Parcours de vectorisation avec pandas

August 29, 2019

Table of Contents

1 Combinaison des propriétés des logements

1.1 Attribution des valeurs de chaque propriété par listes

```
In [1]: import numpy as np

        list_surface = [110, 120, 130, 140, 150]

        list_ceiling_heigh = (2.5 + 0.5 * np.random.random((5, ))).tolist()

        list_share_of_windows = [0.3]
        list_dwelling_per_floor = np.random.randint(2, 10, size=3).tolist()

        list_floors = range(2, 10, 2)
```

1.2 Combinaison des listes de valeurs par une liste à deux dimensions

```
In [2]: housing_characters = list()

        housing_characters.append(list_surface)

        housing_characters.append(list_ceiling_heigh)

        housing_characters += [
            list_share_of_windows, list_dwelling_per_floor, list_floors
        ]

        housing_characters
```

```
Out[2]: [[110, 120, 130, 140, 150],
         [2.9522879195330676,
          2.756376972059785,
          2.628864360935954,
          2.670649106673581,
          2.9237944587795184],
         [0.3],
         [7, 8, 2],
         range(2, 10, 2)]
```

```
In [3]: len(housing_characters)
```

```
Out[3]: 5
```

1.3 Création d'une liste à deux dimensions du produit cartésien

```
In [4]: import itertools

        cartesian_product = list(itertools.product(*housing_characters))

        cartesian_product[:5]
```

```
Out[4]: [(110, 2.9522879195330676, 0.3, 7, 2),
         (110, 2.9522879195330676, 0.3, 7, 4),
         (110, 2.9522879195330676, 0.3, 7, 6),
         (110, 2.9522879195330676, 0.3, 7, 8),
         (110, 2.9522879195330676, 0.3, 8, 2)]
```

```
In [5]: len(cartesian_product)
```

```
Out[5]: 300
```

```
In [6]: len(cartesian_product[0])
```

```
Out[6]: 5
```

2 Instanciation des logements

2.1 Création d’une classe pour un type de logement

```
In [7]: class Collective_Dwelling_CI():
            def __init__(self, a, b, c, d, e):
                self.surface = a
                self.ceiling_heigh = b
                self.share_of_windows = c
                self.dwelling_per_floor = d
                self.floors = e

            def __str__(self):
                return f'\n\nCollective Dwelling CI:' + \
            f'\nSurface: {self.surface},' + \
            f'\nCeiling Heigh: {self.ceiling_heigh},' + \
            f'\nShare of Windows: {self.share_of_windows},' + \
            f'\nDwelling per Floor: {self.dwelling_per_floor},' + \
            f'\nFloors: {self.floors}'

            def __repr__(self):
                return self.__str__()

            def Surface_m(self):
                self.surface_m = (1 - self.share_of_windows) * np.sqrt(
                    self.surface * self.dwelling_per_floor
                ) * 4 * self.ceiling_heigh * (self.floors + 1)
                return self.surface_m

            def Dwellings_total(self):
                self.dwellings_total = self.dwelling_per_floor * (self.floors + 1)
                return self.dwellings_total

            def Surface_component_m(self):
                surface_component_m = self.surface_m / self.dwellings_total
                return surface_component_m

            def Final_calculation(self):
                self.result = dict_min[(self.surface, self.floors)]
                return self.result
```

