

LAB2_PYTHON



Sergio Sánchez González

github: Sergio9322

Gerard Torrent Castell

github: Gtorrentc96

Group 101

Team K

Deliver: 11 march 2022

For Tecnocampus

Index

Part 2: Selected Data Set	-----	2
Part 3: Process selected Data Set	-----	4
How the environment has been saved	-----	15
How the environment must be imported	-----	16
Packages used on the lab	-----	17
Execution of the program	-----	18

Part 2: Selected Data Set

1.- What is the purpose of the data set

The objective of this dataset is the inscription of sport entities that are in Catalonia.

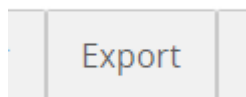
2.- Which is the information you have on each row of the data set

We have the following information per row:

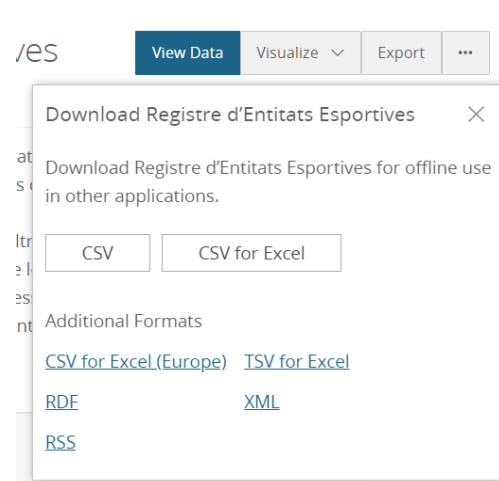
- NÚM_REGISTRE: Number of inscription to the register
- NOM_ENTITAT: Name of the sports entity
- ADREÇA: Address of the entity
- CP: Postal code
- MUNICIPI: Municipality
- COMARCA: District
- TELÈFON_FAX: The contact number of the entity
- CORREU_ELECTRÒNIC: The email of the entity
- TIPUS_ENTITAT: Entity type
- MODALITATS: Different modality of sports

3.- How can the dataset be exported. List of the formats and explain all the formats the dataset can be exported to. How the information is represented in the exported file format. Which is the number of records of each exporting option.

There's a button in the dataset page called Export



When you press it, it opens a sub-menú where you can pick which format you prefer



- A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format. A CSV file typically stores tabular data (numbers and text) in plain text, in which case each line will have the same number of fields.
- A tab-separated values (TSV) file is a simple text format for storing data in a tabular structure, e.g., database table or spreadsheet data, and a way of exchanging information between databases. Each record in the table is one line of the text file. Each field value of a record is separated from the next by a tab character. The TSV format is thus a type of the more general comma-separated values format.
- The Resource Description Framework (RDF) is a World Wide Web Consortium (W3C) standard originally designed as a data model for metadata. It has come to be used as a general method for description and exchange of graph data. RDF provides a variety of syntax notations and data serialization formats with Turtle (Terse RDF Triple Language) currently being the most widely used notation.
- Extensible Markup Language (XML) is a markup language and file format for storing, transmitting, and reconstructing arbitrary data. It defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
- RSS (RDF Site Summary or Really Simple Syndication) is a web feed^[3] that allows users and applications to access updates to websites in a standardized, computer-readable format. Subscribing to RSS feeds can allow a user to keep track of many different websites in a single news aggregator, which constantly monitors sites for new content, removing the need for the user to manually check them. News aggregators (or "RSS readers") can be built into a browser, installed on a desktop computer, or installed on a mobile device.

Part 3: Process selected Data Set

1.- Print a table with the name of PROVINCIAS (and for them the number of items on the file. Number of ENTITATS. Not ROWS).

```
def exercisel():
    with open('file.csv', newline='') as csvfile:
        dialect = csv.Sniffer().sniff(csvfile.read(1024), delimiters=',')
        csvfile.seek(0)
        reader = csv.reader(csvfile, dialect)
        reader = csv.DictReader(csvfile)
        barcelon anum=0
        lleidanum=0
        tarragon anum=0
        giron anum=0
        for row in reader:
            if row['COMARCA'] in barcelonaCom:
                barcelon anum=barcelon anum+1
            elif row['COMARCA'] in lleidaCom:
                lleidanum=lleidanum+1
            elif row['COMARCA'] in tarragonaCom:
                tarragon anum=tarragon anum+1
            elif row['COMARCA'] in gironaCom:
                giron anum=giron anum+1
        print("En barcelona hi ha "+str(barcelon anum)+" , en lleida hi ha "+str(lleidanum)+
              " , en tarragona hi ha "+str(tarragon anum)+" , en girona hi ha "+str(giron anum))
```

In exercise 1 we open the file and we loop through all the rows in the file, then check for each row (Which contains only 1 entity) and we seek if it's comarca is in a list we have created for each provincia, if it's in the list then we add 1 to the counter of each provincia.

```
barcelonaCom=["Alt Penedès", "Anoia", "Bages", "Baix Llobregat", "Barcelonès", "Berguedà", "Maresme", "Garra
lleidaCom=["Alta Ribagorça", "Alt Urgell", "Cerdanya", "Garrigues", "Noguera", "Pallars Jussà", "Pallars So
tarragonaCom=["Alt Camp", "Baix Camp", "Baix Ebre", "Baix Penedès", "Conca de Barberà", "Conca de Barber", "Gi
gironaCom=["Alt Empordà", "Baix Empordà", "Garrotxa", "Gironès", "Pla de l'Estany", "Selva", "Ripollès", "Gi
```

Then we print the result of the counters.

2.- Print a table with the MODALITATS and for each one the number of entities, order TOP DOWN

```
def exercise2():
    with open('file.csv', newline='') as csvfile:
        dialect = csv.Sniffer().sniff(csvfile.read(1024), delimiters=',')
        csvfile.seek(0)
        reader = csv.reader(csvfile, dialect)
        reader = csv.DictReader(csvfile)
        modalitats=[]
        modalitatsTimes=[]
        for row in reader:
            if row['MODALITATS'].find(",")=="-1": #Solo hay una modalidad
                if row['MODALITATS'] in modalitats:
                    modalitatsTimes[modalitats.index(row['MODALITATS'])]=modalitatsTimes[modalitats.index(row['M
                else:
                    modalitats.append(row['MODALITATS'])
                    modalitatsTimes.append(1)
```

In this exercise we open the file again and we look through all the entities looking for all the possible modalitats. Here we find a problem because sometimes we can see modalitats that look like this: "football, baseball, gold, petanca" so we have to separate every modalitat when we find this so we can count them correctly.

