LAB2-PYTHON

Maria Peregrina Usieto – MPEREGRINA

2021-2022

MARCH 2022

INDEX

PART 2 3
1. What is the purpose of the data set 3
2. Which is the information you have on each row of the data set 3
3. How can the dataset be exported 3
4. All the formats that the dataset can be exported 4
5. How the information is represented in the exported file 4
6. Which is the number of records 5
PART 3 7
1. Informatión about the environment 7
1.1 Files that are in the repository 7
1.2 Prepare the virtual environment of Python 7
1.3 How to execute the code 7
2. Information about the packages of the lab 8
3. Main code of the program 9
4. OPTION 1 12
5. OPTION 2 13
6. OPTION 3 15
7. OPTION 4 17
8. OPTION 5 21
9. OPTION 6 23
10. OPTION 7 26
11. OPTION 8 29
12.OPTION 9 32
13.OPTION 10 34

REPORT

PART 2

1. What is the purpose of the data set

This set of data stores information about the sports records of Catalonia. The purpose is to have all the sports entities registered with the location and the type of modality that can be practiced

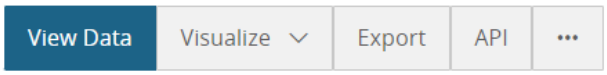
2. Which is the information you have on each row of the data set

In total we have 10 different columns:

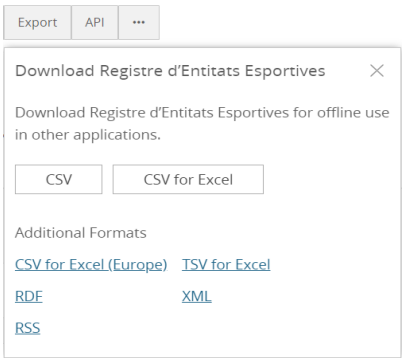
- Núm_registre → Contains the row identifier for the sports record
- Nom_entidad → It contains the full name of the entity, in this case the entity name can also be an identifier since they are not repeated.
- Adreça → Address of the Sports Center (street and street number)
- CP → Postal Code of the Sports Center (of the entity)
- Municipi → Municipality of the Sports Center (of the entity)
- Comarca → Comarca of the Sports Center (of the entity)
- Telèfon_Fax → Contact Fax Phone
- Correu Electrònic → Contact E-Mail
- Tipus_entitat → Type of entity of the entity
- Modalitat → List of sports that can be practiced in this entity.

3. How can the dataset be exported

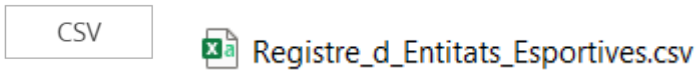
To export the first data from everything we will go to the website where they are (https://analisi.transparenciacatalunya.cat/en/Esport/Registre-d-Entitats-Esportives/qrgc-u7pk) . On the page, we can see the following box where we give us the option to export the data:



if we click on the option Export, we will give us the different options of extensions in which we can export the data.



In our case we will export the data with the CSV extension:



4. All the formats that the dataset can be exported

CSV: (comma separated values) Is a text file that has a specific format that allows data to be stored in a structured table format.
CSV for Excel:(comma separated values) file is a special type of file that you can create or edit in Excel.
TSV for Excel: It is used exclusively by spreadsheet software. The values are tab delimited.
RDF: It is a document file used to define data related to resources on the web.
RSS: (Really Simple Syndication or Rich Site Summary), is an XML data structuring format that facilitates automated access to the information contained in an Internet site.
XML: Is the acronym for Extensible Markup Language, i.e., it is a markup language that defines a set of rules for encoding documents.

5. How the information is represented in the exported file

Once we export the file, we look at the data in three different ways.
- From Excel → We open the data from Excel by importing the CSV file, in this way we could view the data in a very visual way.

NÚM_REGISTRE	NOM_ENTITAT	ADREÇA	CP	MUNICIPI	COMAR
A00001	ASSOCIACIO CENTRE CULTURAL EL SOCIAL	Font Vella, 40	8221	Terrassa	Vallès C
A00002	AGRUPACIO CULTURAL I ESPORTIVA DELS MEMBRES COS BOMBERS DE BARCELONA	Provença, 178	8029	Barcelona	Barcelo
A00003	SOCIETAT ESPORTIVA L'ALZINAR	PL.DE L'ESTACIO, S/N	8783	Masquefa	Anoia
A00004	ASSOCIACIO ESPORTIVA CENTRE CATOLIC DE MATARO	La Riera, 110	8301	Mataró	Maresm
A00005	ASSOCIACIO PROPIETARIS I VEINS MAS GUINARDO	Salvador Riera, 11	8026	Barcelona	Barcelo
A00006	AGRUPACIO SFERIC	Faradai, 41	8224	Terrassa	Vallès C
A00007	ASSOCIACIO DE GENT GRAN L'ESPLAI GRANOLLERS	Roger de Flor, 68-B 4t. 3a.	8400	Granollers	Vallès C
A00008	SECCIO BILLAR CERCLE DE CATOLICS DE BANYOLES	ABEURADOR, 12	17820	Banyoles	Pla de l'
A00009	CENTRO DEPORTIVO RECREATIVO CULTURAL EL CERCLE	URB.TARRUELLA, EDIF.EMBERS, 2N.1R.	43891	Vandellòs i l'Hospitalet de l'Infant	Baix Ca
A00010	ZONA RECREATIVA BARCINOVA	Dr. Pi i Molist, 117 bis	8031	Barcelona	Barcelo
A00011	CASAL CATOLIC DE SANT ANDREU DE PALOMAR SECCIÓ ESCACS	Pons i Gallarza, 58-60	8030	Barcelona	Barcelo
A00012	CASINO DE LA GARRIGA	de la Garriga, 95	8530	Garriga, la	Vallès C

- With Pluma in the VM → We also access the data from the pen editing tool Pluma in the Virtual Machine. This way was a bit more difficult to visualize the data since they are not separated by columns, the data is separated by commas.

- Print data with Python → We use Python code to print in terminal data from the exported file. We can see that in this way it is not very comfortable to visualize the data.

{'NÚM_REGISTRE': '16687', 'NOM_ENTITAT': 'CLUB ESPORTIU MASQUEFA', 'ADREÇA': 'Sant Antoni, 6-8', 'CP': '08783',
'MUNICIPI': 'Masquefa', 'COMARCA': 'Anoia', 'TELÈFON_FAX': '/', 'CORREU_ELECTRÒNIC': '', 'TIPUS_ENTITAT': 'Club
i Associacions Esportives', 'MODALITATS': 'Atletisme, Ciclisme, Futbol'}
{'NÚM_REGISTRE': '16688', 'NOM_ENTITAT': 'CLUB FUTBOL SALA CASTELLAR', 'ADREÇA': 'Joaquim Blume, s/n', 'CP': '0
211', 'MUNICIPI': 'Castellar del Vallès', 'COMARCA': 'Vallès Occidental', 'TELÈFON_FAX': '/', 'CORREU_ELECTRÒNI
': '', 'TIPUS_ENTITAT': 'Clubs i Associacions Esportives', 'MODALITATS': 'Futbol sala (Futbol cinc), Voleibol'}
{'NÚM_REGISTRE': '16689', 'NOM_ENTITAT': 'CLUB DE PESCA ESPORTIVA GAVA', 'ADREÇA': 'Santa Creu de Calafell, 127
, 'CP': '08850', 'MUNICIPI': 'Gavà', 'COMARCA': 'Baix Llobregat', 'TELÈFON_FAX': '936625668 /', 'CORREU_ELECTRÒ
IC': 'spegava@gmail.com', 'TIPUS_ENTITAT': 'Clubs i Associacions Esportives', 'MODALITATS': 'Pesca Esportiva, P
sca continental, Pesca marítima'}
{'NÚM_REGISTRE': '16690', 'NOM_ENTITAT': "CLUB D'ACTIVITATS SUBAQUATIQUES BARCELONA SUD", 'ADREÇA': 'Castanyer,
1', 'CP': '08860', 'MUNICIPI': 'Castelldefels', 'COMARCA': 'Baix Llobregat', 'TELÈFON_FAX': '627981461 /', 'COR
EU_ELECTRÒNIC': 'casbarcelonasud@gmail.com', 'TIPUS_ENTITAT': 'Clubs i Associacions Esportives', 'MODALITATS':
Escafandrisme'}
{'NÚM_REGISTRE': '16691', 'NOM_ENTITAT': 'CLUB HIPICO TOMAS SANTIAGO', 'ADREÇA': 'Camí de les Debeses, 6', 'CP'
'43391', 'MUNICIPI': 'Vinyols i els Arcs', 'COMARCA': 'Baix Camp', 'TELÈFON_FAX': '670488794 /', 'CORREU_ELECT
ÒNIC': 'tssvtc2013@gmasil.com', 'TIPUS_ENTITAT': 'Clubs i Associacions Esportives', 'MODALITATS': "Concurs comp
et d'equitació, Doma clásica, Hípica, Salts d'obstacle"}

Visualizing the data We realized that there were incorrect rows, there were rows with the zip code in the spine of the region, others did not contain any modality ... To solve it, we have removed different rows, and to do so we have reopened the data in the editor of the text pluma.