2.2 Instanciation de la classe de logement à une liste à deux dimensions

```
In [8]: dwellings = list()

        for _tuple in cartesian_product:
            dwellings.append(Collective_Dwelling_CI(*_tuple))

        len(dwellings)
```

```
Out[8]: 300
```

```
In [9]: dwellings[:5]
```

```
Out[9]: [

         Collective Dwelling CI:
         Surface: 110,
         Ceiling Heigh: 2.9522879195330676,
         Share of Windows: 0.3,
         Dwelling per Floor: 7,
         Floors: 2,

         Collective Dwelling CI:
         Surface: 110,
         Ceiling Heigh: 2.9522879195330676,
         Share of Windows: 0.3,
         Dwelling per Floor: 7,
         Floors: 4,

         Collective Dwelling CI:
         Surface: 110,
         Ceiling Heigh: 2.9522879195330676,
         Share of Windows: 0.3,
         Dwelling per Floor: 7,
         Floors: 6,

         Collective Dwelling CI:
         Surface: 110,
         Ceiling Heigh: 2.9522879195330676,
         Share of Windows: 0.3,
         Dwelling per Floor: 7,
         Floors: 8,

         Collective Dwelling CI:
         Surface: 110,
         Ceiling Heigh: 2.9522879195330676,
         Share of Windows: 0.3,
         Dwelling per Floor: 8,
         Floors: 2]
```

```
In [10]: dwellings[0]

Out[10]:

         Collective Dwelling CI:
         Surface: 110,
         Ceiling Heigh: 2.9522879195330676,
         Share of Windows: 0.3,
         Dwelling per Floor: 7,
         Floors: 2

In [11]: dwellings[0].Surface_m()

Out[11]: 688.150386428592

In [12]: dwellings[0].Dwellings_total()

Out[12]: 21

In [13]: dwellings[0].Surface_component_m()

Out[13]: 32.76906602040914
```

3 Combinaisons du calcul économique

3.1 Création du DataFrame selon la liste de combinaison des propriétés

```
In [14]: import pandas as pd

         df = pd.DataFrame(cartesian_product)

         df.shape

Out[14]: (300, 5)

In [15]: df.describe()

Out[15]:                  0           1             2           3           4
         count  300.000000  300.000000  3.000000e+02  300.000000  300.000000
         mean   130.000000    2.786395  3.000000e-01    5.666667    5.000000
         std     14.165765    0.130996  1.668117e-15    2.629055    2.239804
         min    110.000000    2.628864  3.000000e-01    2.000000    2.000000
         25%    120.000000    2.670649  3.000000e-01    2.000000    3.500000
         50%    130.000000    2.756377  3.000000e-01    7.000000    5.000000
         75%    140.000000    2.923794  3.000000e-01    8.000000    6.500000
         max    150.000000    2.952288  3.000000e-01    8.000000    8.000000

In [16]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 5 columns):
0    300 non-null int64
1    300 non-null float64
2    300 non-null float64
3    300 non-null int64
4    300 non-null int64
dtypes: float64(2), int64(3)
memory usage: 11.8 KB
```

3.2 Calcul économique en utilisant des formules uniformes

```
In [17]: df.columns = [
             'Surface', 'Ceiling_Heigh', 'Share_of_Windows', 'Dwelling_per_Floor',
             'Floors'
         ]

         df.head()
```

```
Out[17]:    Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors
         0      110       2.952288               0.3                   7       2
         1      110       2.952288               0.3                   7       4
         2      110       2.952288               0.3                   7       6
         3      110       2.952288               0.3                   7       8
         4      110       2.952288               0.3                   8       2
```

```
In [18]: df['Surface_m'] = (1 - df['Share_of_Windows']) * np.sqrt(
             df['Surface'] * df['Dwelling_per_Floor']) * 4 * df['Ceiling_Heigh'] * (
                 df['Floors'] + 1)

         df['Dwellings_total'] = df['Dwelling_per_Floor'] * (df['Floors'] + 1)

         df['Surface_Component_m'] = df['Surface_m'] / df['Dwellings_total']

         df.head()
```

```
Out[18]:    Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors  \
         0      110       2.952288               0.3                   7       2
         1      110       2.952288               0.3                   7       4
         2      110       2.952288               0.3                   7       6
         3      110       2.952288               0.3                   7       8
         4      110       2.952288               0.3                   8       2

              Surface_m  Dwellings_total  Surface_Component_m
         0   688.150386               21            32.769066
         1  1146.917311               35            32.769066
         2  1605.684235               49            32.769066
         3   2064.451159               63            32.769066
         4    735.663708               24            30.652654
```