First we will tackle what happens when the result is only a modalitat.

```
if row['MODALITATS'].find(",")==-1: #Solo hay una modalidad
    if row['MODALITATS'] in modalitats:
        modalitatsTimes[modalitats.index(row['MODALITATS'])]=modalitatsTimes[modalitats
        .index(row['MODALITATS'])]+1
    else:
        modalitats.append(row['MODALITATS'])
        modalitatsTimes.append(1)
```

Simply if this modalitat is already in the list we can add a +1 to the list of times it is in the list. If it's not yet we add them to the list of modalitats and we start it's counter with the number 1. If the entity contains a list of modalitats, we just separate them into a list and then apply the same as we did before for each one of them.

```
else:
    mods=row['MODALITATS'].split(",")
    for mod in mods:
        if mod=="": break
        if mod[0]==" ":
            mod=mod[1:len(mod)]
        if mod in modalitats:
            modalitatsTimes[modalitats.index(mod)]=modalitatsTimes[modalitats.index(mod)]+1
        else:
            modalitats.append(mod)
            modalitatsTimes.append(1)
```

For the sorting we have to execute this line and this will sort the list of modalitats, and then we sort by number the counter.

```
orderModalitats=[modalitats for _,modalitats in sorted(zip(modalitatsTimes,modalitats))]
modalitatsTimes.sort(key=int)
```

And we can print the result.

```
for num in range(0,len(modalitats)):
    print(str(orderModalitats[num])+" un total de "+str(modalitatsTimes[num]))
```

3.- Print a list of the PROVINCIAS on the file, then the user will select one of them. The script will print the COMARQUES there. The user will select one of them. And then print the list of kind (MODALITAT) of entitats and the number for each type.

In this part we are going to use the list of every comarca per provincia that we had made in the first step:

```
barcelonaCom=["Alt Penedès", "Anoia", "Bages", "Baix Llobregat", "Barcelonès", "Berguedà", "Maresme", "Garr
lleidaCom=["Alta Ribagorça", "Alt Urgell", "Cerdanya", "Garrigues", "Noguera", "Pallars Jussà", "Pallars Se
tarragonaCom=["Alt Camp", "Baix Camp", "Baix Ebre", "Baix Penedès", "Conca de Barberà", "Conca de Barber", "
gironaCom=["Alt Empordà", "Baix Empordà", "Garrotxa", "Gironès", "Pla de l'Estany", "Selva", "Ripollès", "Gi
```

First we ask for which provincia the user wants and we print the list of the comarcas of the one chosen, we make them choose which comarca they want from the list.

```
def exercise3():  
    print("Which provincia you want to look for? ")  
    provincia = str(input())  
    if provincia=="Barcelona":  
        print("Escull quina comarca vols")  
        for num in range(0,len(barcelonaCom)):  
            print(str(num)+". "+str(barcelonaCom[num]))  
        comarca = str(input())  
        comarca = barcelonaCom[int(comarca)]  
    elif provincia=="Tarragona":
```

Then we open the file and in every row where the comarca is the same as the one said before we do what we did in step 2 we add the modalitats to a list

```
with open('file.csv', newline='') as csvfile:  
    dialect = csv.Sniffer().sniff(csvfile.read(1024),delimiters=',')  
    csvfile.seek(0)  
    reader = csv.reader(csvfile, dialect)  
    reader = csv.DictReader(csvfile)  
    modalitats=[]  
    modalitatsTimes=[]  
    for row in reader:  
        if row['COMARCA']==comarca:  
            if row['MODALITATS'].find(",")=="-1": #Solo hay una modalidad  
                if row['MODALITATS'] in modalitats:  
                    modalitatsTimes[modalitats.index(row['MODALITATS'])]=m  
                else:  
                    modalitats.append(row['MODALITATS'])  
                    modalitatsTimes.append(1)  
            else:
```

```
                else:  
                    mods=row['MODALITATS'].split(",")  
                    for mod in mods:  
                        if mod=="": break  
                        if mod[0]==" ":  
                            mod=mod[1:len(mod)]  
                        if mod in modalitats:  
                            modalitatsTimes[modalitats.index(mod)]=modalitatsTimes[modalitats.index(mod)]+1  
                        else:  
                            modalitats.append(mod)  
                            modalitatsTimes.append(1)  
    orderModalitats=[modalitats for _,modalitats in sorted(zip(modalitatsTimes,modalitats))]  
    modalitatsTimes.sort(key=int)  
    for num in range(0,len(modalitats)):  
        print(str(orderModalitats[num])+" un total de "+str(modalitatsTimes[num]))
```

After that we print the list with all the modalitats.

4.- Ask for a COMARCA, and then print all the municipalities (MUNICIPIS) for it. The user will select one municipality... then the list of entities there will be printed, the user will select one, and that one will be shown/plotted on a MAP (right resolution to be able to identify the location). With the right zoom in.

First of all we ask the user wich comarca he wants to look for, when he told us we go to another cvs that we have found that contains the list of municipis of every comarca, we have found it here: <https://do.diba.cat/data/ds/municipis/graella?filter-field=--Tot->.

```
def exercise4():
    print("Which comarca you want to look for? ")
    comarca = str(input())
    with open('listaMunicipio.csv', newline='') as csvfile:
        dialect = csv.Sniffer().sniff(csvfile.read(1024), delimiters=',')
        csvfile.seek(0)
        reader = csv.reader(csvfile, dialect)
        reader = csv.DictReader(csvfile)
        municipis=[]
        for row in reader:
            if row['Nom de la comarca']==comarca:
                municipis.append(row['Nom del municipi'])
```

When we have found a list of the municipis in that comarca, we numerate them and ask the user for a number so we can know which one he is looking for.

```
for num in range(0, len(municipis)):
    print(str(num)+". "+str(municipis[num]))
print("Which municipi you want to look for? (Enter the number)")
municipi = str(input())
municipi = municipis[int(municipi)]
print("Choose from one of the following entities on "+municipi+":")
with open('file.csv', newline='') as csvfile:
    dialect = csv.Sniffer().sniff(csvfile.read(1024), delimiters=',')
    csvfile.seek(0)
    reader = csv.reader(csvfile, dialect)
    reader = csv.DictReader(csvfile)
    entitiesInMunicipi=[]
    for row in reader:
        if row['MUNICIPI']==municipi:
            entitiesInMunicipi.append(row['NOM_ENTITAT'])
```

Then we kinda repeat the last step by looking in our csv all the entities in the said municipi, that list of entities will be used to ask the user using the same method as before which one he wants.


```

for num in range(0,len(entitiesInMunicipi)):
    print(str(num)+". "+str(entitiesInMunicipi[num]))
print("Which entity you want to look for? (Enter the number)")
entity = str(input())
entity = entitiesInMunicipi[int(entity)]
print("Printing on map the following entity: "+entity)