6. Which is the number of records

CSV → To open this file, we just have to open an Excel document, we will go to the data section and we will select the option to obtain data from a CSV file. If we go at the end of the document, we can see that the last row is in the position 20147:

20147

CSV for Excel → We can open this document in the same way we have opened the previous file. In this case when we go at the end of the document, the last row is in the position 20146.

20146

TSV for Excel → We have opened this document with the notepad (Bloc de notas). If we go at the end of the document and position the pointer in the last row, we can see that we are on the line 20146.

Línea 20146, columna 1

RDF → We have opened this document with the notepad (Bloc de notas), but we can open it with different texting tools (also the previous extension). If we go at the end of the document and position the pointer in the last row, we can see that we are on the line 6023.

Línea 6023, columna 5

But if we look at the data, we can see that for each data row, we have 12 lines:

```
<dsbase:qrgc-u7pk rdf:about="https://analisi.transparenciacatalunya.cat/resource/qrgc-u7pk
    <socrata:rowID>row-mvkj~z6ep~xjnx</socrata:rowID>
    <rdfs:member rdf:resource="https://analisi.transparenciacatalunya.cat/resource/qrgc-u7
    <ds:n_m_registre>A00508</ds:n_m_registre>
    <ds:nom_entitat>APA COL·LEGI MARE DE DEU DEL ROSER</ds:nom_entitat>
    <ds:adre_a>Pare Coll, 1-3</ds:adre_a>
    <ds:cp>08600</ds:cp>
    <ds:municipi>Berga</ds:municipi>
    <ds:comarca>Berguedà</ds:comarca>
    <ds:tel_fon_fax>/</ds:tel_fon_fax>
    <ds:tipus_entitat>AMPA</ds:tipus_entitat>
    <ds:modalitats>Atletisme, Basquetbol, Futbol, Handbol, Patinatge sobre rodes</ds:modal
```

To calculate the number of rows in the documents, we will divide the number of lines by the 12 lines that occupies a row. Approximately 502 rows.

6023/12 = 502

RSS → When we have gone to download this file format, the data is opened in the browser. To be able to open the file from an application, what we have done is copy the URL to the Firefox browser and ask us what application we want to open it. We have decided to open it with the notebook. In this case we have 1601 lines, but each row occupies about 32 lines. If we do the same operation as in the previous section the result is 4. Therefore, there are approximately 50 rows

Línea 1601, columna 1

1601/32 = 50

XML → This file format also opens with the browser, but unlike the previous section, this format we have not been able to open it from any application. Therefore, to know how many records there are, let's go at the end of the page and we will look for the CSV files on which line is this row. This is the last row of the document:

```
▼<row _id="row-9uqy_hxzz~k2pk" _uuid="00000000-0000-0000-A39F-1EB57E
   <n_m_registre>05044</n_m_registre>
   <nom_entitat>SOCIETAT DE CAÇADORS ""SOBREPUNY""</nom_entitat>
   <adre_a>c/ Xalets, 23</adre_a>
   <cp>08613</cp>
   <municipi>Vilada</municipi>
   <comarca>Berguedà</comarca>
   <tel_fon_fax>/</tel_fon_fax>
   <correu_electr_nic>sobrepuny@gmail.com</correu_electr_nic>
   <tipus_entitat>Clubs i Associacions Esportives</tipus_entitat>
   <modalitats>Caça, Caça fotogràfica (Caça), Coloms a braç (Caça), C
   Umbert (Caça)</modalitats>
 </row>
```

Let's look for the record number 05044. We see that it is on line 19405

19405	05044

PART 3

1. Informatión about the environment

1.1 Files that are in the repository

```
maria@labvm:~/m/lab2-python-python2_mariaperegrina_alexsamper$ ls
 LAB2-PYTHON.pdf  'Registre_d_Entitats_Esportives (1).csv'   requirements.txt
  script.py   shp
```

- 'Registre_d_Entitats_Esportives (1).csv' → File with all records data from sports entities.
- requirements.txt → File with packages needed to execute the code.
- script.py → The code
- shp → Folder that contains the files with the geometric data of Catalunya

1.2 Prepare the virtual environment of Python

In order to work with the script.py first of all we have to prepare the environment, for this we will follow the following steps:

- We enter the following directory

```
maria@labvm:~$ cd labs/devnet-src/python/
maria@labvm:~/labs/devnet-src/python$
```

- We create a virtual environment of Python 3

```
maria@labvm:~/labs/devnet-src/python$ python3 -m venv lab2-maria
maria@labvm:~/labs/devnet-src/python$
```

- We activate the virtual environment that we have just created

```
maria@labvm:~/labs/devnet-src/python$ source lab2-maria/bin/activate
(lab2-maria) maria@labvm:~/labs/devnet-src/python$
```

- Now let's install the necessary packets. For this we are going to go first to the directory where we have all the repository files. Then, we install the packages that are in the **requirements.txt** file as follows.

```
Collecting affine==2.3.0
  Using cached affine-2.3.0-py2.py3-none-any.whl (15 kB)
Collecting attrs==21.4.0
  Using cached attrs-21.4.0-py2.py3-none-any.whl (60 kB)
```

Now we have the environment prepared to be able to execute the script.

1.3 How to execute the code

In order to run the code, we have to be in the directory where all the files showed in section 1.1 and we have to prepare step 1.2.
Once we are prepared, the only thing we have to do is execute the command **python script.py**.

```
List of options:
--------------

1. Table with the names of provinces and the number of entities
```

2. Information about the packages of the lab

To see the packages installed in the virtual environment in which we are working we are going to use the **pip3 freeze** command.

```
(lab2-maria) maria@labvm:~/
affine==2.3.0
attrs==21.4.0
branca==0.4.2
certifi==2021.10.8
charset-normalizer==2.0.12
click==8.0.4
click-plugins==1.1.1
cligj==0.7.2
contextily==1.2.0
cycler==0.11.0
Fiona==1.8.21
folium==0.12.1.post1
fonttools==4.30.0
geographiclib==1.52
geopandas==0.10.2
geopy==2.2.0
idna==3.3
Jinja2==3.0.3
joblib==1.1.0
kiwisolver==1.4.0
MarkupSafe==2.1.1
matplotlib==3.5.1
mercantile==1.2.1
munch==2.5.0
numpy==1.22.3
packaging==21.3
pandas==1.4.1
Pillow==9.0.1
pyparsing==3.0.7
pyproj==3.3.0
python-dateutil==2.8.2
pytz==2021.3
rasterio==1.2.10
requests==2.27.1
Shapely==1.8.1.post1
six==1.16.0
snuggs==1.4.7
urllib3==1.26.9
xyzservices==2022.3.0
```

We can see that the list of packages is very long, but we have only installed 5 packages (the others have been automatically installed).

We are going to explain so that we have used the different packages we have installed:

- Folium → We are using this package for the creation of maps (and to add the pointers) in options 4 and 5.

- Geopandas → This package is used mainly to open the .shp file and to generate the merge.
- Geopy → From this library we use nominatim to achieve the exact coordinates of the addresses of the entities
- Matplotlib → We use this package for the creation and visualization of the maps in the options 6, 7, 8 and 9. We also use it to represent the Histogram of Option 10.
- Contextily → Library that allows us to import base maps from different providers.