3.3 Calcul économique en utilisant des formules à partir de booléen

```
In [19]: bool_floors = df.Floors >= 8

         df.loc[(bool_floors
                 ), 'Dwelling_total_bool'] = df['Dwelling_per_Floor'] * df['Floors']

         df.head()

Out[19]:     Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors  \
         0       110       2.952288               0.3                   7       2
         1       110       2.952288               0.3                   7       4
         2       110       2.952288               0.3                   7       6
         3       110       2.952288               0.3                   7       8
         4       110       2.952288               0.3                   8       2

               Surface_m  Dwellings_total  Surface_Component_m  Dwelling_total_bool
         0    688.150386               21            32.769066                  NaN
         1   1146.917311               35            32.769066                  NaN
         2   1605.684235               49            32.769066                  NaN
         3   2064.451159               63            32.769066                 56.0
         4    735.663708               24            30.652654                  NaN
```

```
In [20]: df.loc[~bool_floors, 'Dwelling_total_bool'] = df['Dwelling_per_Floor'] * (
             df['Floors'] + 1)

         df.head()

Out[20]:     Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors  \
         0       110       2.952288               0.3                   7       2
         1       110       2.952288               0.3                   7       4
         2       110       2.952288               0.3                   7       6
         3       110       2.952288               0.3                   7       8
         4       110       2.952288               0.3                   8       2

               Surface_m  Dwellings_total  Surface_Component_m  Dwelling_total_bool
         0    688.150386               21            32.769066                 21.0
         1   1146.917311               35            32.769066                 35.0
         2   1605.684235               49            32.769066                 49.0
         3   2064.451159               63            32.769066                 56.0
         4    735.663708               24            30.652654                 24.0
```

3.4 Calcul économique en utilisant des formules à partir d’un intervalle sélectionné

```
In [21]: between_floors = df.Floors.between(2, 6)
```

```
         df['Dwelling_total_between'] = between_floors * df['Dwelling_per_Floor'] \
         * (df['Floors'] + 1) + (1 - between_floors) * df['Dwelling_per_Floor'] * \
         df['Floors']

         df.head()
```

```
Out[21]:    Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors  \
         0      110       2.952288               0.3                   7       2
         1      110       2.952288               0.3                   7       4
         2      110       2.952288               0.3                   7       6
         3      110       2.952288               0.3                   7       8
         4      110       2.952288               0.3                   8       2

              Surface_m  Dwellings_total  Surface_Component_m  Dwelling_total_bool  \
         0   688.150386               21            32.769066                 21.0
         1  1146.917311               35            32.769066                 35.0
         2  1605.684235               49            32.769066                 49.0
         3  2064.451159               63            32.769066                 56.0
         4   735.663708               24            30.652654                 24.0

            Dwelling_total_between
         0                      21
         1                      35
         2                      49
         3                      56
         4                      24
```

3.5 Calcul économique en matrice à partir de booléen

```
In [22]: matrix = df.values
         dwelling_total_matrix = (bool_floors * matrix[:, 4] +
                                  (1 - bool_floors) * (matrix[:, 4] + 1)) * \
         matrix[:, 3]
         df['Dwelling_total_matrix'] = dwelling_total_matrix
         df.head()
```

```
Out[22]:    Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors  \
         0      110       2.952288               0.3                   7       2
         1      110       2.952288               0.3                   7       4
         2      110       2.952288               0.3                   7       6
         3      110       2.952288               0.3                   7       8
         4      110       2.952288               0.3                   8       2

              Surface_m  Dwellings_total  Surface_Component_m  Dwelling_total_bool  \
         0   688.150386               21            32.769066                 21.0
         1  1146.917311               35            32.769066                 35.0
         2  1605.684235               49            32.769066                 49.0
         3  2064.451159               63            32.769066                 56.0
```

```
4   735.663708               24            30.652654                 24.0

   Dwelling_total_between  Dwelling_total_matrix
0                      21                   21.0
1                      35                   35.0
2                      49                   49.0
3                      56                   56.0
4                      24                   24.0
```

```
In [23]: df['Modification'] = df.Dwellings_total > dwelling_total_matrix
         df.head()
```

```
Out[23]:    Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors  \
         0      110       2.952288               0.3                   7       2
         1      110       2.952288               0.3                   7       4
         2      110       2.952288               0.3                   7       6
         3      110       2.952288               0.3                   7       8
         4      110       2.952288               0.3                   8       2

              Surface_m  Dwellings_total  Surface_Component_m  Dwelling_total_bool  \
         0   688.150386               21            32.769066                 21.0
         1  1146.917311               35            32.769066                 35.0
         2  1605.684235               49            32.769066                 49.0
         3  2064.451159               63            32.769066                 56.0
         4   735.663708               24            30.652654                 24.0

            Dwelling_total_between  Dwelling_total_matrix  Modification
         0                      21                   21.0         False
         1                      35                   35.0         False
         2                      49                   49.0         False
         3                      56                   56.0          True
         4                      24                   24.0         False
```