```

When we have the entity selected we go to see its address by looking at the csv file one last time and looking for this entity we get its address, postal code and municipi, this will allow us to find **most of the times** in a map where this place can be. Some places have addresses that do not work with our map search system, so if one doesn't work try another, during the execution example we will provide some examples of places that work.

```

print("Printing on map the following entity: "+entity)
with open('file.csv', newline='') as csvfile:
    dialect = csv.Sniffer().sniff(csvfile.read(1024), delimiters=',')
    csvfile.seek(0)
    reader = csv.reader(csvfile, dialect)
    reader = csv.DictReader(csvfile)
    direccion=""
    for row in reader:
        if row['NOM_ENTITAT']==entity:
            direccion=row['ADREÇA']+", "+row['CP']+", "+row['MUNICIPI']
print("The direction of: "+entity+" is "+direccion)

```

With the direction taken we use the next code to get the latitude and longitude of the direction, for this purpose we use an api called Nominatim, we don't even need to register in that but it asks for our mail, so we add it to the document, this is only to make use that we don't ask too much, while testing we hadn't had any problem with that but if it shows an error with this maybe let it rest a little before trying again.

```

loc = Nominatim(user_agent="sergio08301@gmail.com")

```

When we have the longitude and latitude we get a map and place a marker of where this place is, it saves a map, which must be a .html file so we open the file on the browser to look at it.

```

getLoc = loc.geocode(direccion)

m = folium.Map(location=[getLoc.latitude,getLoc.longitude], zoom_start=15)
folium.Marker(location=[getLoc.latitude,getLoc.longitude], popup="<i>"+entity+"</i>").add_to(m)
m.save('map.html')
webbrowser.open_new_tab('map.html')

```

5.- Ask for a POSTAL CODE ... then all the entities with that postal code will be shown/plotted on a MAP (right resolution to be able to identify the location). With the right zoom in.

We start asking to used what postal code he wants, and we look for the entities in the file that has this same code

```
def exercise5():
    print("Tell me a postal code")
    postalcode = str(input())
    with open('file.csv', newline='') as csvfile:
        dialect = csv.Sniffer().sniff(csvfile.read(1024), delimiters=',')
        csvfile.seek(0)
        reader = csv.reader(csvfile, dialect)
        reader = csv.DictReader(csvfile)
        postallist=[]
        for row in reader:
            if row['CP']==postalcode:
                postallist.append(row['NOM_ENTITAT'])
```

We store these entities in a list and look for their directions the same way we used in the last step, asking for the address, postal code and municipi.

```
with open('file.csv', newline='') as csvfile:
    dialect = csv.Sniffer().sniff(csvfile.read(1024), delimiters=',')
    csvfile.seek(0)
    reader = csv.reader(csvfile, dialect)
    reader = csv.DictReader(csvfile)
    direcciones=[]
    for row in reader:
        for num in range(0, len(postallist)):
            if row['NOM_ENTITAT']==postallist[num]:
                direcciones.append(row['ADREÇA']+", "+row['CP']+", "+row['MUNICIPI'])
```

We now have to represent them in a map, first we get its latitude and longitude and then we point at them in the map, note that this will not work for every entity of the file, because some use addresses like: "Centre" and that's hard to point down in a map, so it will display the ones that can be displayed.

```
getLoc = loc.geocode(direcciones[0])
m = folium.Map(location=[getLoc.latitude, getLoc.longitude], zoom_start=15)
for num in range(0, len(postallist)):
    getLoc = loc.geocode(direcciones[num])
    try:
        folium.Marker(location=[getLoc.latitude, getLoc.longitude], popup="<i>"+postallist[num]+"</i>").add_to(m)
    except:
        print(postallist[num]+ " could not be found")
m.save('map.html')
webbrowser.open_new_tab('map.html')
```

6.- Plot/draw/represent on a map (MAPA DE COMARQUES DE CATALUNYA) per each Comarca the number of entities on a color map per Comarca. Print information only for the TOP10 values (the other one not represented).

For the exercises of color map we will use a list of all of the comarcas of catalunya.

```
Comarca=["Alt Penedès", "Anoia", "Bages", "Baix Llobregat", "Barcelonès", "Berguedà", "Maresme", "Garraf", "Osona", "Alta Ribagorça", "Alt Urgell", "Cerdanya", "Garrigues", "Noguera", "Pallars Jussà", "Pallars Sobirà", "Pla d'Urgell", "Alt Camp", "Baix Camp", "Baix Ebre", "Baix Penedès", "Conca de Barberà", "Ribera d'Ebre", "Montsià", "Priorat", "Alt Empordà", "Baix Empordà", "Garrotxa", "Gironès", "Pla de l'Estany", "Ripollès"]
```

We will use too a geojson of the place we want to make the map of, we need the mapa de comarcas de catalunya, we already got it in the folder and will be needed there for this step and the next ones, we have downloaded it from this link:

https://vangdata.carto.com/tables/shapefiles_catalunya_comarcas/public

But before making the map we have to set up all the information, for this reason we make two lists, one with all the comarcas (the list mentioned before) and another one that will store every entities, we will iterate the list adding to the list of entities +1 every time we find the comarca, and then we order the lists.

```
def exercise6():
    content="Comarca,Entidades\n"
    Nentitats=[]
    for num in range(0,len(Comarca)):
        Nentitats.append(1)
    for num in range(0,len(Comarca)):
        with open('file.csv', newline='') as csvfile:
            dialect = csv.Sniffer().sniff(csvfile.read(1024),delimiters=',')
            csvfile.seek(0)
            reader = csv.reader(csvfile, dialect)
            reader = csv.DictReader(csvfile)
            for row in reader:
                if row['COMARCA']==Comarca[num]:
                    Nentitats[num]=Nentitats[num]+1

    orderComarca=[Comarca for _,Comarca in sorted(zip(Nentitats,Comarca))]
    Nentitats.sort(key=int)
```

Here is where we do the sorting and, thats the part that changes between the exercices 6 to 9, as we want the top 10 we will set the result for the other values as 0, so they don't interfere with the top results, they will be the only ones displayed, the other will have 0 entities in the map.

```
for num in range(0,len(orderComarca)):
    if num<len(orderComarca)-10:
        content=content+orderComarca[num]+", "+str(0)+"\n"
    else:
        content=content+orderComarca[num]+", "+str(Nentitats[num])+"\n"
```

As you may have been seeing between the screenshots, we are trying to make our own csv file with the data we need, because in our original file there's not a column with the total of entities for every comarca, so we make our own csv file, we will need only 2 columns thos, one with the name of the comarca and other with the number of entities it contains

```
content="Comarca,Entidades\n"
```

```
for num in range(0,len(orderComarca)):
    if num<len(orderComarca)-10:
        content=content+orderComarca[num]+", "+str(0)+"\n"
    else:
        content=content+orderComarca[num]+", "+str(Nentitats[num])+"\n"
```

```
text_file = open("ranking.csv", "wt")
n = text_file.write(content)
text_file.close()
```

We will save it as ranking.csv and will use this data to make the map.

