# Evolution météorologique en France et dans le monde

Caio Iglesias, Luca Mainini, Marc Argaud, Amine Louzar, Sandra Nihama, Antoine Naccache

https://gitlab-cw8.centralesupelec.fr/2019nihamas/groupe5_3

# Description globale du produit

Pourquoi? Sensibilisation par rapport au réchauffement climatique

Pour qui? Grand publique

# Ce qu’on a fait

01

Visualisation de l’évolution du climat en France

02

Visualisation de l’évolution du climat dans le monde

03

Visualisation de l’évolution de la concentration en CO2

04

Corrélation entre CO2 et température

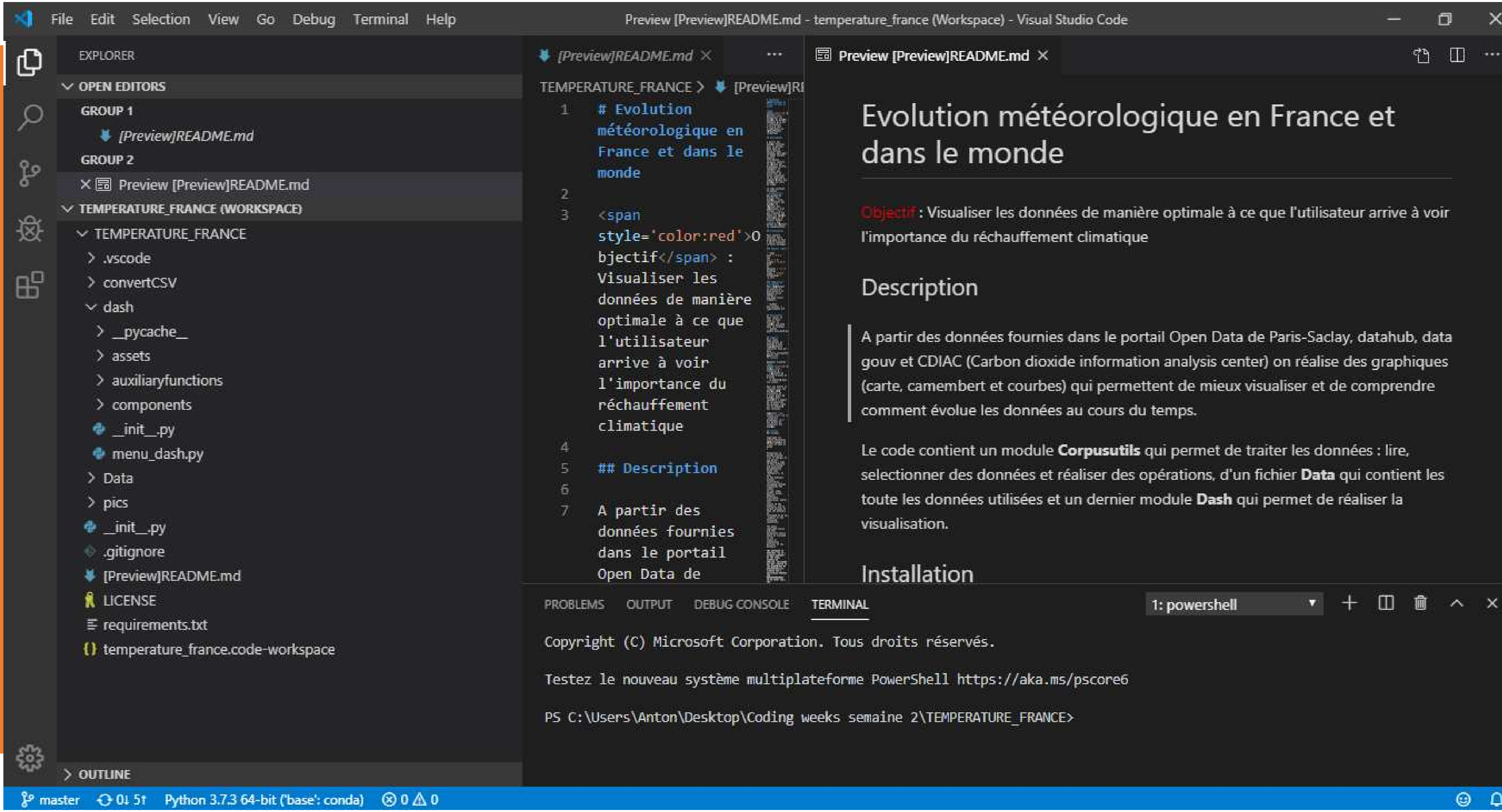
# Le Read me

# Structure du code

Data

Convert CSV

Dash



```
# Evolution météorologique en France et dans le monde

<span style='color:red'>Objectif</span> : Visualiser les données de manière optimale à ce que l'utilisateur arrive à voir l'importance du réchauffement climatique

## Description

A partir des données fournies dans le portail Open Data de
```

# Evolution météorologique en France et dans le monde

Objectif : Visualiser les données de manière optimale à ce que l'utilisateur arrive à voir l'importance du réchauffement climatique