3.6 Calcul économique en utilisant des fonctions

3.6.1 La bonne performance

```
In [24]: dict_hall_info = {False: 'Without Hall', True: 'With Hall'}
         dict_hall_info
```

```
Out[24]: {False: 'Without Hall', True: 'With Hall'}
```

```
In [25]: %%timeit
         df['Hall_Info'] = df.Modification.apply(lambda x: dict_hall_info[x])
```

```
403 ţs ś 13.4 ţs per loop (mean ś std. dev. of 7 runs, 1000 loops each)
```

```
In [26]: df.head()
```

```
Out[26]:    Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors  \
         0      110       2.952288               0.3                   7       2
         1      110       2.952288               0.3                   7       4
         2      110       2.952288               0.3                   7       6
         3      110       2.952288               0.3                   7       8
         4      110       2.952288               0.3                   8       2

              Surface_m  Dwellings_total  Surface_Component_m  Dwelling_total_bool  \
         0   688.150386               21            32.769066                 21.0
         1  1146.917311               35            32.769066                 35.0
         2  1605.684235               49            32.769066                 49.0
         3  2064.451159               63            32.769066                 56.0
         4   735.663708               24            30.652654                 24.0

            Dwelling_total_between  Dwelling_total_matrix  Modification     Hall_Info
         0                      21                   21.0         False  Without Hall
         1                      35                   35.0         False  Without Hall
         2                      49                   49.0         False  Without Hall
         3                      56                   56.0          True     With Hall
         4                      24                   24.0         False  Without Hall
```

3.6.2 La mauvaise performance

```
In [27]: def func_apply(x):

             if x.Dwelling_total_matrix < x.Dwellings_total:
                 return False
             else:
                 return True
```

```
In [28]: %%timeit
         df['Without_Hall'] = df.apply(lambda x: func_apply(x), axis=1)
```

```
8.55 ms ś 330 ţs per loop (mean ś std. dev. of 7 runs, 100 loops each)
```

```
In [29]: df.head()
```

```
Out[29]:    Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors  \
         0      110       2.952288               0.3                   7       2
         1      110       2.952288               0.3                   7       4
         2      110       2.952288               0.3                   7       6
         3      110       2.952288               0.3                   7       8
         4      110       2.952288               0.3                   8       2

              Surface_m  Dwellings_total  Surface_Component_m  Dwelling_total_bool  \
         0   688.150386               21            32.769066                 21.0
         1  1146.917311               35            32.769066                 35.0
         2  1605.684235               49            32.769066                 49.0
```

```
3  2064.451159               63            32.769066                 56.0
4   735.663708               24            30.652654                 24.0

   Dwelling_total_between  Dwelling_total_matrix  Modification     Hall_Info  \
0                      21                   21.0         False  Without Hall
1                      35                   35.0         False  Without Hall
2                      49                   49.0         False  Without Hall
3                      56                   56.0          True     With Hall
4                      24                   24.0         False  Without Hall

   Without_Hall
0          True
1          True
2          True
3         False
4          True
```