Now we will start making our choropleth map, we will use this lines to bind the comarcas in our csv file with the comarcas of the geojson map. This will say that every id of the comarca in the map is the same as the id of the comarca in the csv, that we are already readin too in this part of the code.

```
state_id_map = {}
for feature in catalunya_comarcas["features"]:
    feature["id"] = feature["properties"]["comarca"]
    state_id_map[feature["properties"]["nom_comar"]] = feature["id"]

df = pd.read_csv("ranking.csv")

df["id"] = df["Comarca"].apply(lambda x: state_id_map[x])
```

For the creation of the map we will use this method that ask us for the csv file on the first place, then for the location we want, then we say here that we want to use the "id" we have set before, for geojson we use the one with the comarcas of catalunya, color means which column of the csv will be displayed with the colors in a map, hover_name is for when we put the cursor in every comarca it shows it's name, and then a title.

Then we show up the map to the user.

```
fig = px.choropleth(
    df,
    locations="id",
    geojson=catalunya_comarcas, #nombre del geojson usado
    color="Entidades", #la fila que quieres representar
    hover_name="Comarca",
    title="Entitats esportives per comarca en Catalunya",
)

fig.update_geos(fitbounds="locations", visible=False)
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

7.- Plot/draw/represent on a map (MAPA DE COMARQUES DE CATALUNYA) per each Comarca the number of entities on a colour map per Comarca. Print information only for the DOWN10 values (the other one not represented).

This works the same way as step 6, but now when we make the ranking.csv we will make that only the bottom 10 result will have its value, the other have 0.

```
for num in range(0,len(orderComarca)):
    if num>10:
        content=content+orderComarca[num]+", "+"0"+"\\n"
    else:
        content=content+orderComarca[num]+", "+str(Nentitats[num])+"\\n"
```

The rest of the execution is the same as step 6.

8.- Plot/draw/represent on a map (MAPA DE COMARQUES DE CATALUNYA) per each Comarca the number of entities on a colour map per Comarca. Do not print information for BARCELONES (all the comarques except that one).

This works the same way as step 6, but now when we make the ranking.csv we will make that if the Comarca is Barcelonès then the result is 0.

```
for num in range(0,len(orderComarca)):
    if orderComarca[num]=="Barcelonès":
        content=content+orderComarca[num]+", "+"0"+"\\n"
    else:
        content=content+orderComarca[num]+", "+str(Nentitats[num])+"\\n"
```

The rest of the execution is the same as step 6.

9.- Plot/draw/represent on a map (MAPA DE COMARQUES DE CATALUNYA) per each Comarca the number of entities on a colour map per Comarca. Print information for all the comarques.

This works the same way as step 6, but now we display every comarca with its correct value of entities.

```
for num in range(0,len(orderComarca)):
    content=content+orderComarca[num]+", "+str(Nentitats[num])+"\n"
```

The rest of the execution is the same as step 6.

10.- Plot/draw/represent on an histogram the number of entities per Comarca.

We will need the ranking.csv we created in the other steps, so the first step will be to create this file to get the info from it. We just used the same code as we used in step 9 to get a file with the comarcas and the number of entities that every one of them has.

```
def exercise10():
    content="Comarca,Entidades\n"
    Nentitats=[]
    for num in range(0,len(Comarca)):
        Nentitats.append(1)
    for num in range(0,len(Comarca)):
        with open('file.csv', newline='') as csvfile:
            dialect = csv.Sniffer().sniff(csvfile.read(1024), delimiters=',')
            csvfile.seek(0)
            reader = csv.reader(csvfile, dialect)
            reader = csv.DictReader(csvfile)
            for row in reader:
                if row['COMARCA']==Comarca[num]:
                    Nentitats[num]=Nentitats[num]+1

    orderComarca=[Comarca for _,Comarca in sorted(zip(Nentitats,Comarca))]
    Nentitats.sort(key=int)
    for num in range(0,len(orderComarca)):
        content=content+orderComarca[num]+", "+str(Nentitats[num])+"\n"

    text_file = open("ranking.csv", "wt")
    n = text_file.write(content)
    text_file.close()
```

Then we can proceed to the part of the histogram, this is pretty intuitive, we read yet again the ranking.csv, and from there we take the entities and the comarcas, as the ranking is ordered I think that this could be easier to read, and nicer to look as an histogram, for this reason we use the ranking file. We start doing the histogram telling which data we want in X and which one in Y, and then we just make the program show us the result.

```
data = pd.read_csv("ranking.csv")
entidades = data['Entidades']
comarcas = data['Comarca']

fig = px.bar(x=comarcas, y = entidades, labels=dict(x="Comarcas", y="Entidades"))
fig.update_xaxes(type='category')

fig.show()
```

11.- Ask if the user wants to execute a new option or exit

```
print("Lab 2 Python")
selector()
```

We start the exercise by calling a function called selector, which is just a big if/else of which exercise you want to use (because there's no switch case in python).

```
def selector():
    print("Which exercise you want to execute? ")
    exercise = str(input())
    if exercise=="1":
        exercisel1()
        retry()
    elif exercise=="2":
        exercise2()
        retry()
    elif exercise=="3":
```

...

```
    elif exercise=="10":
        exercisel10()
        retry()
    else:
        print("Not a valid number of exercise")
        selector()
```

This selector will send the user to the correct exercise that he has selected, if he select another option this will send him to the choice again. When we end the execution of an exercise it will execute retry() which basically asks you if you want to continue with another exercise or end the execution.


```
def retry():
    print("Want to try another option?")
    answer= str(input())
    if answer=="yes":
        selector()
    else:
        print("Okay, bye")
```

How the environment has been saved

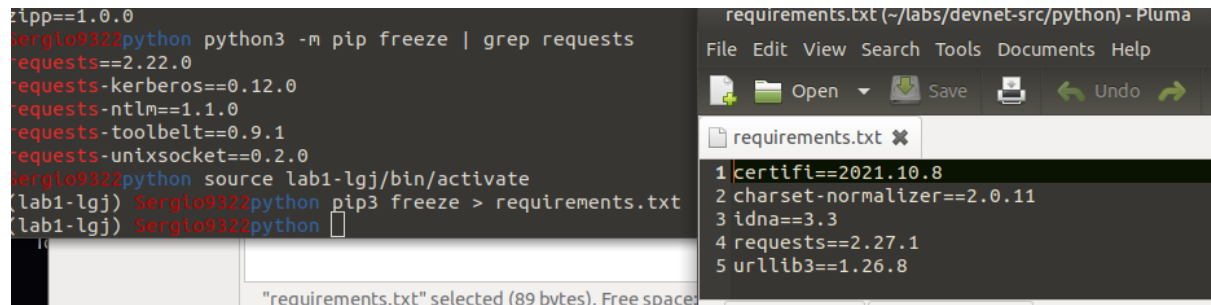
```
Sergio9322~ cd labs/devnet-src/python/  
Sergio9322python python3 -m venv lab1-lgj  
Sergio9322python source lab1-lgj/bin/activate  
(lab1-lgj) Sergio9322python pip3 freeze  
(lab1-lgj) Sergio9322python
```

We go to the folder where we want to save the environment, there we activate it and we type `pip3 freeze` to see that there are no libraries yet.

