script_codici_green

June 17, 2022

[ ]:
```
import pandas as pd
import numpy as np

raw_data = pd.read_excel("2IPCGreenInventoryList_2022WIPO.xlsx", dtype=str)
raw_data = raw_data[[ "IPC"]]

raw_data.dropna(inplace=True)
raw_data
```

[ ]:
```
                       IPC
2     C10L 5/00, 5/40-5/48
3                C10B 53/02
4           C10L 5/40, 9/00
5    C10L 1/00, 1/02, 1/14
6           C10L 1/02, 1/19
..                       …
287                     G21
288                    G21B
289                    G21C
290                    G21D
291              F02C 1/05

[271 rows x 1 columns]
```

[ ]:
```
# sostituire lo spazio speciale nonbreaking space dopo le virgole di ogni riga
# raw_data['IPC'] = raw_data['IPC'].str.replace("\xc2\xa0", " ")
```

[ ]:
```
# import os

# for 2IPCGreenInventoryList_2022WIPO.xlsx in os.listdir():
#         'green_inventory.xlsx' = '2IPCGreenInventoryList_2022WIPO.xlsx'.
 ↪replace("\xa0", " ")
#         os.rename('2IPCGreenInventoryList_2022WIPO.xlsx', 'green_inventory.
 ↪xlsx')
```

[ ]:
```
cols_to_strip = [ 'IPC']
for ccc in cols_to_strip:
```

```
        if isinstance(ccc, str):
                raw_data[ccc] = raw_data[ccc].str.replace(  '\\t',      ' ',␣
↪regex = True)
                raw_data[ccc] = raw_data[ccc].str.replace(  '\xc2\xa0',      '␣
↪', regex = True)
                raw_data[ccc] = raw_data[ccc].str.replace(  '\xa0',      ' ',␣
↪regex = True)
                print(ccc, " ok")
```

```
IPC  ok
```

[ ]:
```
raw_data[ "IPC"].astype(str)
raw_data.head(30)
```

[ ]:

	IPC
2	C10L 5/00, 5/40-5/48
3	C10B 53/02
4	C10L 5/40, 9/00
5	C10L 1/00, 1/02, 1/14
6	C10L 1/02, 1/19
7	C07C 67/00, 69/00
8	C10G
9	C10L 1/02, 1/19
10	C11C 3/10
11	C12P 7/649
12	C10L 1/02, 1/182
13	C12N 9/24
14	C12P 7/06-7/14
15	C02F 3/28, 11/04
16	C10L 3/00
17	C12M 1/107
18	C12P 5/02
19	C12N 1/13, 1/15, 1/21, 5/10, 15/00
20	A01H
21	C10L 3/00
22	F02C 3/28
23	H01M 4/86-4/98, 8/00-8/24, 12/00-12/08
24	H01M 4/86-4/98
25	H01M 4/86-4/98
26	H01M 8/00-8/24, 50/00-50/171
27	H01M 12/00-12/08
28	C10B 53/00
29	C10J
31	C10L 5/00
32	C10L 5/42, 5/44

[ ]:
```
raw_data['radice'] = raw_data['IPC'].str.split(' ').str[0]

raw_data['primo'] = raw_data['IPC'].str.split(',').str[0]
raw_data['primo'] = raw_data['primo'].str.split(' ').str[1]

raw_data['secondo'] = raw_data['IPC'].str.split(',').str[1]
raw_data['terzo'] = raw_data['IPC'].str.split(',').str[2]
raw_data['quarto'] = raw_data['IPC'].str.split(',').str[3]
raw_data['quinto'] = raw_data['IPC'].str.split(',').str[4]
raw_data['sesto'] = raw_data['IPC'].str.split(',').str[5]
raw_data['settimo'] = raw_data['IPC'].str.split(',').str[6]

#drop colonne che non ci servono
# raw_data = raw_data.drop(columns=['IPC'])

# drop delle colonne che presentano tutti valori NaN
raw_data = raw_data.dropna(axis='columns', how ='all')

raw_data.head(27)
```