4 Jointure de table

```
In [30]: df_autre = pd.DataFrame(list(itertools.product(*housing_characters[0:2])))
         df_autre.shape
```

```
Out[30]: (25, 2)
```

```
In [31]: df_autre.head()
```

```
Out[31]:      0         1
         0  110  2.952288
         1  110  2.756377
         2  110  2.628864
         3  110  2.670649
         4  110  2.923794
```

```
In [32]: df_autre.columns = ['Surface', 'Ceiling_Heigh']

         dict_size_info = {
             110: 'Small',
             120: 'Middle-Small',
             130: 'Middl',
             140: 'Middl-Large',
             150: 'Large'
         }

         df_autre['Size_Info'] = df_autre.Surface.apply(lambda x: dict_size_info[x])

         df_autre.shape
```

```
Out[32]: (25, 3)
```

```
In [33]: df_autre.head()
```

```
Out[33]:    Surface  Ceiling_Heigh Size_Info
         0      110       2.952288     Small
         1      110       2.756377     Small
         2      110       2.628864     Small
         3      110       2.670649     Small
         4      110       2.923794     Small
```

```
In [34]: df = df.merge(df_autre, how='left', on=['Surface', 'Ceiling_Heigh'])
         df.shape
```

```
Out[34]: (300, 15)
```

```
In [35]: df.head()
```

```
Out[35]:    Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors  \
         0      110       2.952288               0.3                   7       2
         1      110       2.952288               0.3                   7       4
         2      110       2.952288               0.3                   7       6
         3      110       2.952288               0.3                   7       8
         4      110       2.952288               0.3                   8       2

              Surface_m  Dwellings_total  Surface_Component_m  Dwelling_total_bool  \
         0   688.150386               21            32.769066                 21.0
         1  1146.917311               35            32.769066                 35.0
         2  1605.684235               49            32.769066                 49.0
         3  2064.451159               63            32.769066                 56.0
         4   735.663708               24            30.652654                 24.0

            Dwelling_total_between  Dwelling_total_matrix  Modification     Hall_Info  \
         0                      21                   21.0         False  Without Hall
         1                      35                   35.0         False  Without Hall
         2                      49                   49.0         False  Without Hall
         3                      56                   56.0          True     With Hall
         4                      24                   24.0         False  Without Hall

            Without_Hall Size_Info
         0          True     Small
         1          True     Small
         2          True     Small
         3         False     Small
         4          True     Small
```

5 Calcul du minimum par logement

5.1 Calcul avec des sorts

```
In [36]: essential_properties = ['Surface', 'Floors']
         target_value = ['Dwelling_per_Floor']
```

```
In [37]: df_min = df.sort_values(essential_properties + target_value)
         df_min.shape

Out[37]: (300, 15)

In [38]: df_min.head()

Out[38]:     Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors  \
         8       110       2.952288               0.3                   2       2
         20      110       2.756377               0.3                   2       2
         32      110       2.628864               0.3                   2       2
         44      110       2.670649               0.3                   2       2
         56      110       2.923794               0.3                   2       2

              Surface_m  Dwellings_total  Surface_Component_m  Dwelling_total_bool  \
         8   367.831854                6            61.305309                  6.0
         20  343.422891                6            57.237148                  6.0
         32  327.535822                6            54.589304                  6.0
         44  332.741873                6            55.456979                  6.0
         56  364.281793                6            60.713632                  6.0

             Dwelling_total_between  Dwelling_total_matrix  Modification     Hall_Info  \
         8                        6                    6.0         False  Without Hall
         20                       6                    6.0         False  Without Hall
         32                       6                    6.0         False  Without Hall
         44                       6                    6.0         False  Without Hall
         56                       6                    6.0         False  Without Hall

             Without_Hall Size_Info
         8           True     Small
         20          True     Small
         32          True     Small
         44          True     Small
         56          True     Small

In [39]: df_min_droped = df.sort_values(essential_properties +
                                        target_value).drop_duplicates(
                                            tuple(essential_properties))
         df_min_droped.shape

Out[39]: (20, 15)

In [40]: df_min_droped.head()

Out[40]:     Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors  \
         8       110       2.952288               0.3                   2       2
         9       110       2.952288               0.3                   2       4
         10      110       2.952288               0.3                   2       6
         11      110       2.952288               0.3                   2       8
```

```
68      120       2.952288               0.3                     2       2

      Surface_m  Dwellings_total  Surface_Component_m  Dwelling_total_bool  \
8    367.831854                6            61.305309                  6.0
9    613.053090               10            61.305309                 10.0
10   858.274326               14            61.305309                 14.0
11  1103.495561               18            61.305309                 16.0
68   384.187841                6            64.031307                  6.0

    Dwelling_total_between  Dwelling_total_matrix  Modification     Hall_Info  \
8                        6                    6.0         False  Without Hall
9                       10                   10.0         False  Without Hall
10                      14                   14.0         False  Without Hall
11                      16                   16.0          True     With Hall
68                       6                    6.0         False  Without Hall

    Without_Hall     Size_Info
8           True         Small
9           True         Small
10          True         Small
11         False         Small
68          True  Middle-Small
```