```
(lab1-lgjj) Sergio9322python pip3 install requests
Collecting requests
  Downloading requests-2.27.1-py2.py3-none-any.whl (63 kB)
    |████████████████████████████████████████| 63 kB 2.0 MB/s
Collecting charset-normalizer~=2.0.0; python_version >= "3"
  Downloading charset_normalizer-2.0.11-py3-none-any.whl (39 kB)
Collecting certifi>=2017.4.17
  Downloading certifi-2021.10.8-py2.py3-none-any.whl (149 kB)
    |████████████████████████████████████████| 149 kB 5.1 MB/s
Collecting idna<4,>=2.5; python_version >= "3"
  Downloading idna-3.3-py3-none-any.whl (61 kB)
    |████████████████████████████████████████| 61 kB 19.5 MB/s
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.8-py2.py3-none-any.whl (138 kB)
    |████████████████████████████████████████| 138 kB 31.9 MB/s
Installing collected packages: charset-normalizer, certifi, idna, urll
sts
Successfully installed certifi-2021.10.8 charset-normalizer-2.0.11 idn
ests-2.27.1 urllib3-1.26.8
(lab1-lgjj) Sergio9322python pip3 freeze
certifi==2021.10.8
charset-normalizer==2.0.11
idna==3.3
requests==2.27.1
urllib3==1.26.8
(lab1-lgjj) Sergio9322python
```


We have now installed all the libraries that are needed to run python3.

How the environment must be imported

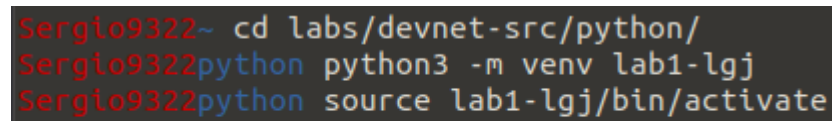


The screenshot shows a terminal window on the left and a text editor window on the right. In the terminal, the user runs `python3 -m pip freeze | grep requests` and `python3 pip3 freeze > requirements.txt`. The text editor shows the contents of `requirements.txt`:

```
certifi==2021.10.8
charset-normalizer==2.0.11
idna==3.3
requests==2.27.1
urllib3==1.26.8
```

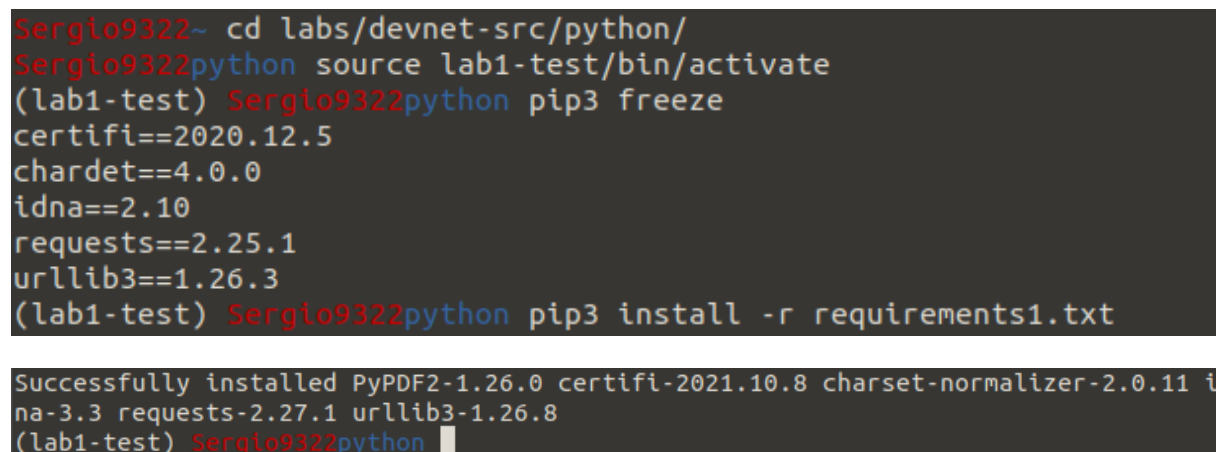
In our virtual environment we create a .txt file with all the libraries, then we can pass this file to whoever wants to have the same virtual environment as us.

For example you can test our work, by creating your virtual environment, as we said in the beginning of the document.



```
Sergio9322~ cd labs/devnet-src/python/
Sergio9322python python3 -m venv lab1-lgj
Sergio9322python source lab1-lgj/bin/activate
```

When we have the new virtual environment, we can load all the libraries using the requirements.txt files, as we can see if we are in the same directory as this files are stored we can load them using `pip3 install -r requirements.txt`.

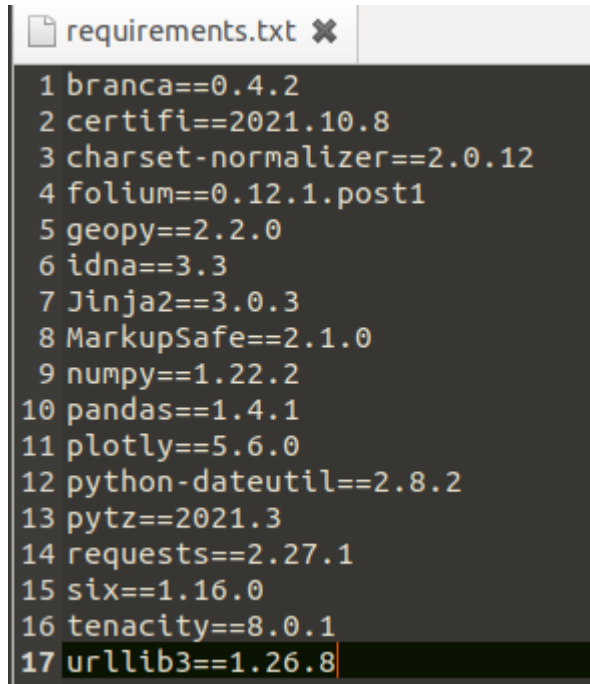


```
Sergio9322~ cd labs/devnet-src/python/
Sergio9322python source lab1-test/bin/activate
(lab1-test) Sergio9322python pip3 freeze
certifi==2020.12.5
chardet==4.0.0
idna==2.10
requests==2.25.1
urllib3==1.26.3
(lab1-test) Sergio9322python pip3 install -r requirements1.txt

Successfully installed PyPDF2-1.26.0 certifi-2021.10.8 charset-normalizer-2.0.11 i
na-3.3 requests-2.27.1 urllib3-1.26.8
(lab1-test) Sergio9322python
```

Now you can use this virtual environment to run our scripts

Packages used on the lab

A screenshot of a code editor window showing a file named 'requirements.txt'. The file contains 17 lines of text, each representing a package and its version. The lines are numbered from 1 to 17. The packages listed are: branca==0.4.2, certifi==2021.10.8, charset-normalizer==2.0.12, folium==0.12.1.post1, geopy==2.2.0, idna==3.3, Jinja2==3.0.3, MarkupSafe==2.1.0, numpy==1.22.2, pandas==1.4.1, plotly==5.6.0, python-dateutil==2.8.2, pytz==2021.3, requests==2.27.1, six==1.16.0, tenacity==8.0.1, and urllib3==1.26.8. The text is in a monospaced font on a dark background.