3. Main code of the program

Before everything we will explain the main code. All the execution of the program is within a loop that is controlled from the variable “next” initialized to True. We are going to explain that it makes the code for every return it gives. First of all, print the list of options there. To print the options we have preferred to create a function called “printOptions()” then we will show you the function. Then we expect the user to indicate one of the options, we add the code to make sure that the value that the user introduces is correct.

```
#main
next = True
while (next):
    print("List of options: \n-------------\n ")
    printOptions()
    option = int(input())
    while(option<1 or option>10):
        print("Incorrect option, the correct values are [1-10].\n")
        option = int(input())
```

Here we can see the printOption() function that we have talked about.

```
def printOptions():
    print("1. Table with the names of provinces and the number of entities\n")
    print("2. Table with the modalities and the number of entities\n")
    print("3. Modalities of the 'COMARCA' chosen\n")
    print("4. Map of the entity that you choose\n")
    print("5. Map with the entities of a postal code\n")
    print("6. Color map of entities per 'comarca'. The TOP10 'comarcas'\n")
    print("7. Color map of entities per 'comarca'. The DOWN10 'comarcas'\n")
    print("8. Color map of entities per 'comarca'. Without BArcelonés\n")
    print("9. Color map of entities per 'comarca'.\n")
    print("10. Histograma of entities per Comarca\n")
```

Next, according to the option that the user chooses, the corresponding function is executed. We have created each option in a different function with the name FunctionNumoption ().

```
print("\n")
print("Your choice is the option", option, ":\n")
if option == 1:
    function1()
elif option == 2:
    function2()
elif option == 3:
    function3()
elif option == 4:
    function4()
elif option == 5:
    function5()
elif option == 6:
    function6()
elif option == 7:
    function7()
elif option == 8:
    function8()
elif option == 9:
    function9()
elif option == 10:
    function10()
```

Finally, when the function has been executed, we ask the user if he wants to execute another option or if he wants to finish the program. If the user introduces the character 'and' then the program will continue and ask you another option, but if you enter any other character the program ends.

```
    print("\n")
    print("Do you want to run another option? [y, n]")
    o = input()
    if o.lower() != 'y':
        next = False

print("The program has ended\n")
```

Last but not least, we show you the path variable, which defines the name of the file with the necessary data. As we are working on the same directory where the file is, it is only necessary to put the name of the file.

```
path = 'Registre_d_Entitats_Esportives (1).csv'
```

We are going to teach you the execution of the main code, first we execute the script **script.py**.

```
List of options:
-------------

1. Table with the names of provinces and the number of entities

2. Table with the modalities and the number of entities

3. Modalities of the 'COMARCA' chosen

4. Map of the entity that you choose

5. Map with the entities of a postal code

6. Color map of entities per 'comarca'. The TOP10 'comarcas'

7. Color map of entities per 'comarca'. The DOWN10 'comarcas'

8. Color map of entities per 'comarca'. Without BArcelonés

9. Color map of entities per 'comarca'.

10. Histograma of entities per Comarca
```

A listing of us appears with the options we have, now it is to choose one of the options, for example 1. But if we put a incorrect value,we ask us to return to a value within the range [1-10].

```
1


Your choice is the option 1 :

PROVINCES      ENTITIES
-------------------------
BARCELONA:       10654
GIRONA:          2192
LLEIDA:          1874
TARRAGONA:       2267


Do you want to run another option? [y, n]
```

```
15
Incorrect option, the correct values are [1-10].
```

We can see that the function1 has been executed and then asked us if we want to continue, let's see the response options:

- Enter Y

```
Do you want to run another option? [y, n]
Y
List of options:
--------------

1. Table with the names of provinces and the

2. Table with the modalities and the number
```

- Enter n

```
Do you want to run another option? [y, n]
n
The program has ended
```

- Enter some character

```
Do you want to run another option? [y, n]
M
The program has ended
```

4. OPTION 1

The option 1 wants a table with the names of provinces and for each of them the number of entities on the file.
First of all we have prepared what is necessary for the code. We have opened the corresponding file and have read it. For this we have imported the csv libraries and codecs. There are more ways to open the document, but we have finally decided by the DictreAder () function. Then we have prepared a variable initialized to 0 for each province (Barcelona, Girona, Lleida and Tarragona).

```
def function1():
    file = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
    reader = csv.DictReader(file)
    b=0
    l=0
    g=0
    t=0
```

Now we start a tour for each row of the Reader file. The objective is to get the number of entities for each province, in this case, we know that the number of entities is equivalent to the number of rows, therefore we are going to tell the times the province appears. We know that the provinces don't appear in the file. To know each entity in which province is, we have looked at the postal code.

- For Barcelona, the postal code starts at '08'
- For Lleida, the postal code starts at '25'
- For Girona, the postal code starts at '17'
- For Tarragona, the postal code starts at '43'

Then we take the first two digits of the postal code of the row and look at what province it is. With this "row['CP']" we have said that we want us from the value of postal code

```
for row in reader:
    cp = row['CP']
    if (cp[0:2] == '08'):
        b +=1
    elif (cp[0:2] == '25'):
        l +=1
    elif (cp[0:2] == '17'):
        g +=1
    elif (cp[0:2] == '43'):
        t +=1
```

Once the tour ends, we can already print the result:

```
print("PROVINCES      ENTITIES")
print("-------------------------")
print("BARCELONA:      ",b)
print("GIRONA:         ",g)
print("LLEIDA:         ",l)
print("TARRAGONA:      ",t)
```

Next we can see the result of the execution:

```
Your choice is the option 1 :

PROVINCES      ENTITIES
-------------------------
BARCELONA:       10654
GIRONA:          2192
LLEIDA:          1874
TARRAGONA:       2267
```

5. OPTION 2

This option wants to print the different modalities that there are, with the number of entities in which you can practice and order from top to down.
For this, we have first opened the file and read it. Also we create a array of Defaultdict types. This format is very comfortable because we indexed by the name of the modality and the value will be the counter.

```
def function2():
    file = codecs.open(path, encoding='utf-8', errors='backslashreplace')
    reader = csv.DictReader(file)
    total = defaultdict(int)
```

Now we travel through the file (row per row). From each row we recover the modalities (we keep it in the “mod” variable), but as there are entities that may have more than one modality, what we have done has been separating the different modalities that it has with the split() function and separating by the character ",". Now we have the

different modalities saved in the “sep” variable and we take a loop of this variable to keep it in our variable result "Total". We can see that we take the value of the “sep[i]” component and we increment 1. In this way, if the modality had already been added, it will be increased, else will be added and initialized to 1.

```
for row in reader:
    mod = row['MODALITATS']
    sep = mod.split(', ')
    for i in range(len(sep)):
        total[sep[i]] +=1
```

Once we travel, we have all the necessary information, but we will miss order it. This is what we are going to do. For this we have made two simple loops and we have been saving the data ordered in the “order” variable.

```
#Order  TOP DOWN
order = defaultdict(int)
for i in range(len(total)):
    aux = 0
    for m, cont in total.items():
        if(aux < cont and cont != 0):
            f_count = cont
            f_mod = m
            aux = cont
    order[f_mod]= f_count
    total[f_mod]=0
```

We see that the code is very simple, the only thing we do is look every time the modality that most appears and then matched it to 0 so as not to take it again. Finally, we printed the values ordered:

```
for letra, cuenta in order.items():
    print(letra, cuenta)
```

As there are many types of modalities, we can not see them all but we will show different parts of the solution:

```
Curses d'orientació 141
Piragüisme 135
Esports aeris 134
Billar 131
Bitlles catalanes 131
Rugby 126
Kick boxing 122
Ciclisme en carretera 121
Futbol Set 117
```

```
Esports de trineu 10
Tenis platja 10
Acrobàcia 10
Caça submarina  (act subaquàtiques) 10
Escacs (sords) 10
Petanca (sords) 10
Eslalom (automobilisme) 10
Tresc (Trekking) 9
Boccia Paralitics Cerebrals 9
```

```
Pista 1
Lacrosse 1
Floorball 1
Bike Polo 1
Futbolgolf 1
Hidrotrineu (Hidrospeed) 1
Tanguilla 1
Parapent d'arrossegament 1
Toc-Bol 1
```

6. OPTION 3

The option 1 wants the user to select a province and a COMARCA of the selected province. For the selected COMARCA print the different modalities with the number of entities that appear. The first thing we do is stop prepared the file read to use it later.

```
def function3():
    file = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
    reader = csv.DictReader(file)
```

Now let's print the provinces so that the user chooses one. To work with the provinces, what we are going to do is save the value of the first two digits of the postal code in the “cp” variable. We can see that if the user introduces an incorrect value, the program will end.

```
print("Select one option: \n")
print("1.Barcelona\n2.Girona\n3.Lleida\n4.Tarragona")
option = input()
if int(option) == 1:
    cp = '08'
elif int(option) == 2:
    cp = '17'
elif int(option) == 3:
    cp = '25'
elif int(option) == 4:
    cp = '43'
else:
    print("WRONG VALUE. THE PROGRAM ENDS")
    quit()
```

As of the chosen province, we will print the COMARCAS and let's ask you to choose one.