[ ]:

	IPC	radice	primo	secondo	\
2	C10L 5/00, 5/40-5/48	C10L	5/00	5/40-5/48	
3	C10B 53/02	C10B	53/02	NaN	
4	C10L 5/40, 9/00	C10L	5/40	9/00	
5	C10L 1/00, 1/02, 1/14	C10L	1/00	1/02	
6	C10L 1/02, 1/19	C10L	1/02	1/19	
7	C07C 67/00, 69/00	C07C	67/00	69/00	
8	C10G	C10G	NaN	NaN	
9	C10L 1/02, 1/19	C10L	1/02	1/19	
10	C11C 3/10	C11C	3/10	NaN	
11	C12P 7/649	C12P	7/649	NaN	
12	C10L 1/02, 1/182	C10L	1/02	1/182	
13	C12N 9/24	C12N	9/24	NaN	
14	C12P 7/06-7/14	C12P	7/06-7/14	NaN	
15	C02F 3/28, 11/04	C02F	3/28	11/04	
16	C10L 3/00	C10L	3/00	NaN	
17	C12M 1/107	C12M	1/107	NaN	
18	C12P 5/02	C12P	5/02	NaN	
19	C12N 1/13, 1/15, 1/21, 5/10, 15/00	C12N	1/13	1/15	
20	A01H	A01H	NaN	NaN	
21	C10L 3/00	C10L	3/00	NaN	
22	F02C 3/28	F02C	3/28	NaN	
23	H01M 4/86-4/98, 8/00-8/24, 12/00-12/08	H01M	4/86-4/98	8/00-8/24	
24	H01M 4/86-4/98	H01M	4/86-4/98	NaN	
25	H01M 4/86-4/98	H01M	4/86-4/98	NaN	
26	H01M 8/00-8/24, 50/00-50/171	H01M	8/00-8/24	50/00-50/171	
27	H01M 12/00-12/08	H01M	12/00-12/08	NaN	

```
28                              C10B 53/00   C10B        53/00            NaN
```

```
           terzo quarto  quinto sesto
2            NaN    NaN     NaN   NaN
3            NaN    NaN     NaN   NaN
4            NaN    NaN     NaN   NaN
5           1/14    NaN     NaN   NaN
6            NaN    NaN     NaN   NaN
7            NaN    NaN     NaN   NaN
8            NaN    NaN     NaN   NaN
9            NaN    NaN     NaN   NaN
10           NaN    NaN     NaN   NaN
11           NaN    NaN     NaN   NaN
12           NaN    NaN     NaN   NaN
13           NaN    NaN     NaN   NaN
14           NaN    NaN     NaN   NaN
15           NaN    NaN     NaN   NaN
16           NaN    NaN     NaN   NaN
17           NaN    NaN     NaN   NaN
18           NaN    NaN     NaN   NaN
19          1/21   5/10   15/00   NaN
20           NaN    NaN     NaN   NaN
21           NaN    NaN     NaN   NaN
22           NaN    NaN     NaN   NaN
23   12/00-12/08    NaN     NaN   NaN
24           NaN    NaN     NaN   NaN
25           NaN    NaN     NaN   NaN
26           NaN    NaN     NaN   NaN
27           NaN    NaN     NaN   NaN
28           NaN    NaN     NaN   NaN
```