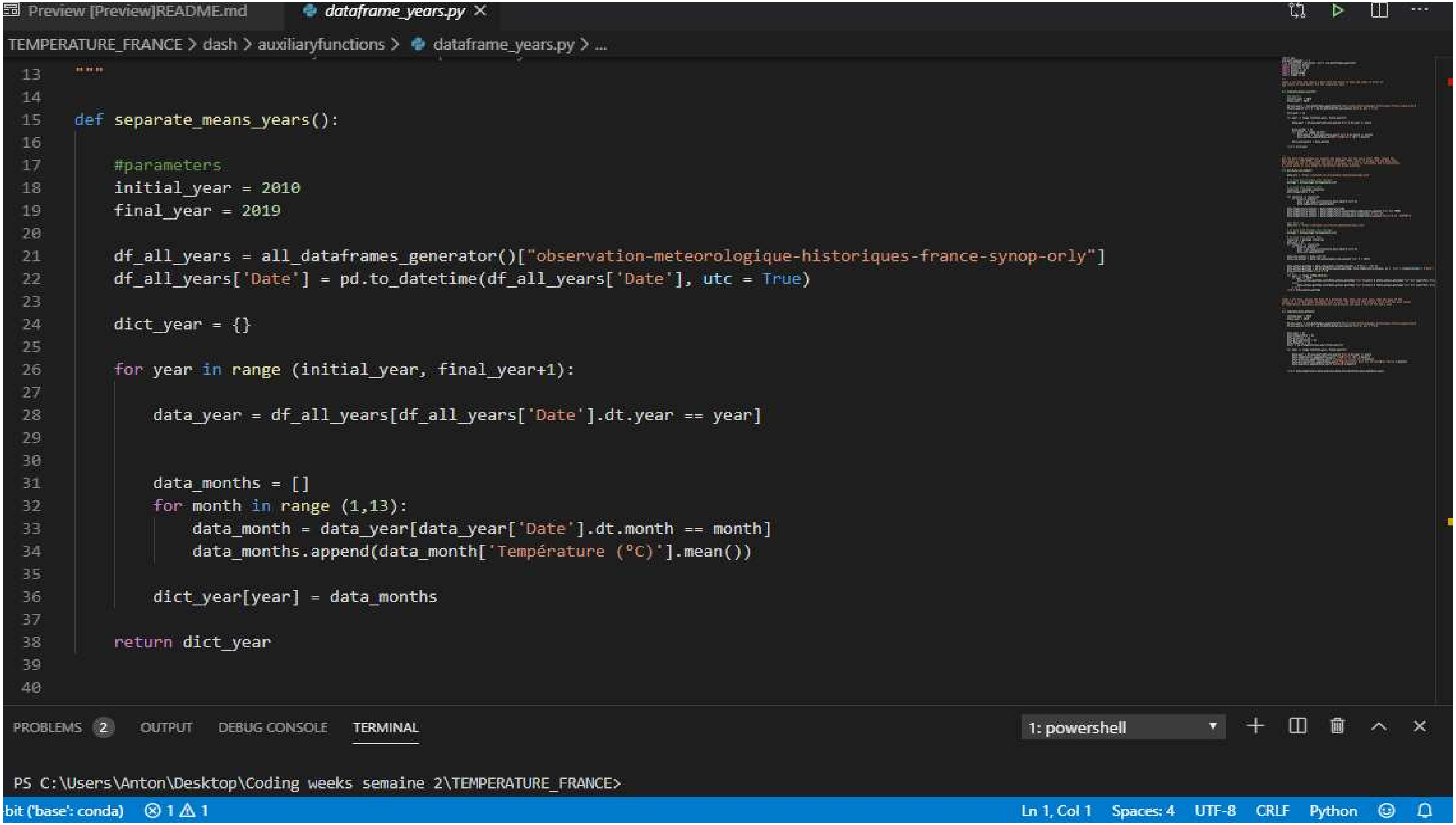
## Description

> A partir des données fournies dans le portail Open Data de Paris-Saclay, datahub, data gouv et CDIAC (Carbon dioxide information analysis center) on réalise des graphiques (carte, camembert et courbes) qui permettent de mieux visualiser et de comprendre comment évolue les données au cours du temps.

Le code contient un module **Corpusutils** qui permet de traiter les données : lire, selectionner des données et réaliser des opérations, d'un fichier **Data** qui contient les toute les données utilisées et un dernier module **Dash** qui permet de réaliser la visualisation.