```
1 branca==0.4.2
2 certifi==2021.10.8
3 charset-normalizer==2.0.12
4 folium==0.12.1.post1
5 geopy==2.2.0
6 idna==3.3
7 Jinja2==3.0.3
8 MarkupSafe==2.1.0
9 numpy==1.22.2
10 pandas==1.4.1
11 plotly==5.6.0
12 python-dateutil==2.8.2
13 pytz==2021.3
14 requests==2.27.1
15 six==1.16.0
16 tenacity==8.0.1
17 urllib3==1.26.8
```

This is the requirements6.txt file, with all the libraries used in all the lab:

- **certifi, charse-normalizer, idna, requests and urllib3** belong to the “requests” libraries needed to execute python that we have added at the creation of the virtual environment.
- **jinja2, markupSafe, branca and folium** work together to make us see in a map a point knowing latitude and longitude.
- **geopy** allows us to get the latitude and longitude of a direction.
- **numpy, pytz, six, dateutil** come with pandas and pandas won’t work without them.
- **plotly and tenacity** allow us to make the choropleth map (mapa de color).
- **plotly and pandas** allow us to make an histogram.

Execution of the program

When we set up the virtual environment we go to the folder where we have our job and execute the code.

```
(lab2-1gj) Sergio9322LAB2_PYTHON_SergioSanchez_GerardTorrent python3 lab2.py
Lab 2 Python
Which exercise you want to execute?
```

Exercise 1:

```
Which exercise you want to execute?
1
En barcelona hi ha 12420, en lleida hi ha 2259, en tarragona hi ha 2259, en girona
hi ha 2571
```

It works just fine, it tells us how many entities there are in every provincia.

Exercise 2:

```
yes
Which exercise you want to execute?
2
```

The list is too long to be displayed completely here but it shows all the different modalitats and how many places practice them.

```
Escacs un total de 676
Tennis taula un total de 722
Tennis un total de 844
Handbol un total de 886
Voleibol un total de 889
Petanca un total de 934
Natació un total de 991
Excursionisme un total de 1061
Caça un total de 1214
Atletisme un total de 1422
Ciclisme un total de 1529
Futbol sala (Futbol cinc) un total de 2322
Basquetbol un total de 2400
Futbol un total de 3663
Want to try another option?
```

Exercise 3:

```
Which exercise you want to execute?  
3  
Which provincia you want to look for?  
Barcelona  
Escull quina comarca vols  
0. Alt Penedès  
1. Anoia  
2. Bages  
3. Baix Llobregat  
4. Barcelonès  
5. Berguedà  
6. Maresme  
7. Garraf  
8. Moianès  
9. Osona  
10. Vallès Occidental  
11. Vallès Oriental  
5
```

The program first ask us witch provincia we want, we can tell him whatever we want,between: Barcelona, Lleida, Tarragona and Girona, We try putting Barcelona, and the it show us all the comarcas , we can pick a number of the comarca, for example if we want Berguedà we pick 5, and then a big list of modalitats will show up.

```
5  
Acampada (excursionisme) un total de 1  
Acrobàcia un total de 1  
Ball esportiu un total de 1  
Biketrial un total de 1  
Billar un total de 1
```

...

```
Futbol sala (Futbol cinc) un total de 20  
Excursionisme un total de 21  
Ciclisme un total de 24  
Caça un total de 34  
Futbol un total de 34  
Want to try another option?
```

Exercise 4:

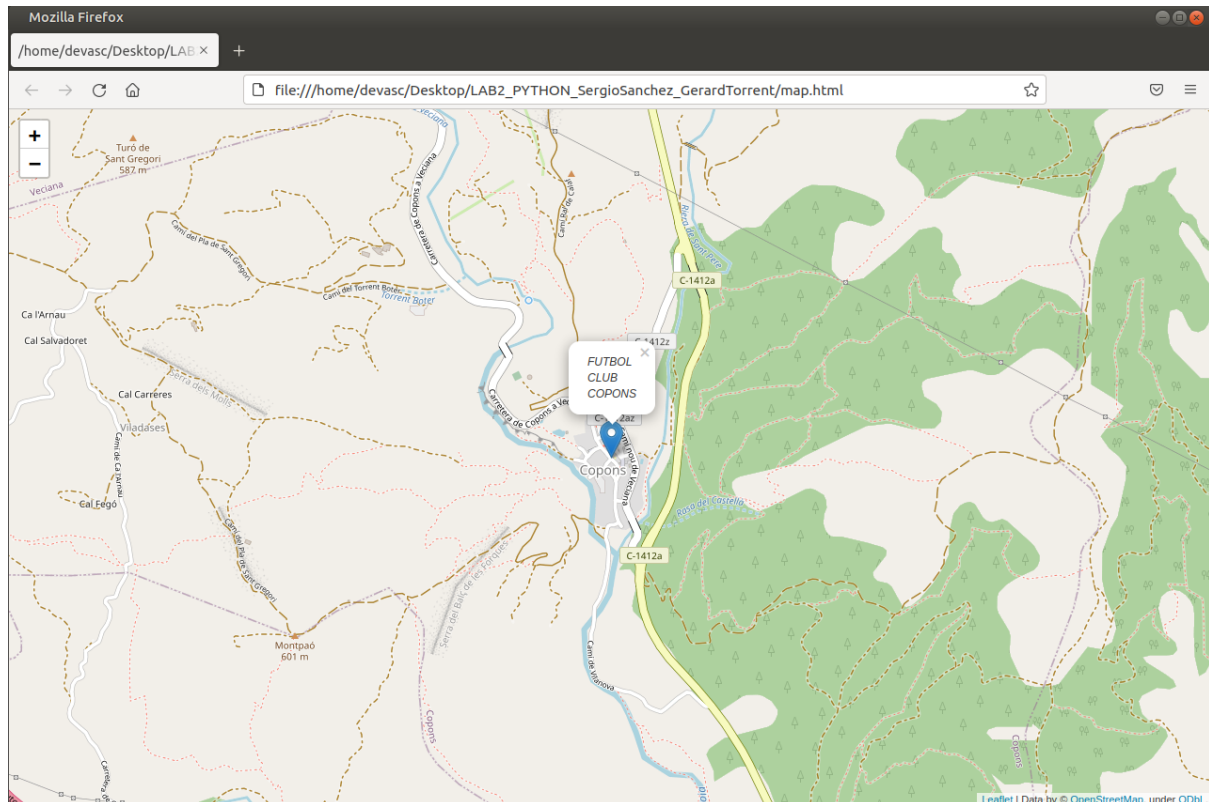
We have chosen Anoia when the program ask us what is the Comarca, then we pick between all the municipis of Anoia