recuer

[ ]:
```
# substr = ","
# i=0

# for index, row in raw_data.iterrows():
#     i+=1
#     codicecompleto = row['IPC']

#     if substr in codicecompleto:
#         raw_data['radice'] = raw_data['IPC'].str.split(' ').str[0]
#         raw_data['resto'] = raw_data['IPC'].str.split(' ').str[1]

#     raw_data['primo'] = raw_data['resto'].str.split(',').str[0]
#     raw_data['secondo'] = raw_data['resto'].str.split(',').str[1]
#     raw_data['terzo'] = raw_data['resto'].str.split(',').str[2]
```

```
#     raw_data['quarto'] = raw_data['resto'].str.split(',').str[3]
#     raw_data['quinto'] = raw_data['resto'].str.split(',').str[4]
#     raw_data['sesto'] = raw_data['resto'].str.split(',').str[5]
#     raw_data['settimo'] = raw_data['resto'].str.split(',').str[6]

# #drop colonne che non ci servono
# # raw_data = raw_data.drop(columns=['resto', 'IPC'])

# # drop delle colonne che presentano tutti valori NaN
# raw_data = raw_data.dropna(axis='columns', how ='all')

# #inserisco NaN nelle celle dove non c'è (magari c'è uno spazio...)
# # raw_data = raw_data.fillna(0)

# raw_data.head(30)
```

[ ]:
```
conteggio = raw_data.value_counts('secondo')
conteggio
```

[ ]:
```
secondo
 18/00           2
 9/00            2
 1/19            2
 7/10            2
 3/00            2
 7/00            2
 19/04           2
 1/02            1
 43/16           1
 48/00-48/30     1
 5/00            1
 5/40-5/48       1
 5/44            1
 5/48            1
 50/00-50/171    1
 51/50           1
 51/42-51/48     1
 31/00-31/078    1
 53/04           1
 53/22           1
 6/20            1
 6/30            1
 69/00           1
 7/04            1
 8/00-8/24       1
 35/02           1
 3/28            1
```

```
 3/46            1
 3/35            1
 1/182           1
 1/32            1
 1/41            1
 1/74-1/80       1
 10/06           1
 10/54           1
 101/00          1
 11/04           1
 11/14           1
 13/00-13/04     1
 13/18           1
 16/24           1
 19/30           1
 20/02           1
 21/14           1
 23/04           1
 25/03           1
 27/02           1
 1/15            1
 9/52            1
dtype: int64
```

[ ]:
```
conteggio = raw_data.value_counts('sesto')
conteggio
```

[ ]:
```
sesto
 53/24    1
dtype: int64
```

[ ]:
```
raw_data.to_csv('test_raw_data.csv', sep =';', index=False)
```

Codici singoli

[ ]:
```
# if raw_data['primo'] == np.nan & raw_data['secondo'] == np.nan &
 ↪raw_data['terzo'] == np.nan:
#     raw_data['codice'] == raw_data['radice']
# else:
condition = [raw_data['primo'].isnull()]
choice = [raw_data['radice']]

df_codici = pd.DataFrame()
df_codici['codici'] = ''
df_codici

df_codici['codici'] = np.select(condition, choice, default=0)
```

```
#drop rows that contain any value in the list
values = [0]
df_codici = df_codici[df_codici['codici'].isin(values) == False]

# Creo colonna binaria true e false per evidenziare i codici che devono essere␣
 ↪"esplosi"
df_codici['to_explode'] = df_codici['codici'].apply(lambda x: 'True' if "-" in␣
 ↪x else 'False')

print(df_codici)
df_codici.shape
```

```
    codici to_explode
6      C10G      False
18     A01H      False
27     C10J      False
45     B09B      False
51     F03B      False
52     F03C      False
57     F03D      False
65     F24S      False
66     H02S      False
81     F24S      False
102    F24T      False
103    F01K      False
148    B62K      False
150     B61      False
163    H02J      False
166    G01R      False
188    B09B      False
189    B65F      False
194    B09C      False
196    F23G      False
201    C05F      False
247    C02F      False
254    E03F      False
261    C05F      False
262    G06Q      False
263    G08G      False
264    G06Q      False
266     G21      False
267    G21B      False
268    G21C      False
269    G21D      False
```

[ ]: (31, 2)