To make the process easier, we will save the COMARCAS at the array “list” to control the repetition and be able to easily recover the user's choice.

We can see that we make a tour of the file, in the first IF they will enter all the rows that are from the province chosen by the user and in the second IF all the COMARCAS that have not been added in the list even. Then we add the region to the list and print the index and the COMARCA so that the user has a list.

Finally we collect the user's choice and recover the value int the list of value with the index “User Option -1”

```
print("Select one option: \n")
list = []
cont = 1
for row in reader:
    if row['CP'][0:2] == cp:
        if(row['COMARCA'] not in list):
            list.append(row['COMARCA'])
            print(cont,". ",row['COMARCA'])
            cont += 1
option2 = input()
c = list[int(option2)-1]
```

Now we will have to make another loop for the file, but for this we will have to read it again since it's empty.

```
file = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
reader = csv.DictReader(file)
```

Now we have recovered the code we have seen in the previous section to achieve the different modalities with the number of entities in which it appears. In this case we add the condition of the COMARCA.

```
total = defaultdict(int)
for row1 in reader:
    if row1['COMARCA'] == c:
        mod = row1['MODALITATS']
        sep = mod.split(', ')
        for i in range(len(sep)):
            total[sep[i]] +=1
```

Finally we print the result of the operation:

```
for moda, valor in total.items():
    print(moda, valor)
print("\n")
```

Now we are going to show you the result of the execution:

```
Your choice is the option 3 :

Select one option:

1.Barcelona
2.Girona
3.Lleida
4.Tarragona
```

For example we can choose the option 1

```
1
Select one option:

1 .  Vallès Occidental
2 .  Barcelonès
3 .  Anoia
4 .  Maresme
5 .  Vallès Oriental
6 .  Garraf
7 .  Osona
8 .  Alt Penedès
9 .  Baix Llobregat
10 .  Bages
11 .  Berguedà
12 .  Moianès
13 .  Selva
14 .  Garrigues
```

We can see that now it makes us choose a region, let's choose option 8. This is the result:

```
8
Atletisme 26
Basquetbol 38
Beisbol 3
Futbol 67
Futbol sala (Futbol cinc) 29
Handbol 13
Voleibol 19
Ciclisme 23
Natació 11
[. . .]
Telemark 1
Salts d'obstacle 1
Power Lifting 1
Boxa 2
Skate-Board 1
Stand Up Paddle 1
Surf 1
```

7. OPTION 4

The objective of the option is shown a MAP to identify the location of one entity that the user chose. For this the user needs to choose first a COMARCA and a MUNICIPI. First of all we have read the file as in the other sections.

```
def function4():
    file = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
    reader = csv.DictReader(file)
```

Then we have used code of the previous option for the user to select a region. We can see that the code is the same but now we do not have restrictions.

```
list_com = []
cont = 1
for row in reader:
    if(row['COMARCA'] not in list_com):
        list_com.append(row['COMARCA'])
        print(cont,". ",row['COMARCA'])
        cont += 1
option = input()
c = list_com[int(option)-1]
```

Then we will read the file again because we will have to make a similar loop so that the user chooses a municipality.

```
read = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
reader = csv.DictReader(read)
```

The code is very similar, we add the restriction of the region and now what we printed are the municipalities. Finally, we expect the user to choose one.

```
list_mun = []
cont = 1
for row in reader:
    if(row['COMARCA']==c):
        if(row['MUNICIPI'] not in list_mun):
            list_mun.append(row['MUNICIPI'])
            print(cont,". ",row['MUNICIPI'])
            cont += 1
option2 = input()
m = list_mun[int(option2)-1]
print(m)
```

Once the user chooses a municipality, we are going to reread the file to be able to make a new tour.

```
read = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
reader = csv.DictReader(read)
```

The user has already indicated a region, of that region we have printed all the municipalities and the user has chosen one. Now from the municipality you have chosen, we have to print the entities and the user should choose one. The tour is the same, the only variations are the columns we work with.

```
entidades = []
cont = 1
for row in reader:
    if(row['MUNICIPI']==m):
        if(row['NOM_ENTITAT'] not in entidades):
            entidades.append(row['NOM_ENTITAT'])
            print(cont,". ",row['NOM_ENTITAT'])
            cont += 1
option2 = input()
e = entidades[int(option2)-1]
```

We read the file again, this is the last time we have to read it.

```
read = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
reader = csv.DictReader(read)
```

This last tour about the file we do it to recover the address of the entity that the user has chosen.

```
for row in reader:
    if(row['NOM_ENTITAT']==e):
        dir = row['ADREÇA']
```

Now we have the necessary data to create the map. The first thing we do is put together the address, the municipality and add "spain" to have a more complete direction.

```
l = dir.capitalize()+", "+m+", Spain"
```

For the creation of the map we have imported Nominatim of Geopy and Folium. Nominatim uses OpenStreetMap data to find locations on Earth by name and address (geocoding) and Nominatim must need a user_agent, we use this to achieve the coordinates of the entity. In the “loc” variable we keep the exact coordinates that we have achieved thanks to the function geocode() of the Nominatim variable.
Then what we do is create the map from the acquired coordinates and also indicate a right zoom. Now the last thing left is to create the pointer to view the exact location on the map, this we get it with the function Marker() indicating the coordinates, and we add the pointer to the map we have created. This map can be visualized in the terminal or in the browser. We have decided to show it in the browser, for this we have saved the map as an HTML type file with the name of mapa4.html.

```
geo = Nominatim(user_agent="agent")
loc = geo.geocode(l)
mapa = folium.Map(location=[loc.latitude, loc.longitude], zoom_start=15)
folium.Marker([loc.latitude, loc.longitude], tooltip= l).add_to(mapa)
mapa.save('mapa4.html')
```

Next we can see the result of the execution. (The combination of options we will use will be 2-5-3):

```
Your choice is the option 4 :

1 .  Vallès Occidental
2 .  Barcelonès
3 .  Anoia
4 .  Maresme
5 .  Vallès Oriental
6 .  Pla de l'Estany
7 .  Baix Camp
8 .  Garraf
9 .  Osona
10 .  Alt Penedès
11 .  Baix Llobregat
```

[...]

```
35 .  Baix Empordà
36 .  Pallars Jussà
37 .  Pallars Sobirà
38 .  Aran
39 .  Alt Urgell
40 .  Cerdanya
41 .  Solsonès
42 .  Moianès
2
```

We have selected Option 2 of Comarca, now let's choose a municity:

```
1 .  Barcelona
2 .  Sant Adrià de Besòs
3 .  Santa Coloma de Gramenet
4 .  Hospitalet de Llobregat, l'
5 .  Badalona
5
Badalona
```

We can see that our choice has been Badalona. This municipality has a total 341 entities, we are going to choose the option 3

```
1 .  APA COLEGIO JESUS MARIA DE BADALONA
2 .  CIRCULO CATOLICO DE BADALONA
3 .  AAVV BACHS-BUFALA
4 .  APA COOPERATIVA ENSEÑANZA GARCIA MARQUEZ
5 .  APA CP PAU PICASSO
6 .  APA ESCOLA JUNGFRAU
7 .  APA ESCOLES BAETULO
8 .  CENTRO CULTURAL CALDERON
9 .  APA DEL COLEGIO SANTISIMA TRINIDAD
```