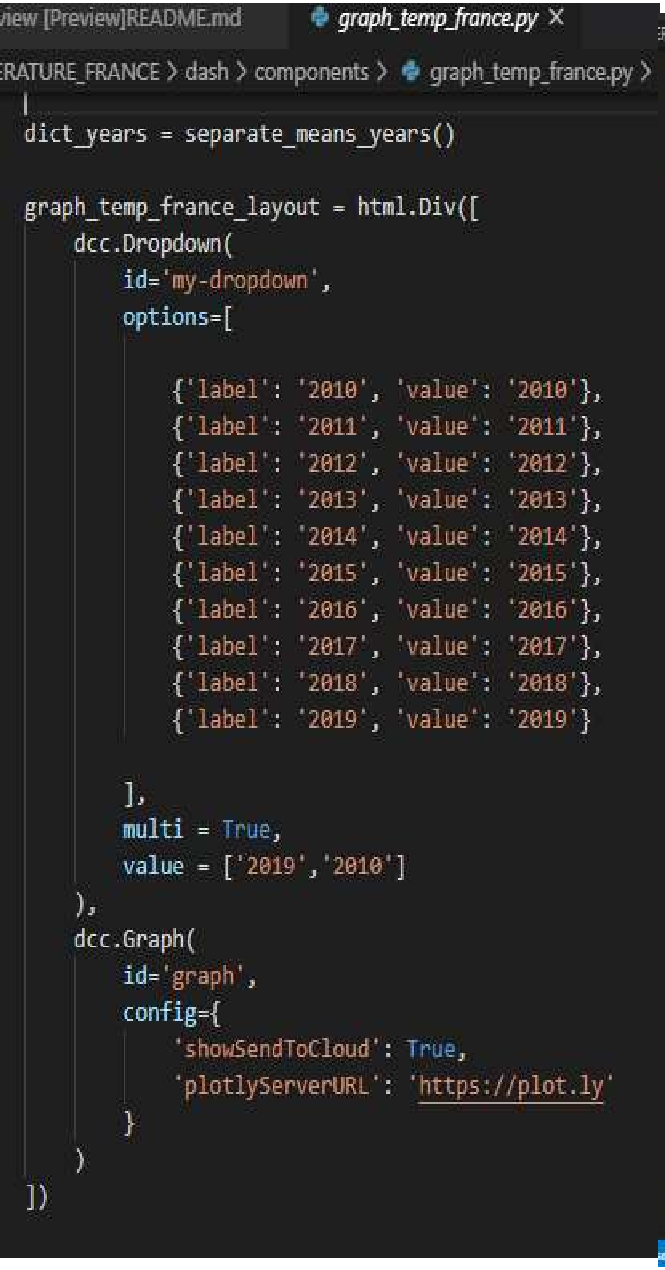
## Installation

```
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS C:\Users\Anton\Desktop\Coding weeks semaine 2\TEMPERATURE_FRANCE>
```

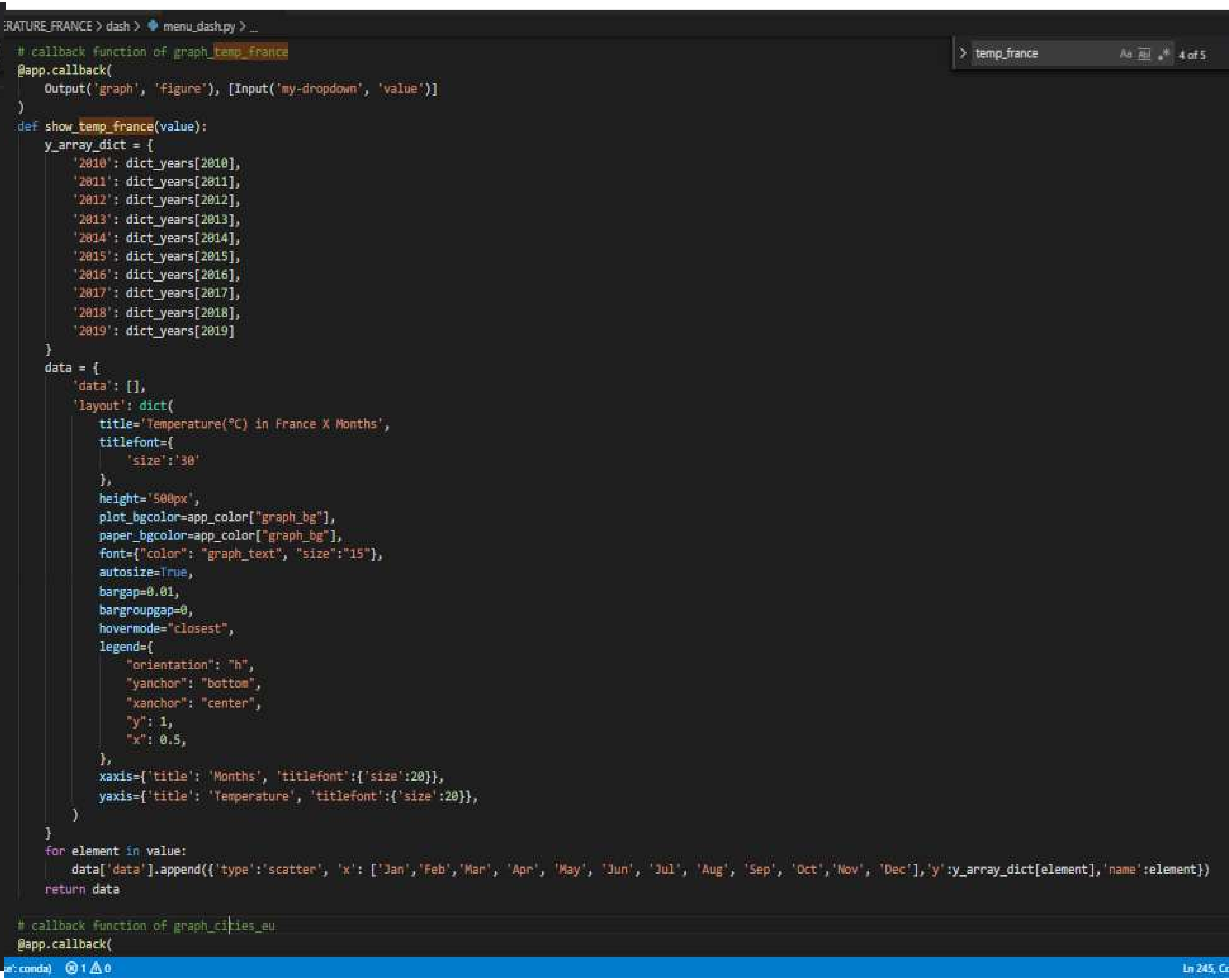


```python
    """

def separate_means_years():

    #parameters
    initial_year = 2010
    final_year = 2019

    df_all_years = all_dataframes_generator()["observation-meteorologique-historiques-france-synop-orly"]
    df_all_years['Date'] = pd.to_datetime(df_all_years['Date'], utc = True)

    dict_year = {}

    for year in range (initial_year, final_year+1):

        data_year = df_all_years[df_all_years['Date'].dt.year == year]


        data_months = []
        for month in range (1,13):
            data_month = data_year[data_year['Date'].dt.month == month]
            data_months.append(data_month['Température (°C)'].mean())

        dict_year[year] = data_months

    return dict_year