5.2 Transformation en dictionnaire

```
In [41]: dict_min = df_min_droped.set_index(['Surface',
                                             'Floors']).to_dict(orient='index')
         dict_min.keys()
```

```
Out[41]: dict_keys([(110, 2), (110, 4), (110, 6), (110, 8), (120, 2), (120, 4), (120, 6), (120
```

```
In [42]: list(dict_min.values())[:2]
```

```
Out[42]: [{'Ceiling_Heigh': 2.9522879195330676,
           'Share_of_Windows': 0.3,
           'Dwelling_per_Floor': 2,
           'Surface_m': 367.83185379884054,
           'Dwellings_total': 6,
           'Surface_Component_m': 61.305308966473426,
           'Dwelling_total_bool': 6.0,
           'Dwelling_total_between': 6,
           'Dwelling_total_matrix': 6.0,
           'Modification': False,
           'Hall_Info': 'Without Hall',
           'Without_Hall': True,
           'Size_Info': 'Small'},
          {'Ceiling_Heigh': 2.9522879195330676,
           'Share_of_Windows': 0.3,
           'Dwelling_per_Floor': 2,
           'Surface_m': 613.0530896647342,
           'Dwellings_total': 10,
           'Surface_Component_m': 61.30530896647342,
           'Dwelling_total_bool': 10.0,
           'Dwelling_total_between': 10,
           'Dwelling_total_matrix': 10.0,
           'Modification': False,
           'Hall_Info': 'Without Hall',
           'Without_Hall': True,
           'Size_Info': 'Small'}]
```

```
In [43]: dict_min[(110, 2)]
```

```
Out[43]: {'Ceiling_Heigh': 2.9522879195330676,
          'Share_of_Windows': 0.3,
          'Dwelling_per_Floor': 2,
          'Surface_m': 367.83185379884054,
          'Dwellings_total': 6,
          'Surface_Component_m': 61.305308966473426,
          'Dwelling_total_bool': 6.0,
          'Dwelling_total_between': 6,
          'Dwelling_total_matrix': 6.0,
          'Modification': False,
          'Hall_Info': 'Without Hall',
          'Without_Hall': True,
          'Size_Info': 'Small'}
```

```
In [44]: list(dict_min.items())[:2]
```

```
Out[44]: [((110, 2),
           {'Ceiling_Heigh': 2.9522879195330676,
            'Share_of_Windows': 0.3,
            'Dwelling_per_Floor': 2,
            'Surface_m': 367.83185379884054,
            'Dwellings_total': 6,
            'Surface_Component_m': 61.305308966473426,
            'Dwelling_total_bool': 6.0,
            'Dwelling_total_between': 6,
            'Dwelling_total_matrix': 6.0,
            'Modification': False,
            'Hall_Info': 'Without Hall',
            'Without_Hall': True,
            'Size_Info': 'Small'}),
          ((110, 4),
           {'Ceiling_Heigh': 2.9522879195330676,
            'Share_of_Windows': 0.3,
            'Dwelling_per_Floor': 2,
            'Surface_m': 613.0530896647342,
            'Dwellings_total': 10,
            'Surface_Component_m': 61.30530896647342,
            'Dwelling_total_bool': 10.0,
            'Dwelling_total_between': 10,
            'Dwelling_total_matrix': 10.0,
            'Modification': False,
            'Hall_Info': 'Without Hall',
            'Without_Hall': True,
            'Size_Info': 'Small'})]
```