Codici con primo riferimento valorizzato

[ ]:
```
condition = [raw_data['primo'].notnull()]
choice = [raw_data['radice'] + ' ' + raw_data['primo']]
# choice = [raw_data['radice'] + raw_data['primo']]

df_codici_1 = pd.DataFrame()
df_codici_1['codici'] = ''
df_codici_1

df_codici_1['codici'] = np.select(condition, choice, default=0)

#drop rows that contain any value in the list
values = [0]
df_codici_1 = df_codici_1[df_codici_1['codici'].isin(values) == False]

# Creo colonna binaria true e false per evidenziare i codici che devono essere␣
 ↪"esplosi"
df_codici_1['to_explode'] = df_codici_1['codici'].apply(lambda x: 'True' if "-"␣
 ↪in x else 'False')

print(df_codici_1.tail(10))
```

```
              codici to_explode
252         E03C 1/12      False
253         C02F 1/00      False
255        G21C 13/10      False
256        A01G 23/00      False
257        A01G 25/00      False
258  A01N 25/00-65/00       True
259        C09K 17/00      False
260         E02D 3/00      False
265         E04H 1/00      False
270         F02C 1/05      False
```

Codici con secondo riferimento valorizzato

[ ]:
```
condition = [raw_data['secondo'].notnull()]
# choice = [raw_data['radice'] + ' ' + raw_data['secondo']]
choice = [raw_data['radice'] + raw_data['secondo']]

df_codici_2 = pd.DataFrame()
df_codici_2['codici'] = ''
df_codici_2

df_codici_2['codici'] = np.select(condition, choice, default=0)
```

```
#drop rows that contain any value in the list
values = [0]
df_codici_2 = df_codici_2[df_codici_2['codici'].isin(values) == False]

# Creo colonna binaria true e false per evidenziare i codici che devono essere␣
 ↪"esplosi"
df_codici_2['to_explode'] = df_codici_2['codici'].apply(lambda x: 'True' if "-"␣
 ↪in x else 'False')

print(df_codici_2.shape)
df_codici_2.head(10)
```

```
(56, 2)
```

[ ]:

	codici	to_explode
0	C10L 5/40-5/48	True
2	C10L 9/00	False
3	C10L 1/02	False
4	C10L 1/19	False
5	C07C 69/00	False
7	C10L 1/19	False
10	C10L 1/182	False
13	C02F 11/04	False
17	C12N 1/15	False
21	H01M 8/00-8/24	True

Codici con terzo riferimento valorizzato

[ ]:
```
condition = [raw_data['terzo'].notnull()]
# choice = [raw_data['radice'] + ' ' + raw_data['terzo']]
choice = [raw_data['radice'] + raw_data['terzo']]

df_codici_3 = pd.DataFrame()
df_codici_3['codici'] = ''
df_codici_3

df_codici_3['codici'] = np.select(condition, choice, default=0)

#drop rows that contain any value in the list
values = [0]
df_codici_3 = df_codici_3[df_codici_3['codici'].isin(values) == False]

# Creo colonna binaria true e false per evidenziare i codici che devono essere␣
 ↪"esplosi"
df_codici_3['to_explode'] = df_codici_3['codici'].apply(lambda x: 'True' if "-"␣
 ↪in x else 'False')
```

```
df_codici_3
```

[ ]:
```
              codici to_explode
3          C10L 1/14      False
17         C12N 1/21      False
21   H01M 12/00-12/08       True
46       B01D 53/047      False
71         H01L 25/16      False
83         F24D 11/00      False
85         F03D 13/20      False
149         B62M 5/00      False
162        H02J 15/00      False
174         E04B 1/88      False
175  E04C 2/284-2/296       True
182        E04D 13/16      False
214        C22B 25/06      False
219        B01D 53/62      False
253         C02F 9/00      False
```

Codici con quarto riferimento valorizzato

[ ]:
```
condition = [raw_data['quarto'].notnull()]
# choice = [raw_data['radice'] + ' ' + raw_data['quarto']]
choice = [raw_data['radice'] + raw_data['quarto']]

df_codici_4 = pd.DataFrame()
df_codici_4['codici'] = ''
df_codici_4

df_codici_4['codici'] = np.select(condition, choice, default=0)

#drop rows that contain any value in the list
values = [0]
df_codici_4 = df_codici_4[df_codici_4['codici'].isin(values) == False]

# Creo colonna binaria true e false per evidenziare i codici che devono essere␣
 ↪"esplosi"
df_codici_4['to_explode'] = df_codici_4['codici'].apply(lambda x: 'True' if "-"␣
 ↪in x else 'False')

df_codici_4
```