```

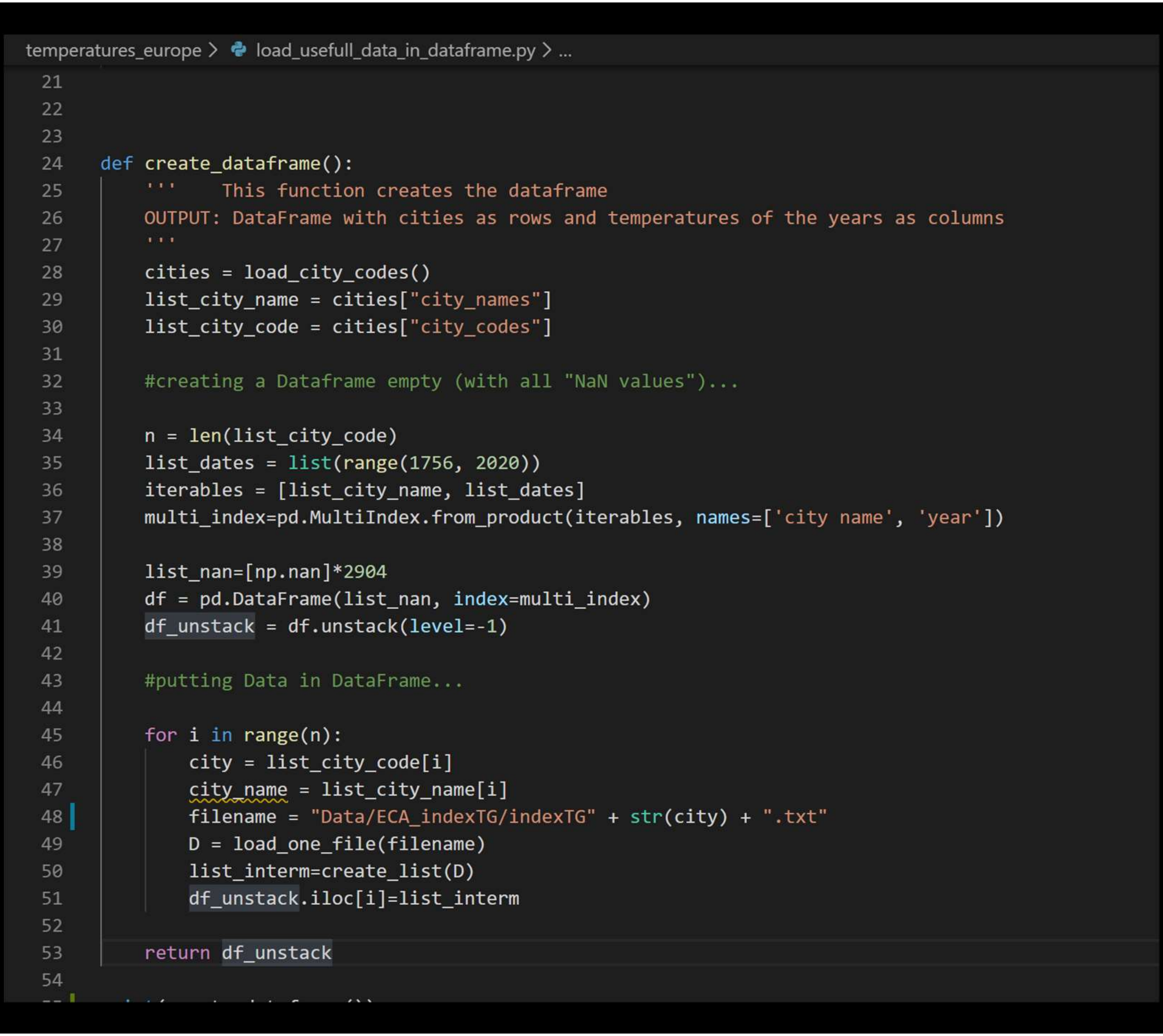
```
PS C:\Users\Anton\Desktop\Coding weeks semaine 2\TEMPERATURE_FRANCE>
```