6 Calcul final qui consiste uniquement à requêter le résultat dans le dictionnaire réduit

```
In [45]: list_surface_component_m = []
         for dwelling in dwellings:
             list_surface_component_m.append(dwelling.Final_calculation())
         len(list_surface_component_m)
```

```
Out[45]: 300
```

```
In [46]: list_surface_component_m[0]
```

```
Out[46]: {'Ceiling_Heigh': 2.9522879195330676,
          'Share_of_Windows': 0.3,
          'Dwelling_per_Floor': 2,
          'Surface_m': 367.83185379884054,
          'Dwellings_total': 6,
          'Surface_Component_m': 61.305308966473426,
          'Dwelling_total_bool': 6.0,
          'Dwelling_total_between': 6,
          'Dwelling_total_matrix': 6.0,
          'Modification': False,
          'Hall_Info': 'Without Hall',
          'Without_Hall': True,
          'Size_Info': 'Small'}
```

7 Dimensionnement et segmentation du problème

```
In [47]: df_extended = pd.DataFrame()

         for i in range(0, 6):
             df_partition = pd.DataFrame(
                 cartesian_product,
                 columns=[
                     'Surface', 'Ceiling_Heigh', 'Share_of_Windows',
                     'Dwelling_per_Floor', 'Floors'
                 ]).iloc[int((300 / 6) * i):int((300 / 6) * (i + 1)), :]
             df_extended = pd.concat([df_extended, df_partition], axis=0)
         df_extended.shape
```

```
Out[47]: (300, 5)
```

```
In [48]: df_extended.iloc[250:, :].head()
```

```
Out[48]:      Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors
         250      150       2.952288               0.3                   2       6
         251      150       2.952288               0.3                   2       8
         252      150       2.756377               0.3                   7       2
         253      150       2.756377               0.3                   7       4
         254      150       2.756377               0.3                   7       6
```

```
In [49]: df_partition.shape
```

```
Out[49]: (50, 5)
```

```
In [50]: df_partition.head()
```

```
Out[50]:      Surface  Ceiling_Heigh  Share_of_Windows  Dwelling_per_Floor  Floors
         250      150       2.952288               0.3                   2       6
         251      150       2.952288               0.3                   2       8
         252      150       2.756377               0.3                   7       2
         253      150       2.756377               0.3                   7       4
         254      150       2.756377               0.3                   7       6
```

8 Virement des combinaisons

```
In [51]: combination_to_ignore = (110, 6)
```

```
In [52]: list_filtered = [element for element in cartesian_product if not \
                           set(combination_to_ignore) < set(element)]
         len(list_filtered)
```

```
Out[52]: 285
```

```
In [53]: list_filtered[:10]
```

```
Out[53]: [(110, 2.9522879195330676, 0.3, 7, 2),
          (110, 2.9522879195330676, 0.3, 7, 4),
          (110, 2.9522879195330676, 0.3, 7, 8),
          (110, 2.9522879195330676, 0.3, 8, 2),
          (110, 2.9522879195330676, 0.3, 8, 4),
          (110, 2.9522879195330676, 0.3, 8, 8),
          (110, 2.9522879195330676, 0.3, 2, 2),
          (110, 2.9522879195330676, 0.3, 2, 4),
          (110, 2.9522879195330676, 0.3, 2, 8),
          (110, 2.756376972059785, 0.3, 7, 2)]
```