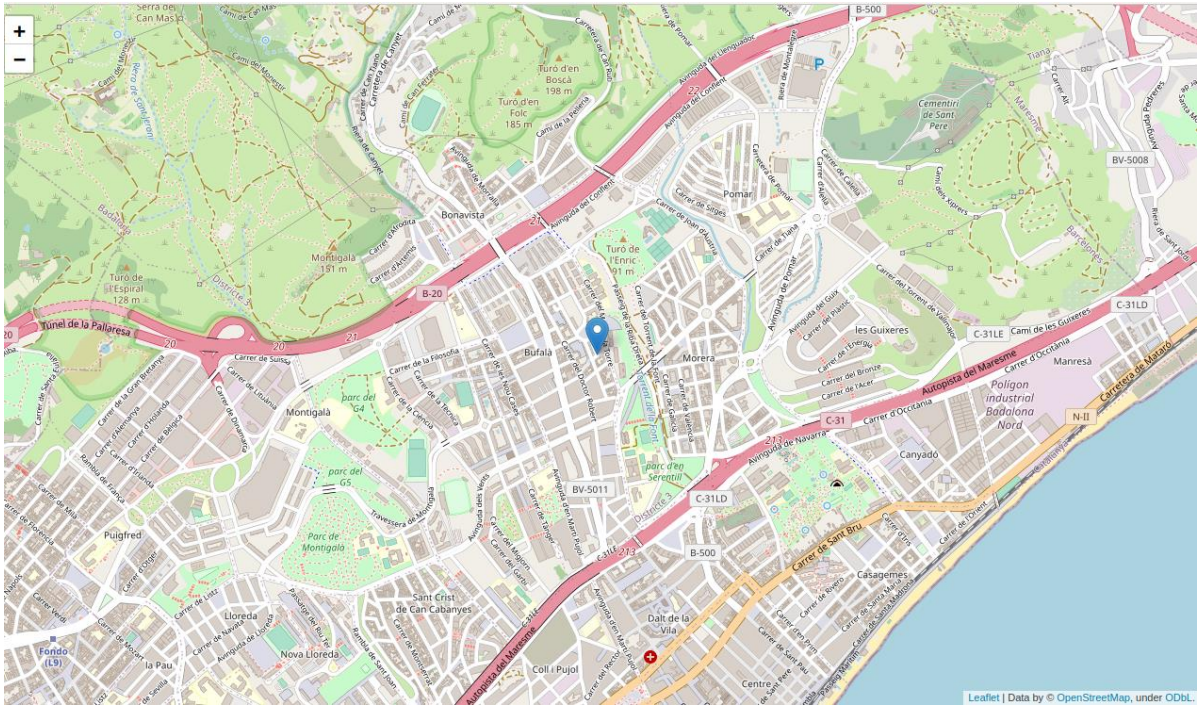
```
338 .  CLUB ESPORTIU TAPIOL TEAM
339 .  CLUB TWIRLING CIUTAT DE BADALONA
340 .  CLUB CICLES LA SALUT TRIATLO
341 .  BADALONA PADEL CLUB
3
```

We can see that now the function is over, and that you are asking us if we want to execute another option. As we have decided to see the map by the browser, we have to open it manually.

If we look at what we have in the directory, we can see that the map4.html file is located.

```
mapa4.html
```

Now let's open the file to see the result:



8. OPTION 5

This option is similar to the previous one, but now the user indicated a postal code and we will show a map with all the entities that are in that postal code. This option starts as the other, first we read the file.

```
def function5():
    file = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
    reader = csv.DictReader(file)
```

Next we tell the user to introduce a postal code.

```
print("Enter a postal code\n")
cp = input()
```

Once we have the postal code, we save in a list all the addresses that are inside this postal code. We can see that it is a loop similar to those we have seen and the format of the direction is the same as in the previous section.

```
list_dir = []
for row in reader:
    if(row['CP']==cp):
        l = row['ADRECA'].capitalize()+", "+row['MUNICIPI']+", Spain"
        if(l not in list_dir):
            list_dir.append(l)
```

Now we can start creating the map. First of all, we take the first address of the list to create the map base, for this we use exactly the same as in the previous section. From the address we have in the first position of the list, we take the coordinate and the coordinates we created the map.

```
aux = list_dir[0]
geo = Nominatim(user_agent="agent")
loc_i = geo.geocode(aux)
mapa = folium.Map(location=[loc_i.latitude, loc_i.longitude], zoom_start=14)
```

Once created now we will lack add the pointers with the exact location of each entity. For this we have made a loop by the list of addresses. And for each direction we have recovered the coordinates and we have added them as a pointer in the same way as in the previous section. We have put the code within a TRY..CATCH because when we look for coordinates, not all directions are apt for this. In this way we will put the pointers of all directions that the coordinates can get

```
for l in range(len(list_dir)):
    try:
        loc = geo.geocode(list_dir[l])
        folium.Marker([loc.latitude, loc.longitude], tooltip= l).add_to(mapa)
    except Exception:
        print(Exception)
```

Finally, we keep the map as HTML so we can open it from a browser.

```
mapa.save('mapa5.html')
```

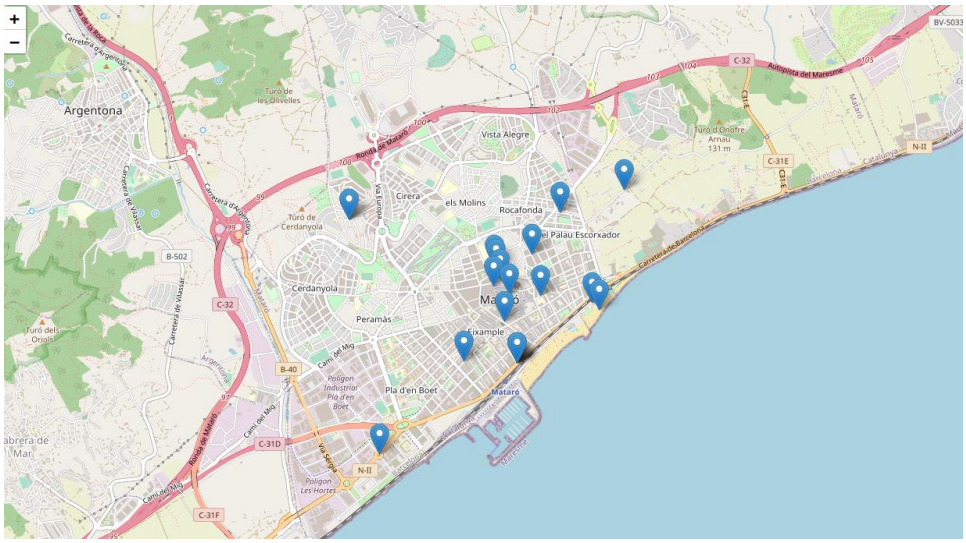
Now we are going to show you the execution, for the test we will use the postal code 08301.

```
Your choice is the option 5 :

FUNCION PARA OPCION 5
Enter a postal code

08301
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
<class 'Exception'>
```

The exceptions that give us are normal, is what we have commented before. We can see that the map has been generated:

`mapa5.html`

Now let's see the result of the map opening the file map5.html.



9. OPTION 6

The objective of this option is to show a MAP with the TOP10 countries per number of entities on a color map.
First of all we define a function6(), inside that function we read the file.

```
def function6():
    read = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
    reader = csv.DictReader(read)
```

Next we have the code where we save in the array top[] the top 10 counties. In this code there is a loop where what it does is to compare the number of entities that has a county, in this case as we want the counties with more entities, we are looking at which county has more and we are saving it in the array, once we have the 10 leaves the for.

```
comarcas = defaultdict(int)
for row in reader:
    comarcas[row['COMARCA']] += 1

top = defaultdict(int)
for i in range(10):
    aux =0
    aux_t = defaultdict(int)
    for c, v in comarcas.items():
        if v >= aux:
            if v == aux:
                aux_t[c]=v
            else:
                aux = v
                aux_t = defaultdict(int)
                aux_t[c]=v
    for cj, vj in aux_t.items():
        comarcas[cj] = 0
        top[cj] = vj
```

Now we create the file TOP.csv where we save the TOP 10 counties with more entities. We have assigned the name 'NOMCOMAR' to the column where are the names of the counties and the name 'NUM_ENTITATS' for the column where are the numbers of entities of each county.

```
with open('TOP.csv', 'w') as file_top:
    filenames = ['NOMCOMAR', 'NUM_ENTITATS']
    write = csv.DictWriter(file_top, fieldnames=filenames)

    write.writeheader()
    for name, num in top.items():
        write.writerow({'NOMCOMAR':name, 'NUM_ENTITATS':num})
```

Next we proceed to open and read the csv file called TOP, we also open and read the .shp file where is located the geometry of all the counties of Catalonia.

```
file_path = 'TOP.csv'
df = pd.read_csv(file_path)
#we read the file
shp_path = "./shp/divisions-administratives-v2r0-comarques-1000000-20210701.shp"
#We use utf-8 encoding to recognize accents and special characters in the shp file such as the letters "Ñ"
sf = gpd.read_file(shp_path,encoding = 'utf-8')
```

To finish we make a merge between the two files, this what it does is join the two files by joining a column, in this case the same column is 'NOMCOMAR'.
We transform the coordinates to epsg 3857 which is the one used by most of the maps represented on the web.