[ ]:
```
         codici to_explode
17    C12N 5/10      False
46   B01D 53/14      False
71   H01L 25/18      False
```

```
83   F24D 19/00      False
149   B62M 6/00      False
174   E04B 1/90      False
```

Codici con quinto riferimento valorizzato

[ ]:
```
condition = [raw_data['quinto'].notnull()]
# choice = [raw_data['radice'] + ' ' + raw_data['quinto']]
choice = [raw_data['radice'] + raw_data['quinto']]

df_codici_5 = pd.DataFrame()
df_codici_5['codici'] = ''
df_codici_5

df_codici_5['codici'] = np.select(condition, choice, default=0)

#drop rows that contain any value in the list
values = [0]
df_codici_5 = df_codici_5[df_codici_5['codici'].isin(values) == False]

# Creo colonna binaria true e false per evidenziare i codici che devono essere␣
 ↪"esplosi"
df_codici_5['to_explode'] = df_codici_5['codici'].apply(lambda x: 'True' if "-"␣
 ↪in x else 'False')

df_codici_5
```

[ ]:
```
         codici to_explode
17   C12N 15/00      False
46   B01D 53/22      False
71  H01L 31/042      False
```

Codici con sesto riferimento valorizzato

[ ]:
```
condition = [raw_data['sesto'].notnull()]
# choice = [raw_data['radice'] + ' ' + raw_data['sesto']]
choice = [raw_data['radice'] + ' ' + raw_data['sesto']]

df_codici_6 = pd.DataFrame()
df_codici_6['codici'] = ''
df_codici_6

df_codici_6['codici'] = np.select(condition, choice, default=0)

#drop rows that contain any value in the list
values = [0]
df_codici_6 = df_codici_6[df_codici_6['codici'].isin(values) == False]
```

```
# Creo colonna binaria true e false per evidenziare i codici che devono essere
 ↪"esplosi"
df_codici_6['to_explode'] = df_codici_6['codici'].apply(lambda x: 'True' if "-"
 ↪in x else 'False')

df_codici_6
```

[ ]:
```
        codici to_explode
46  B01D  53/24      False
```

Append di tutti i dataframe

[ ]:
```
codici_tutti = df_codici.append([df_codici_1, df_codici_2, df_codici_3,
 ↪df_codici_4, df_codici_5, df_codici_6])

# cancello i doppioni
codici_no_duplicati = codici_tutti.drop_duplicates()

print(f'Il dataframe con tutti i codici con doppioni ha grandezza:
 ↪{codici_tutti.shape}')
print(f'Il dataframe senza i codici doppi ha grandezza: {codici_no_duplicati.
 ↪shape}')
```

```
Il dataframe con tutti i codici con doppioni ha grandezza: (352, 2)
Il dataframe senza i codici doppi ha grandezza: (311, 2)
```

[ ]:
```
codici_no_duplicati.to_csv('test1.csv', sep =';', encoding='utf-8-sig',
 ↪index=False)
```

Creo una colonna origine e una destinazione per i codici valorizzati true

[ ]:
```
# creo due dataframe:
# 1 - df con i codici "sistemati", quindi senza il trattino ('to_explode =
 ↪False')
# 2 - df con i codici "da esplodere", quindi con il trattino ('to_explode =
 ↪True')
# Si potrà lavorare solamente sul secondo df e poi unire il primo con il
 ↪risultato del secondo

conteggio = codici_no_duplicati.value_counts('to_explode')
print(conteggio)

filtro_false = codici_no_duplicati['to_explode'] == 'False'
codici_false = codici_no_duplicati[filtro_false].copy()

filtro_true = codici_no_duplicati['to_explode'] == 'True'
codici_true = codici_no_duplicati[filtro_true].copy()
```

```
print(f'Il dataframe con tutti i codici false ha grandezza: {codici_false.
 ↪shape}')
print(f'Il dataframe senza i codici true ha grandezza: {codici_true.shape}')
```

```
to_explode
False    272
True      39
dtype: int64
Il dataframe con tutti i codici false ha grandezza: (272, 2)
Il dataframe senza i codici true ha grandezza: (39, 2)
```