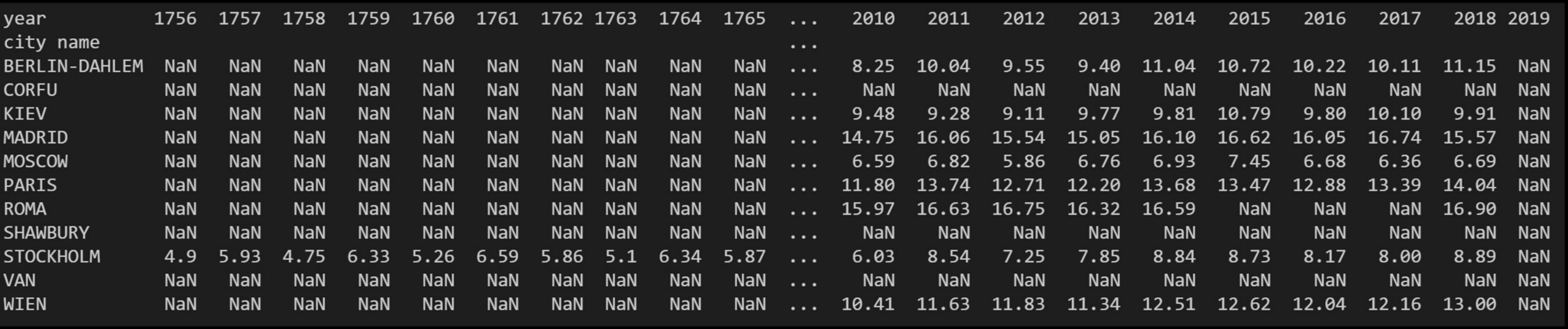
```python
dict_years = separate_means_years()

graph_temp_france_layout = html.Div([
    dcc.Dropdown(
        id='my-dropdown',
        options=[

            {'label': '2010', 'value': '2010'},
            {'label': '2011', 'value': '2011'},
            {'label': '2012', 'value': '2012'},
            {'label': '2013', 'value': '2013'},
            {'label': '2014', 'value': '2014'},
            {'label': '2015', 'value': '2015'},
            {'label': '2016', 'value': '2016'},
            {'label': '2017', 'value': '2017'},
            {'label': '2018', 'value': '2018'},
            {'label': '2019', 'value': '2019'}

        ],
        multi = True,
        value = ['2019','2010']
    ),
    dcc.Graph(
        id='graph',
        config={
            'showSendToCloud': True,
            'plotlyServerURL': 'https://plot.ly'
        }
    )
])
```



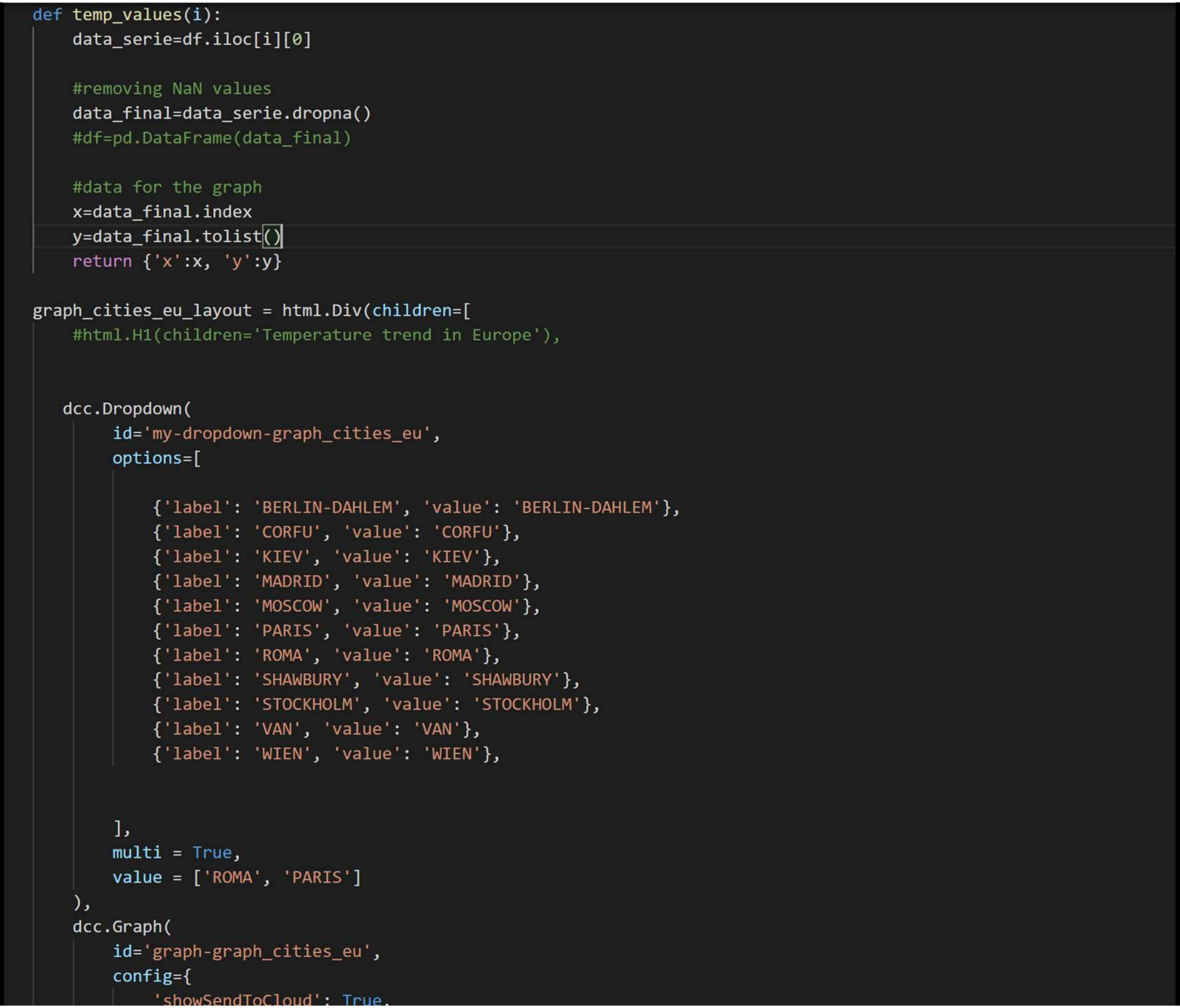
```python
# callback function of graph_temp_france
@app.callback(
    Output('graph', 'figure'), [Input('my-dropdown', 'value')]
)
def show_temp_france(value):
    y_array_dict = {
        '2010': dict_years[2010],
        '2011': dict_years[2011],
        '2012': dict_years[2012],
        '2013': dict_years[2013],
        '2014': dict_years[2014],
        '2015': dict_years[2015],
        '2016': dict_years[2016],
        '2017': dict_years[2017],
        '2018': dict_years[2018],
        '2019': dict_years[2019]
    }
    data = {
        'data': [],
        'layout': dict(
            title='Temperature(°C) in France X Months',
            titlefont={
                'size':'30'
            },
            height='500px',
            plot_bgcolor=app_color["graph_bg"],
            paper_bgcolor=app_color["graph_bg"],
            font={"color": "graph_text", "size":"15"},
            autosize=True,
            bargap=0.01,
            bargroupgap=0,
            hovermode="closest",
            legend={
                "orientation": "h",
                "yanchor": "bottom",
                "xanchor": "center",
                "y": 1,
                "x": 0.5,
            },
            xaxis={'title': 'Months', 'titlefont':{'size':20}},
            yaxis={'title': 'Temperature', 'titlefont':{'size':20}},
        )
    }
    for element in value:
        data['data'].append({'type':'scatter', 'x': ['Jan','Feb','Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct','Nov', 'Dec'],'y':y_array_dict[element],'name':element})
    return data

# callback function of graph_cities_eu
@app.callback(
```