Once the merge is done we proceed to configure the window where the drawing will be displayed, we make the window of 16x16 pixels and we deactivate the axes of the sides.
In the 'fig' is where we edit the representation of the map and assign colors depending on the column 'NUM_ENTITATS', which is where we have the number of entities that are in each county, that we do with cmap in this case we chose ‘Set2’ we also set the contour, opacity etc..
Finally we add the base map in this case from 'ctx.providers.OpenStreetMap.Mapnik' and display the created map.

```
merged = pd.merge(df, sf)
merged = gpd.GeoDataFrame(merged)
merged = merged.to_crs(epsg=3857)

fig, ax = plt.subplots(1, figsize=(16, 16))

ax.axis('off')

ax.set_title('CATALUNYA',fontdict={'fontsize': '24', 'fontweight': '3'})

#We use the pandas plot function to select the column we want to plot and the formatting characteristics on the map.

fig=merged.plot(column='NUM_ENTITATS', cmap='Set2', alpha=0.8,zorder=1,edgecolor='green',linewidth=0.1, ax = ax,legend=True)
ctx.add_basemap(ax,source=ctx.providers.OpenStreetMap.Mapnik)

plt.show()
```

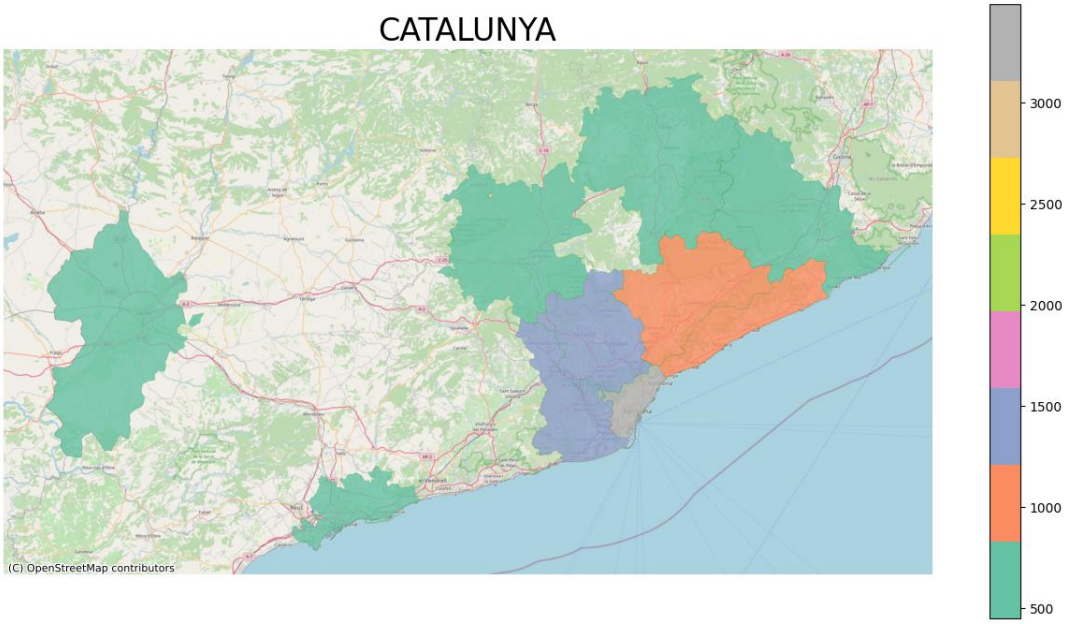
Now we are going to show you the execution, we selected option 6.

```
6. Color map of entities per 'comarca'. The TOP10 'comarcas'
```

The program gives us a confirmation of the option we have chosen while we wait for the map to be generated.

```
6


Your choice is the option 6 :
```

Now let's see the result of the mapa.



When we close the map the program asks us if we want to run another option.

```
Do you want to run another option? [y, n]
n
The program has ended
```

10. OPTION 7

The objective of this option is to show a MAP with the LESS10 counties per number of entities on a color map.
First of all we define a function7(), inside that function we read the file.

```
def function7():
    read = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
    reader = csv.DictReader(read)
```

Next we have the code where we save in the array down[] the top less 10 counties. In this code there is a loop where what it does is to compare the number of entities that has a county, in this case as we want the counties with less entities, we are looking at which county has less and we are saving it in the array, once we have the 10 leaves the for.

```
comarcas = defaultdict(int)
for row in reader:
    comarcas[row['COMARCA']] += 1


down = defaultdict(int)
for i in range(10):
    aux =10000
    aux_t = defaultdict(int)
    for c, v in comarcas.items():
        if v < aux:
            aux = v
            aux_t = defaultdict(int)
            aux_t[c]=v
    for cj, vj in aux_t.items():
        comarcas[cj] = 10000
        down[cj] = vj
```

Now we create the file LOW.csv where we save the TOP 10 counties with less entities. We have assigned the name 'NOMCOMAR' to the column where are the names of the counties and the name 'NUM_ENTITATS' for the column where are the numbers of entities of each county.

```
with open('LOW.csv', 'w') as file_top:
    filenames = ['NOMCOMAR', 'NUM_ENTITATS']
    write = csv.DictWriter(file_top, fieldnames=filenames)

    write.writeheader()
    for name, num in down.items():
        write.writerow({'NOMCOMAR':name, 'NUM_ENTITATS':num})
```

Next we proceed to open and read the csv file called LOW, we also open and read the .shp file where is located the geometry of all the counties of Catalonia.
We make a merge between the two files, this what it does is join the two files by joining a column, in this case the same column is 'NOMCOMAR'.
We transform the coordinates to epsg 3857 which is the one used by most of the maps represented on the web.

```
file_path = 'LOW.csv'
df = pd.read_csv(file_path)
shp_path = "./shp/divisions-administratives-v2r0-comarques-1000000-20210701.shp"
sf = gpd.read_file(shp_path,encoding = 'utf-8')

merged = pd.merge(df, sf)
merged = gpd.GeoDataFrame(merged)
merged = merged.to_crs(epsg=3857)
```

Once the merge is done we proceed to configure the window where the drawing will be displayed, we make the window of 16x16 pixels and we deactivate the axes of the sides.
In the 'fig' is where we edit the representation of the map and assign colors depending on the column 'NUM_ENTITATS', which is where we have the number of entities that are in each county, that we do with cmap in this case we chose ‘Set2’ we also set the contour, opacity etc..
Finally we add the base map in this case from 'ctx.providers.OpenStreetMap.Mapnik' and display the created map.

```
fig, ax = plt.subplots(1, figsize=(16, 16))

ax.axis('off')

ax.set_title('CATALUNYA',fontdict={'fontsize': '24', 'fontweight': '3'})

fig=merged.plot(column='NUM_ENTITATS', cmap='Set2', alpha=0.8,zorder=1,edgecolor='green',linewidth=0.1, ax = ax,legend=True)
ctx.add_basemap(ax,source=ctx.providers.OpenStreetMap.Mapnik)
plt.show()
```

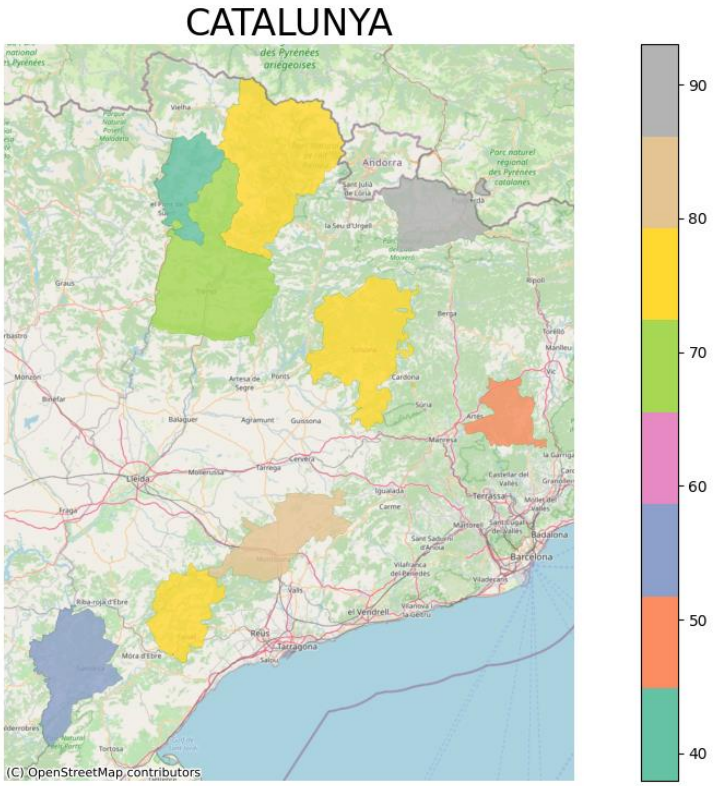
Now we are going to show you the execution, we selected option 7.

```
7. Color map of entities per 'comarca'. The DOWN10 'comarcas'
```

The program gives us a confirmation of the option we have chosen while we wait for the map to be generated.

```
7


Your choice is the option 7 :
```

Now let's see the result of the mapa.