[ ]:
```
# lavoro sul dataframe con i soli codici da esplodere
codici_true['origine'] = codici_true['codici'].str.split('-').str[0]
codici_true['radice'] = codici_true['origine'].str.split(' ').str[0]

codici_true['destinazione'] = codici_true['radice'] + ' ' +␣
 ↪codici_true['codici'].str.split('-').str[1]

# codici_true['origine'] = codici_true['num'].str.split('-').str[0]
# codici_true['destinazione'] = codici_true['num'].str.split('-').str[1]

codici_true.tail(10)
```

[ ]:

	codici	to_explode	origine	radice	destinazione
235	B01D 45/00-51/00	True	B01D 45/00	B01D	B01D 51/00
258	A01N 25/00-65/00	True	A01N 25/00	A01N	A01N 65/00
0	C10L 5/40-5/48	True	C10L 5/40	C10L	C10L 5/48
24	H01M 50/00-50/171	True	H01M 50/00	H01M	H01M 50/171
67	H01L 31/00-31/078	True	H01L 31/00	H01L	H01L 31/078
70	H01L 51/42-51/48	True	H01L 51/42	H01L	H01L 51/48
135	F16H 48/00-48/30	True	F16H 48/00	F16H	F16H 48/30
174	E04B 1/74-1/80	True	E04B 1/74	E04B	E04B 1/80
204	C11B 13/00-13/04	True	C11B 13/00	C11B	C11B 13/04
175	E04C 2/284-2/296	True	E04C 2/284	E04C	E04C 2/296

[ ]:
```
codici_true.to_csv('test2.csv', sep =';', index=False)
```

[ ]:
```
codici_true = codici_true.drop(columns=['radice'])
codici_true.shape
```

[ ]:
```
(39, 4)
```

[ ]:
```
# Append dei due dataframe
codici_false['origine'] = 'null'
codici_false['destinazione'] = 'null'
```

```
codici_finale = codici_true.append([codici_false])

print(codici_finale.head(5))
print(codici_finale.tail(5))
```

```
             codici to_explode     origine destinazione
12    C12P 7/06-7/14       True   C12P 7/06    C12P 7/14
21    H01M 4/86-4/98       True   H01M 4/86    H01M 4/98
24    H01M 8/00-8/24       True   H01M 8/00    H01M 8/24
25  H01M 12/00-12/08       True  H01M 12/00   H01M 12/08
49    E02B 9/00-9/06       True   E02B 9/00    E02B 9/06
         codici to_explode origine destinazione
149    B62M 6/00      False    null         null
174    E04B 1/90      False    null         null
17    C12N 15/00      False    null         null
71   H01L 31/042      False    null         null
46   B01D  53/24      False    null         null
```

[ ]:
```
print(codici_finale.iloc[33:42])
```

```
               codici to_explode     origine destinazione
67   H01L 31/00-31/078       True  H01L 31/00  H01L 31/078
70    H01L 51/42-51/48       True  H01L 51/42   H01L 51/48
135   F16H 48/00-48/30       True  F16H 48/00   F16H 48/30
174     E04B 1/74-1/80       True   E04B 1/74    E04B 1/80
204   C11B 13/00-13/04       True  C11B 13/00   C11B 13/04
175   E04C 2/284-2/296       True  E04C 2/284   E04C 2/296
6                 C10G      False        null         null
18                A01H      False        null         null
27                C10J      False        null         null
```

[ ]:
```
codici_finale.to_csv('green_inventory_codes.csv', sep =';',␣
 ↪encoding='utf-8-sig', doublequote= False, index=False)
```