temperatures_europe > load_usefull_data_in_dataframe.py > ...

```python



def create_dataframe():
    '''    This function creates the dataframe
    OUTPUT: DataFrame with cities as rows and temperatures of the years as columns
    '''
    cities = load_city_codes()
    list_city_name = cities["city_names"]
    list_city_code = cities["city_codes"]

    #creating a Dataframe empty (with all "NaN values")...

    n = len(list_city_code)
    list_dates = list(range(1756, 2020))
    iterables = [list_city_name, list_dates]
    multi_index=pd.MultiIndex.from_product(iterables, names=['city name', 'year'])

    list_nan=[np.nan]*2904
    df = pd.DataFrame(list_nan, index=multi_index)
    df_unstack = df.unstack(level=-1)

    #putting Data in DataFrame...

    for i in range(n):
        city = list_city_code[i]
        city_name = list_city_name[i]
        filename = "Data/ECA_indexTG/indexTG" + str(city) + ".txt"
        D = load_one_file(filename)
        list_interm=create_list(D)
        df_unstack.iloc[i]=list_interm

    return df_unstack

```



```
year           1756  1757  1758  1759  1760  1761  1762 1763  1764  1765  ...   2010   2011   2012   2013   2014   2015   2016   2017   2018 2019
city name                                                                 ...
BERLIN-DAHLEM   NaN   NaN   NaN   NaN   NaN   NaN   NaN  NaN   NaN   NaN  ...   8.25  10.04   9.55   9.40  11.04  10.72  10.22  10.11  11.15  NaN
CORFU           NaN   NaN   NaN   NaN   NaN   NaN   NaN  NaN   NaN   NaN  ...    NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN  NaN
KIEV            NaN   NaN   NaN   NaN   NaN   NaN   NaN  NaN   NaN   NaN  ...   9.48   9.28   9.11   9.77   9.81  10.79   9.80  10.10   9.91  NaN
MADRID          NaN   NaN   NaN   NaN   NaN   NaN   NaN  NaN   NaN   NaN  ...  14.75  16.06  15.54  15.05  16.10  16.62  16.05  16.74  15.57  NaN
MOSCOW          NaN   NaN   NaN   NaN   NaN   NaN   NaN  NaN   NaN   NaN  ...   6.59   6.82   5.86   6.76   6.93   7.45   6.68   6.36   6.69  NaN
PARIS           NaN   NaN   NaN   NaN   NaN   NaN   NaN  NaN   NaN   NaN  ...  11.80  13.74  12.71  12.20  13.68  13.47  12.88  13.39  14.04  NaN