When we close the map the program asks us if we want to run another option.

```
Do you want to run another option? [y, n]
y
```

11. OPTION 8

The objective of this option is to show a MAP with all counties except Barcelonès on a color map.
First of all we define a function8(), inside that function we read the file.

```
def function8():
    read = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
    reader = csv.DictReader(read)
```

Next we have the code where we save in the array comarcas[] all counties except ‘Barcelonès’, in this code there is a loop where what it does is to compare if the county is different to ‘Barcelonès’ if this condition is right, the county will be add to the array comarcas.
Now we create the file NOBAR.csv where we save all counties except Barcelonès.
We have assigned the name 'NOMCOMAR' to the column where are the names of the counties and the name 'NUM_ENTITATS' for the column where are the numbers of entities of each county

```
comarcas = defaultdict(int)
for row in reader:
    if(row['COMARCA']!='Barcelonès'):
        comarcas[row['COMARCA']] += 1

with open('NOBAR.csv', 'w') as file_top:
    filenames = ['NOMCOMAR', 'NUM_ENTITATS']
    write = csv.DictWriter(file_top, fieldnames=filenames)

    write.writeheader()
    for name, num in comarcas.items():
        write.writerow({'NOMCOMAR':name, 'NUM_ENTITATS':num})
```

Next we proceed to open and read the csv file called NOBAR, we also open and read the .shp file where is located the geometry of all the counties of Catalonia.
We make a merge between the two files, this what it does is join the two files by joining a column, in this case the same column is 'NOMCOMAR'.
We transform the coordinates to epsg 3857 which is the one used by most of the maps represented on the web.

```
file_path = 'NOBAR.csv'
df = pd.read_csv(file_path)
shp_path = "./shp/divisions-administratives-v2r0-comarques-1000000-20210701.shp"
sf = gpd.read_file(shp_path,encoding = 'utf-8')

merged = pd.merge(df, sf)
merged = gpd.GeoDataFrame(merged)
merged = merged.to_crs(epsg=3857)
```

Once the merge is done we proceed to configure the window where the drawing will be displayed, we make the window of 16x16 pixels and we deactivate the axes of the sides.
In the 'fig' is where we edit the representation of the map and assign colors depending on the column 'NUM_ENTITATS', which is where we have the number of entities that are in each county, that we do with cmap in this case we chose ‘plasma_r’ we also set the contour, opacity etc..
Finally we add the base map in this case from 'ctx.providers.OpenStreetMap.Mapnik' and display the created map.

```
fig, ax = plt.subplots(1, figsize=(16, 16))

ax.axis('off')

ax.set_title('CATALUNYA',fontdict={'fontsize': '24', 'fontweight': '3'})

fig=merged.plot(column='NUM_ENTITATS', cmap='plasma_r', alpha=0.8,zorder=1,edgecolor='green',linewidth=0.1, ax = ax,legend=True)
ctx.add_basemap(ax,source=ctx.providers.OpenStreetMap.Mapnik)

plt.show()
```

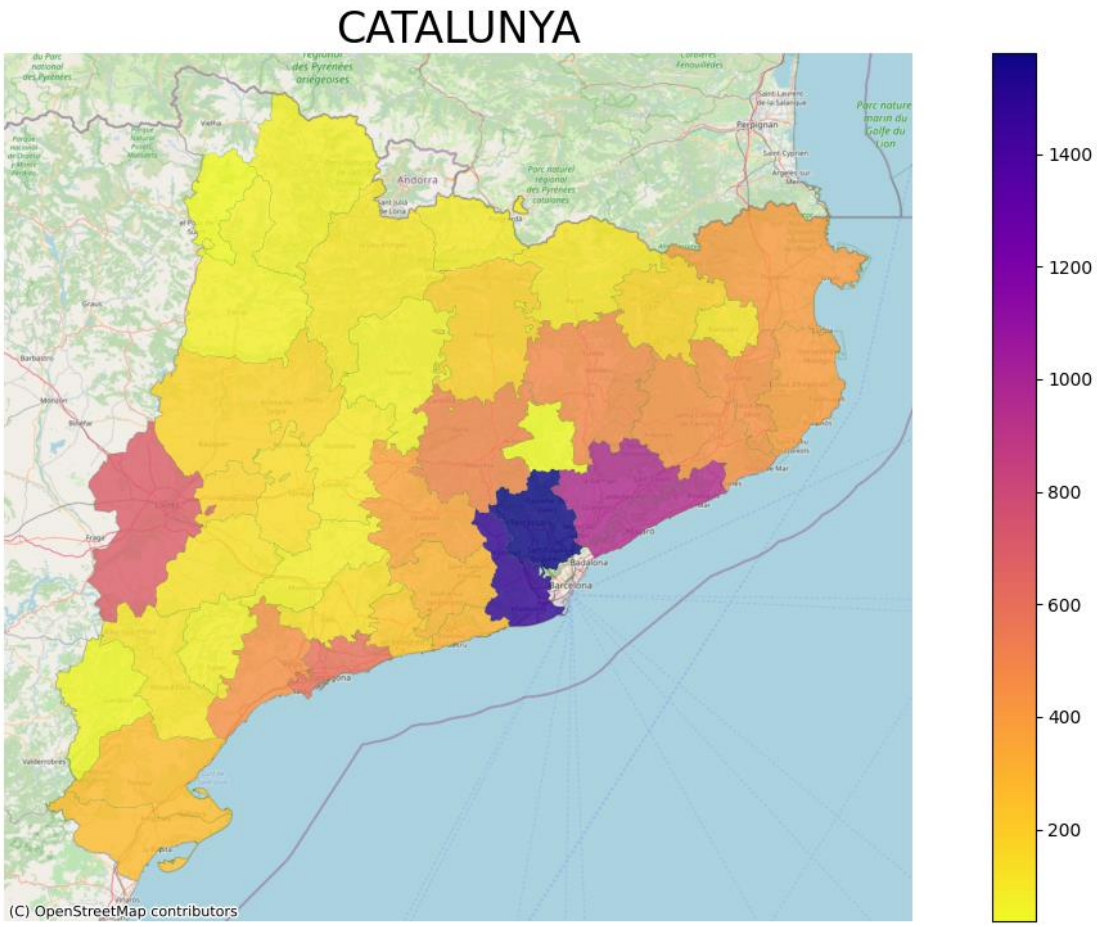
Now we are going to show you the execution, we selected option 8.

```
8. Color map of entities per 'comarca'. Without BArcelonés
```

The program gives us a confirmation of the option we have chosen while we wait for the map to be generated.

```
8


Your choice is the option 8 :
```

Now let's see the result of the mapa.



When we close the map the program asks us if we want to run another option.

```
Do you want to run another option? [y, n]
y
```

12.OPTION 9

The objective of this option is to show a MAP with all counties on a color map.
First of all we define a function9(), inside that function we read the file.

```
def function9():
    read = codecs.open(path, 'r', encoding='utf-8', errors='backslashreplace')
    reader = csv.DictReader(read)
```

Next we have the code where we save in the array comarcas[] all counties, in this code there is a loop where what it does is add to the array comarcas all counties.
Now we create the file ALL.csv where we save all counties.
We have assigned the name 'NOMCOMAR' to the column where are the names of the counties and the name 'NUM_ENTITATS' for the column where are the numbers of entities of each county.

```
comarcas = defaultdict(int)
for row in reader:
    comarcas[row['COMARCA']] += 1


with open('ALL.csv', 'w') as file_top:
    filenames = ['NOMCOMAR', 'NUM_ENTITATS']
    write = csv.DictWriter(file_top, fieldnames=filenames)

    write.writeheader()
    for name, num in comarcas.items():
        write.writerow({'NOMCOMAR':name, 'NUM_ENTITATS':num})
```

Next we proceed to open and read the csv file called ALL, we also open and read the .shp file where is located the geometry of all the counties of Catalonia.
We make a merge between the two files, this what it does is join the two files by joining a column, in this case the same column is 'NOMCOMAR'.
We transform the coordinates to epsg 3857 which is the one used by most of the maps represented on the web.

```
file_path = 'ALL.csv'
df = pd.read_csv(file_path)
shp_path = "./shp/divisions-administratives-v2r0-comarques-1000000-20210701.shp"
sf = gpd.read_file(shp_path,encoding = 'utf-8')

merged = pd.merge(df, sf)
merged = gpd.GeoDataFrame(merged)
merged = merged.to_crs(epsg=3857)
```

Once the merge is done we proceed to configure the window where the drawing will be displayed, we make the window of 16x16 pixels and we deactivate the axes of the sides.

