaînes de caractères :

```
>>> a=(1,2,3,4,2,2,1)
>>> len(a) # nombre d'éléments du tuple
7
>>> a.count(2) # nombre d'apparition d'un élément dans un tuple
3
>>> a.count(12)
0
>>> for i in a : # parcourir un tuple
    print(i)
1
2
3
4
2
2
1
>>> for i in range(len(a)): # parcourir un tuple
    print(a[i])
1
2
3
4
2
2
1
>>> min(a)
1
>>> max(a)
4
```

a) Données sans effectifs.

On considère par exemple un relevé de données numériques :

```
a=(2,1,3,6,1,2,8,10,3,1,1,3,4,8,9,10)
```

Exercice 1

Compléter les fonctions suivantes pour qu'elles respectent leurs docstring :

```
def effectif_total(donnees):
    '''
    retourne le nombre de valeurs contenue dans donnees
    donnees : tuple ne contenant que des valeurs de type int ou float
    return : int

    >>> a=(1,2,5)
    >>> effectif_total(a)
    >>> 3

    >>> b=(2,1,3,6,1,2,8,10,3,1,1,3,4,8,9,10)
    >>> effectif_total(b)
    16
    '''


def somme(donnees):
    '''
    retourne la somme des valeurs contenues dans donnee
    : donnees : tuple ne contenant que des valeurs de type int ou float
    : return : int ou float

    >>> a=(1,2,5)
    >>> somme(a)
    >>> 8

    >>> b=(2,1,3,6,1,2,8,10,3,1,1,3,4,8,9,10)
    >>> somme(b)
    72
    '''


def moyenne(donnees):
    '''
    retourne la moyenne des valeurs contenues dans donnees
    : donnee : tuple ne contenant que des valeurs de type int ou float
    : return : moyenne des donnees , float

    >>> a=(1,2,5)
    >>> moyenne(a)
    >>> 2.0

    >>> b=(2,1,3,6,1,2,8,10,3,1,1,3,4,8,9,10)
    >>> moyenne(b)
```

```
    4.5
    '''


def val_max(donnees):
    '''
    retourne la valeur maximale du tuple donnees
    : return : int ou float

    >>> a=(1,2,5,0)
    >>> val_max(a)
    5
    >>> a=(2,1,3,6,1,2,8,10,3,1,1,3,4,8,9,10)
    >>> val_max(a)
    10
    '''


def val_min(donnees):
    '''
    retourne la valeur min du tuple donnees
    : return : int ou float

    >>> a=(1,2,5,0)
    >>> val_min(a)
    0
    >>> b=(2,1,3,6,1,2,8,10,3,1,1,3,4,8,9,10)
    >>> val_min(b)
    1
    '''


def etendue(donnees):
    '''
    retourne l'étendue du tuple donnees
    : return : int ou float

    >>> a=(1,2,5,0)
    >>> etendue(a)
    5
    >>> b=(2,1,3,6,1,2,8,10,3,1,1,3,4,8,9,10)
    >>> etendue(b)
    9
    '''


def effectif_valeur(donnees,valeur):
    '''
    retourne l'effectif de la valeur dans la série donnees
```

```
    : return : effectif de la valeur dans donnees , type int :

    >>> a=(1,2,5,0)
    >>> effectif_valeur(a,1)
    1
    >>> effectif_valeur(a,8)
    0
    >>> b=(2,1,3,6,1,2,8,10,3,1,1,3,4,8,9,10)
    >>> effectif_valeur(b,10)
    2
    '''

def frequence_valeur(donnees,valeur):
  '''
    retourne la fréquence de la valeur dans la série donnees
    : return : effectif de la valeur dans donnees , type float :

    >>> a=(1,2,5,0)
    >>> frequence_valeur(a,1)
    0.25
    >>> frequence_valeur(a,8)
    0.0
    >>> b=(2,1,3,6,1,2,8,10,3,1,1,3,4,8,9,10)
    >>> frequence_valeur(b,10)
    0.125
    '''
```

2. Mini-projet : exploitation de données météorologiques.

Partie A.

On considère le programme suivant, qui ne contient pour l'instant que des données :

```
donnees=(
    ("Lille",
     (2.8,3.4,6.2,9.3,12.6,15.4,17.1,17.3,15.2,11,6.6,3.6),
     (52,44,49,42,54,60,62,60,60,63,69,58)),
    ("Turin",
     (1.4,3.6,8.3,12.6,17.1,20.7,23.6,22.4,18.8,13,6.9,2.9),
     (38,52,71,97,108,89,55,70,68,86,71,41)),
    ("Moscou",
     (-9.2,-8,-2.5,5.9,12.8,16.8,18.4,16.6,11.2,4.9,-1.5,-6.2),
     (43,35,33,42,49,78,89,76,63,61,57,53)),
    ("Madrid",
     (5,6.4,9.6,12.2,15.8,20.4,24,23.2,19.6,14,8.9,5.4),
```

```
     (43,44,35,45,44,28,11,11,30,51,58,50)),
    ("Alméria",
     (11.8,12.2,14.1,15.8,18.5,21.8,24.8,25.3,23.3,19.3,15.4,12.7),
     (30,22,22,24,18,9,1,2,13,31,27,29)),
    ("Berlin",
     (-0.9,0,3.9,8.6,13.5,16.8,18.6,18,14.4,10.4,4.4,1),
     (43,34,35,41,54,70,57,61,44,37,45,49))
    )
# donnees sous la forme (
#(nom de la villle, relevé température moyenne par mois, relevé cumul des
précipitations en mm par mois)
# )
```

Les données sont issues du site : https://fr.climate-data.org/info/sources/

En utilisant les fonctions définies à la partie 1, définir les trois fonctions répondant aux docstring suivantes :

```
def ville_dans_liste(ville):
    '''
    retourne False si la ville n'est pas dans données, sinon retourne
    l'indice du tuple de données contenant ville
    : ville : str
    : return : bool ou int

    >>>ville_dans_liste("Lille")
    0
    >>>ville_dans_liste("Turin")
    1
    >>>ville_dans_liste("Marseille")
    False
    '''

def temp_moyenne(ville):
    '''
    retourne la température moyenne annuelle de la ville
    arrondie à 1 chiffre après la virgule
    : ville : str
    : return : float

    >>>temp_moyenne("Lille")
    10.0
    >>>temp_moyenne("Marseille")
    'Désolé, nous n'avons pas les données de Marseille'
    '''

def cumul_precipitation(ville):
    '''
    retourne le cumul des précipitations sur l'année
```

```
    arrondi à 1 chiffre après la virgule
    : ville : str
    : return : float

    >>>cumul_precipitation("Lille")
    673
    >>>cumul_precipitation("Marseille")
    'Désolé, nous n'avons pas les données de Marseille'
    '''
```

Partie B

En vous servant du site :

https://www.infoclimat.fr/climatologie/annee/2018/lille-lesquin/valeurs/07015.html

- ajouter aux données, les données concernant Lille_2018 pour l'année 2018.
- comparer les températures moyennes et le total des précipitation des villes Lille et Lille_2018.