ROMA            NaN   NaN   NaN   NaN   NaN   NaN   NaN  NaN   NaN   NaN  ...  15.97  16.63  16.75  16.32  16.59    NaN    NaN    NaN  16.90  NaN
SHAWBURY        NaN   NaN   NaN   NaN   NaN   NaN   NaN  NaN   NaN   NaN  ...    NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN  NaN
STOCKHOLM       4.9  5.93  4.75  6.33  5.26  6.59  5.86  5.1  6.34  5.87  ...   6.03   8.54   7.25   7.85   8.84   8.73   8.17   8.00   8.89  NaN
VAN             NaN   NaN   NaN   NaN   NaN   NaN   NaN  NaN   NaN   NaN  ...    NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN  NaN
WIEN            NaN   NaN   NaN   NaN   NaN   NaN   NaN  NaN   NaN   NaN  ...  10.41  11.63  11.83  11.34  12.51  12.62  12.04  12.16  13.00  NaN
```



```
def temp_values(i):
    data_serie=df.iloc[i][0]

    #removing NaN values
    data_final=data_serie.dropna()
    #df=pd.DataFrame(data_final)

    #data for the graph
    x=data_final.index
    y=data_final.tolist()
    return {'x':x, 'y':y}

graph_cities_eu_layout = html.Div(children=[
    #html.H1(children='Temperature trend in Europe'),


   dcc.Dropdown(
        id='my-dropdown-graph_cities_eu',
        options=[

            {'label': 'BERLIN-DAHLEM', 'value': 'BERLIN-DAHLEM'},
            {'label': 'CORFU', 'value': 'CORFU'},
            {'label': 'KIEV', 'value': 'KIEV'},
            {'label': 'MADRID', 'value': 'MADRID'},
            {'label': 'MOSCOW', 'value': 'MOSCOW'},
            {'label': 'PARIS', 'value': 'PARIS'},
            {'label': 'ROMA', 'value': 'ROMA'},
            {'label': 'SHAWBURY', 'value': 'SHAWBURY'},
            {'label': 'STOCKHOLM', 'value': 'STOCKHOLM'},
            {'label': 'VAN', 'value': 'VAN'},
            {'label': 'WIEN', 'value': 'WIEN'},


        ],
        multi = True,
        value = ['ROMA', 'PARIS']
    ),
    dcc.Graph(
        id='graph-graph_cities_eu',
        config={
            'showSendToCloud': True,
```

# Dash

- Assets : Images
- Auxiliary functions : toutes les fonctions utilisés pour le traitement de la data
- Components : Tous les graphiques
- Menu dash

# Comment ?

## Le MVP

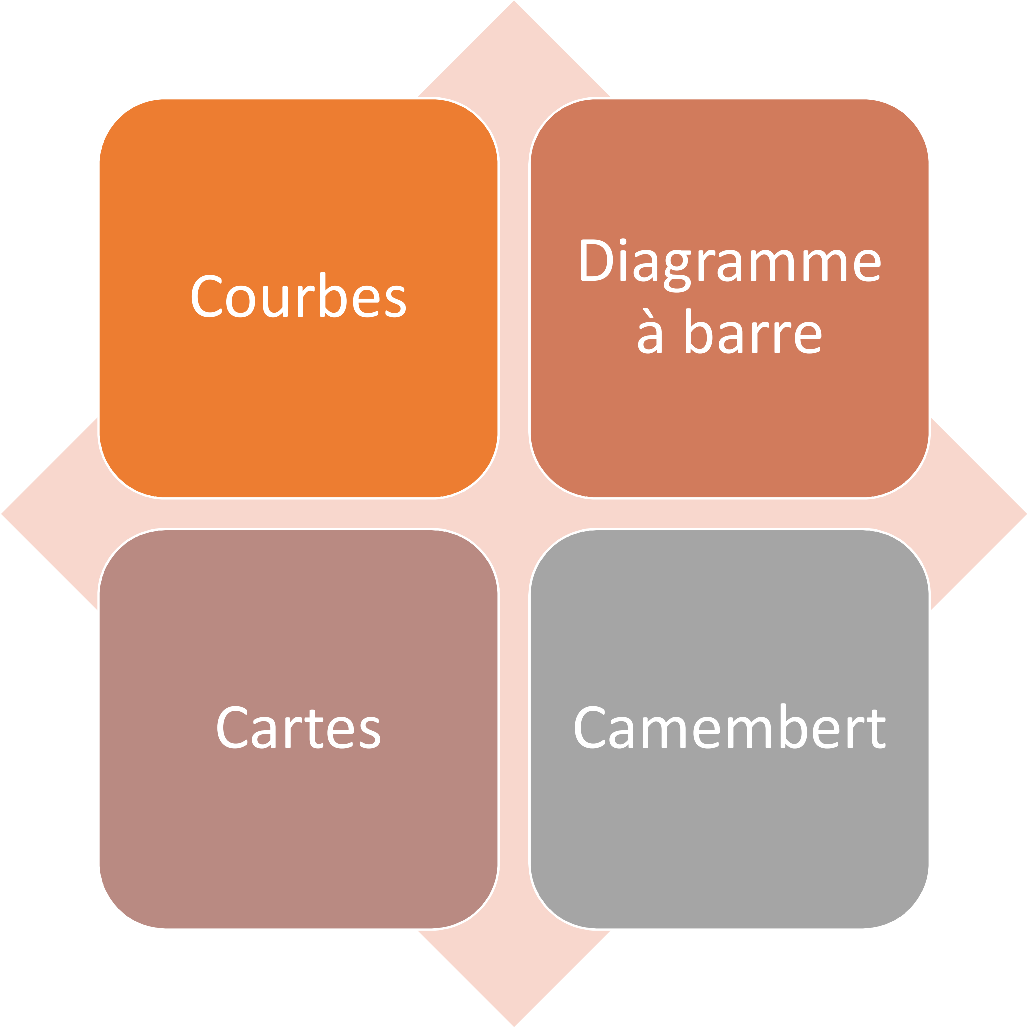
- Température en France par mois
- Température en France par année
- Température dans différentes villes d’Europe par année