In the 'fig' is where we edit the representation of the map and assign colors depending on the column 'NUM_ENTITATS', which is where we have the number of entities that are in each county, that we do with cmap in this case we chose ‘inferno_r’ we also set the contour, opacity etc..
Finally we add the base map in this case from 'ctx.providers.OpenStreetMap.Mapnik' and display the created map.

```
fig, ax = plt.subplots(1, figsize=(16, 16))

ax.axis('off')

ax.set_title('CATALUNYA',fontdict={'fontsize': '24', 'fontweight': '3'})

fig=merged.plot(column='NUM_ENTITATS', cmap='inferno_r', alpha=0.8,zorder=1,edgecolor='green',linewidth=0.1, ax = ax,legend=True)
ctx.add_basemap(ax,source=ctx.providers.OpenStreetMap.Mapnik)

plt.show()
```

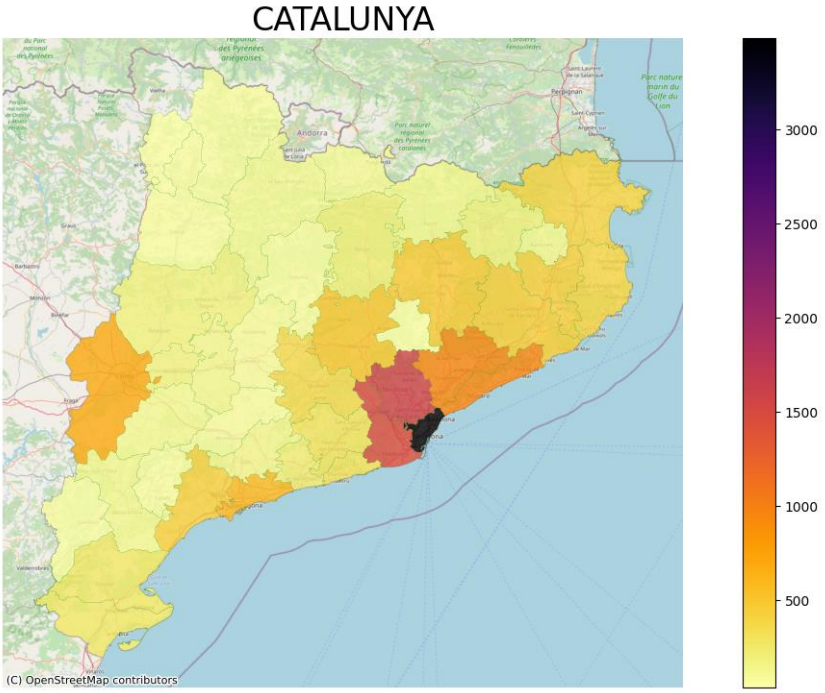
Now we are going to show you the execution, we selected option 9.

```
9. Color map of entities per 'comarca'.
```

The program gives us a confirmation of the option we have chosen while we wait for the map to be generated.

```
9


Your choice is the option 9 :
```

Now let's see the result of the mapa.



When we close the map the program asks us if we want to run another option.

```
Do you want to run another option? [y, n]
n
The program has ended
```

13.OPTION 10

The objective of this option is to show a histogram of the number of entities per Comarca.
First of all we define a function10(), inside that function we read the file ALL.csv. We have to keep in mind that in order to open the file ALL.csv, first we have to run option 9, which is where the file is created.
Inside this file there are all the counties of Catalonia.

```
def function10():

    file_path = 'ALL.csv'
    df = pd.read_csv(file_path)
```

In this fragment you can see that what is done is to relate 'NOMCOMAR' with 'NUM_ENTITATS', in this way we can configure the histogram correctly.
We place 'NOMCOMAR' on the X-axis and 'NUM_ENTITATS' on the y-axis, then we set the histogram size and display it on the screen.

```
    ff=pd.DataFrame(df,columns=["NOMCOMAR","NUM_ENTITATS"])

    ff.plot(x="NOMCOMAR",y="NUM_ENTITATS",kind="bar",stacked=True,figsize=(10,10))
    plt.legend(loc="lower left",bbox_to_anchor=(0.8,1.0))
    plt.show()
```

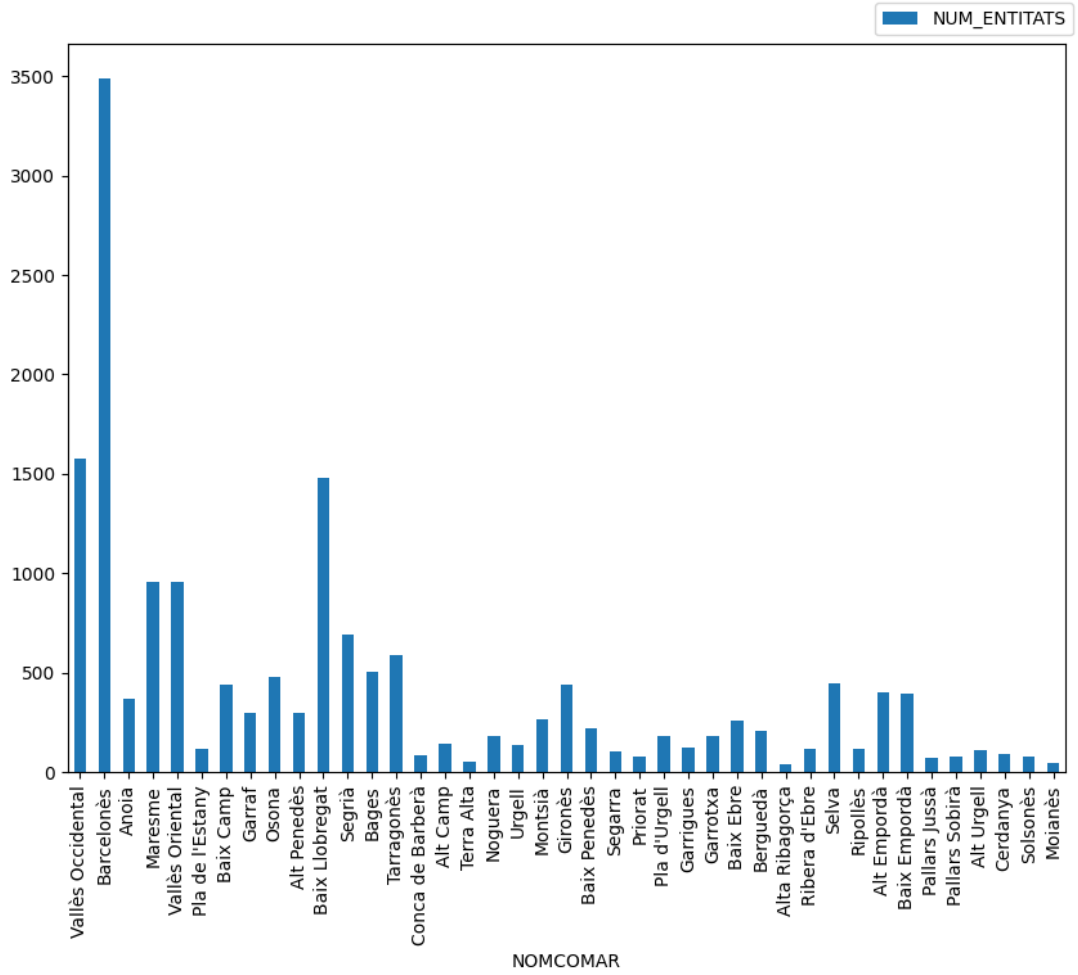
Now we are going to show you the execution, we selected option 10.

```
10. Histograma of entities per Comarca
```

The program gives us a confirmation of the option we have chosen while we wait for the map to be generated.

```
10


Your choice is the option 10 :
```

Now let's see the result of the histogram.



When we close the map the program asks us if we want to run another option.

```
Do you want to run another option? [y, n]
n
The program has ended
```