## La suite

- Emissions de CO2 dans le monde
- Humidité pression précipitations en France
- Corrélation entre CO2 et température
- Dash
- La présentation

# Types de graphes

# (DASH



Courbes

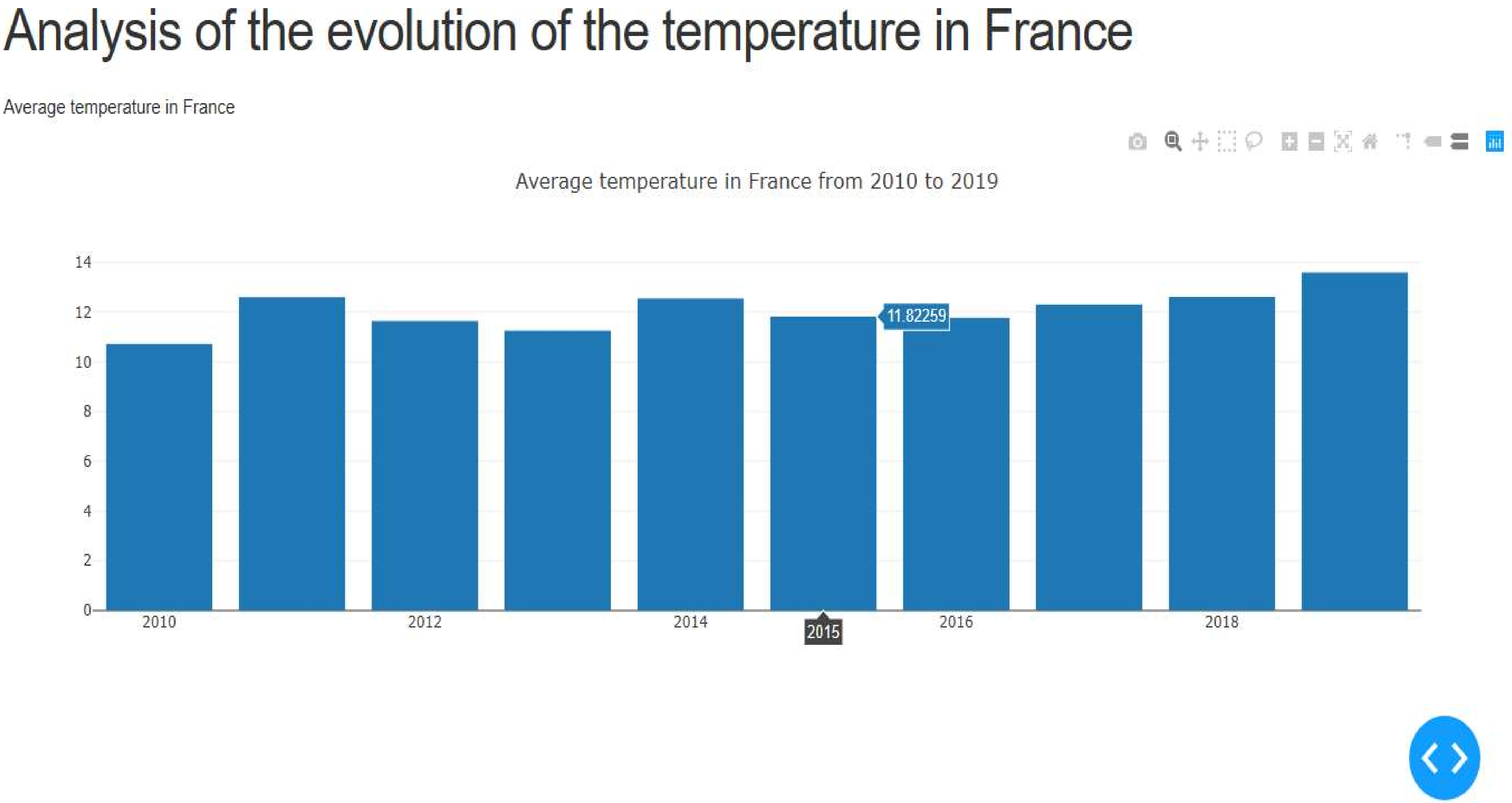
Diagramme à barre

Cartes

Camembert

Merci pour votre attention

# Présentation du dashboard



# Analysis of the evolution of the temperature in France

Average temperature in France

Average temperature in France from 2010 to 2019