```
Which exercise you want to execute?  
4  
Which comarca you want to look for?  
Anoia  
0. Els Prats de Rei  
1. Montmaneu  
2. Castellfollit de Riubregós  
3. Vilanova del Camí  
4. Capellades  
5. Sant Martí Sesgueioles  
6. Santa Maria de Miralles  
7. Orpí
```

```
24. Copons  
25. Piera  
26. Sant Pere Sallavinera  
27. Calonge de Segarra  
28. La Llacuna  
29. El Bruc  
30. Masquefa  
31. Jorba  
32. Igualada  
Which municipi you want to look for? (Enter the number)  
24  
Choose from one of the following entities on Copons:  
0. ASSOCIACIO DE MARES I PARES D'ALUMNES DEL CEIP COPONS  
1. FUTBOL CLUB COPONS  
2. SOCIETAT DE CAÇADORS COPONS - VECIANA  
3. ALLRIDERS PRO CLUB ESPORTIU
```

We have picked Copons, so we input number 24, then we chose between the 3 entities that are in Anoia.

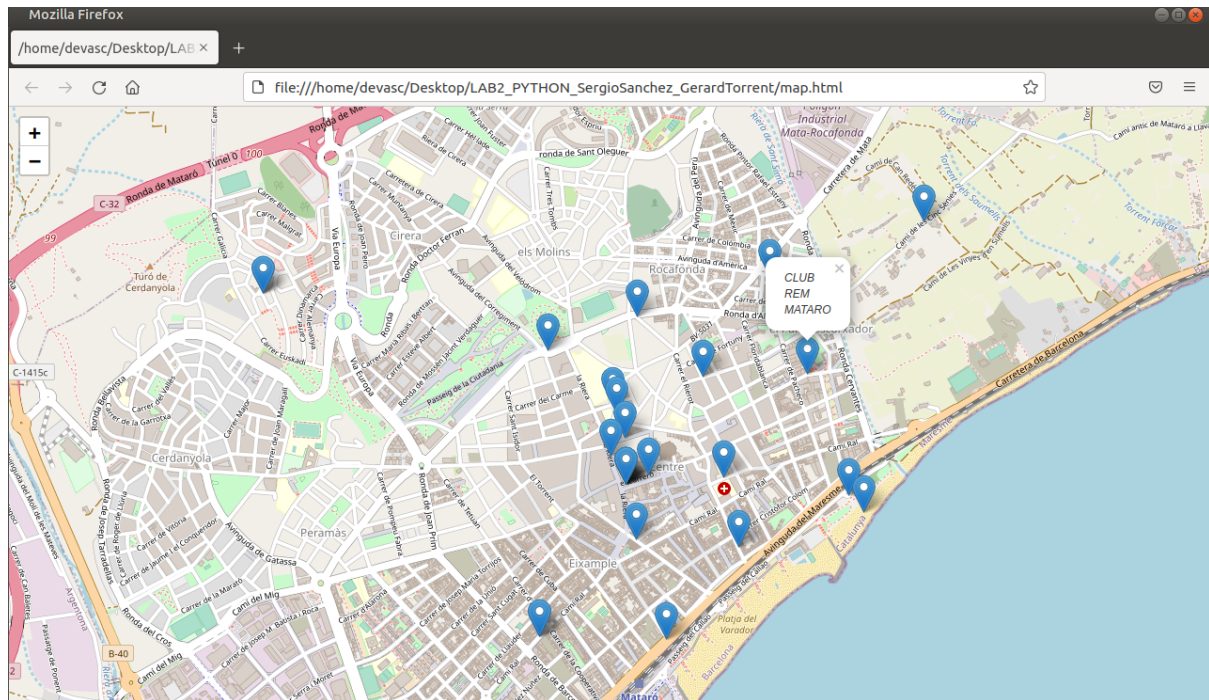
We have inputted 1 so we chose “Futbol club copons” and then the map will provide us with its location



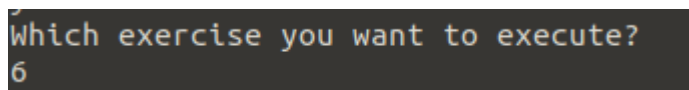
Exercise 5:

```
Tell me a postal code
08301
UNION RECREATIVA LARU could not be found
ASSOCIACIO FISTBALL CATALUNYA could not be found
C.F. PENYA X could not be found
CLUB DEPORTIVO CRISTALERIAS DE MATARO could not be found
CLUB ESPORTIU VALLDEMIÀ-MARISTES could not be found
GRUPFA (GRUP FALCONERIA ESTUDI I PROTECCIO D'AUS DE PRESA) could not be found
CLUB PETANCA ROCAFONDA could not be found
CLUB PETANCA CROS could not be found
CLUB OLIMPIC KORIO could not be found
SOCIETAT DE PESCA I ACTIVITATS SUBAQUÀTIQUES DE MATARO could not be found
GRUP MARATONIA MATARO could not be found
PENYA L'ESPLANADA could not be found
CLUB ESPORTIU DE PESCA PEKIN could not be found
CLUB VELA MATARO could not be found
CLUB DE FUTBOL ATLETIC LAIETA could not be found
CLUB VOLEIBOL MATARO could not be found
CLUB ESPORTIU RCZ MATARO could not be found
MOTO CLUB PARCMOTOR CASTELLOLI could not be found
CLUB ESPORTIU PATINATGE ARTÍSTIC SANT ISCLE DE VALLALTA could not be found
CLUB PESCA ESPORTIVA MAR MEDITERRANI could not be found
CLUB DE FUTBOL SALA CIUTAT DE MATARO could not be found
SKATE-BOARD CLUB MATARO could not be found
CLUB ESCALADA MATARO could not be found
Want to try another option?
```

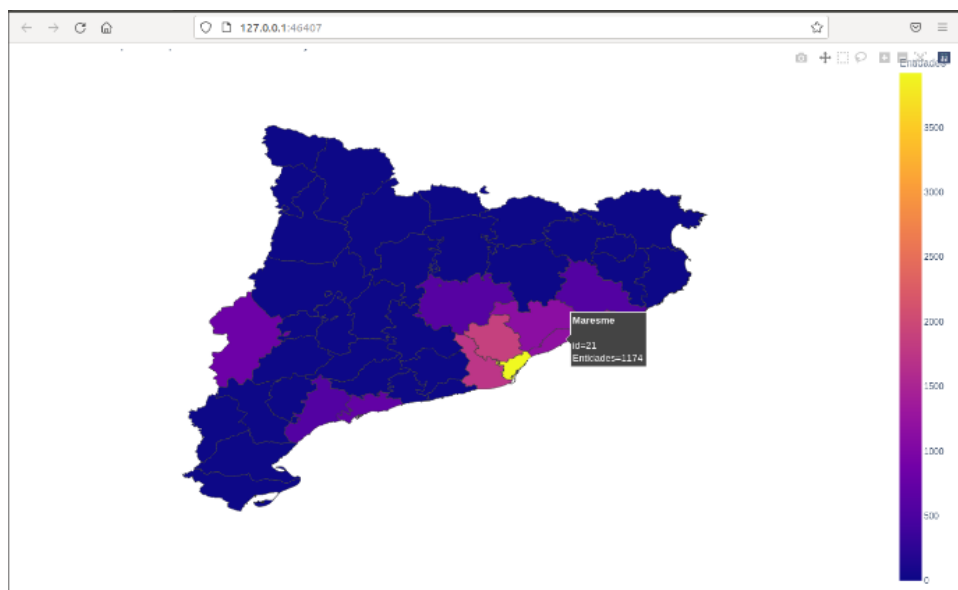

We told the program to look for the postal code of 08301, which belongs to Mataró, sadly it will tell us that some places can not be found, but then it will show that it has found a bunch, because we have told him a zone with a lot of entities.



Exercise 6:



We input number 6 and then the program will show us a map of the comarcas of catalonia, with only the results of the first 10 comarcas in entities, the other will be set at 0.

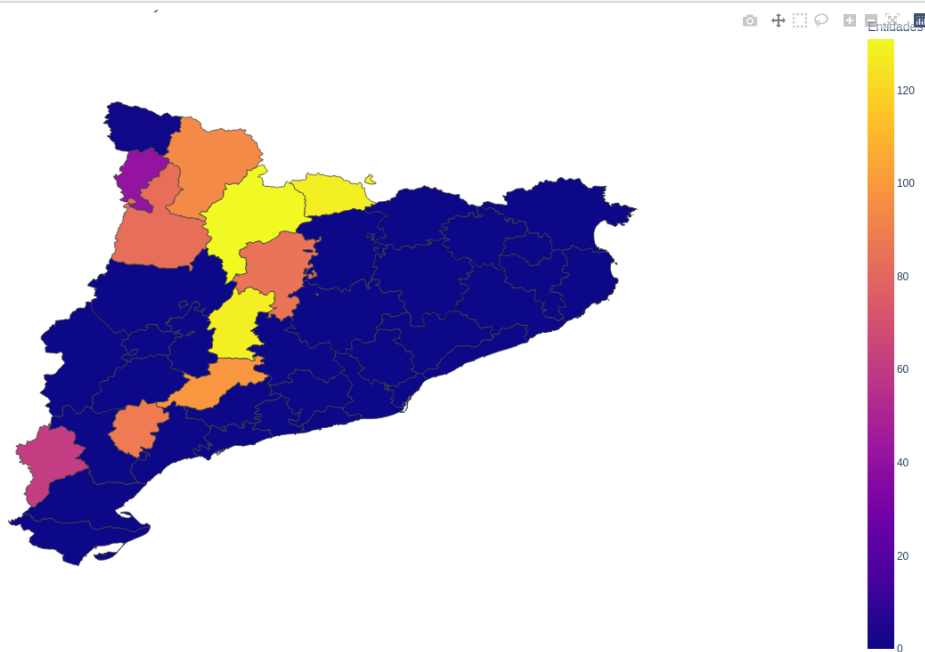


We can use our mouse to hover a comarca and it will tell us how many entities it has.

Exercise 7:

```
Which exercise you want to execute?  
7
```

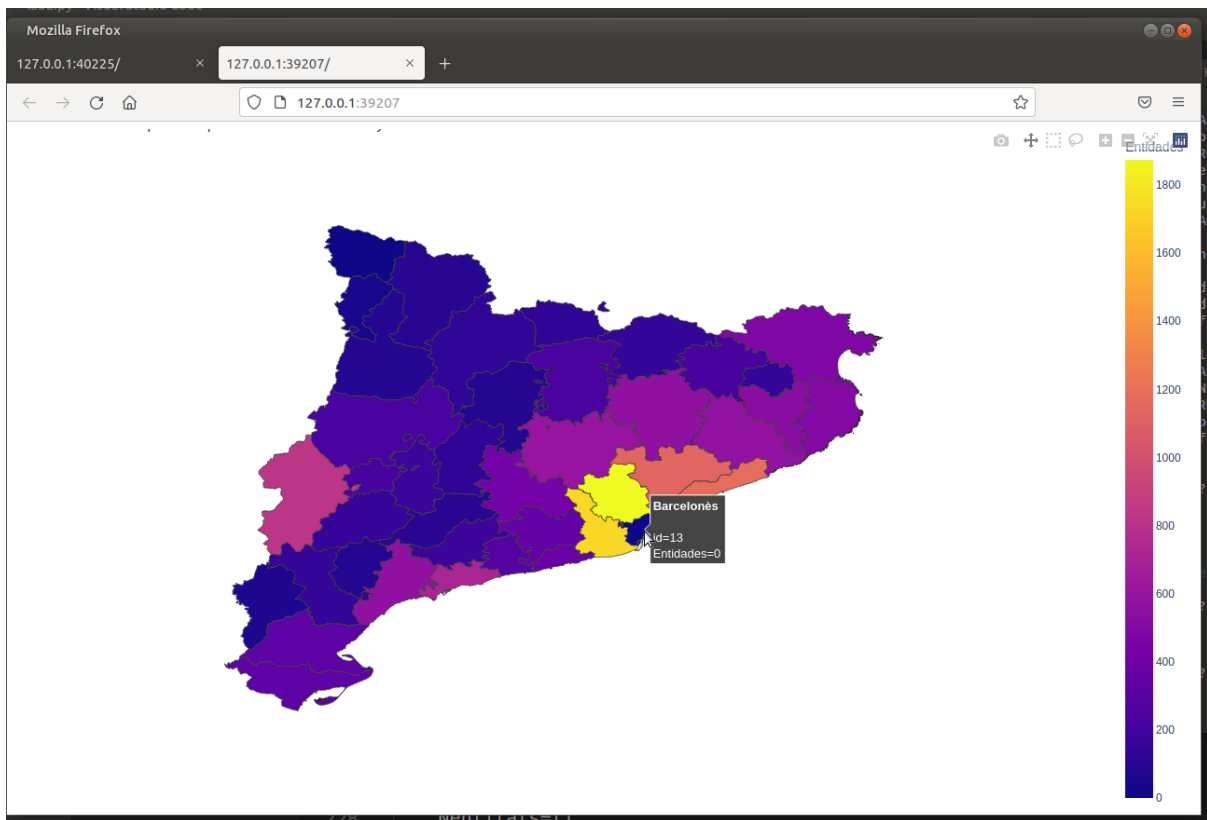
This result is the down 10 comarcas in number of entities



Exercise 8:

```
Which exercise you want to execute?  
8
```

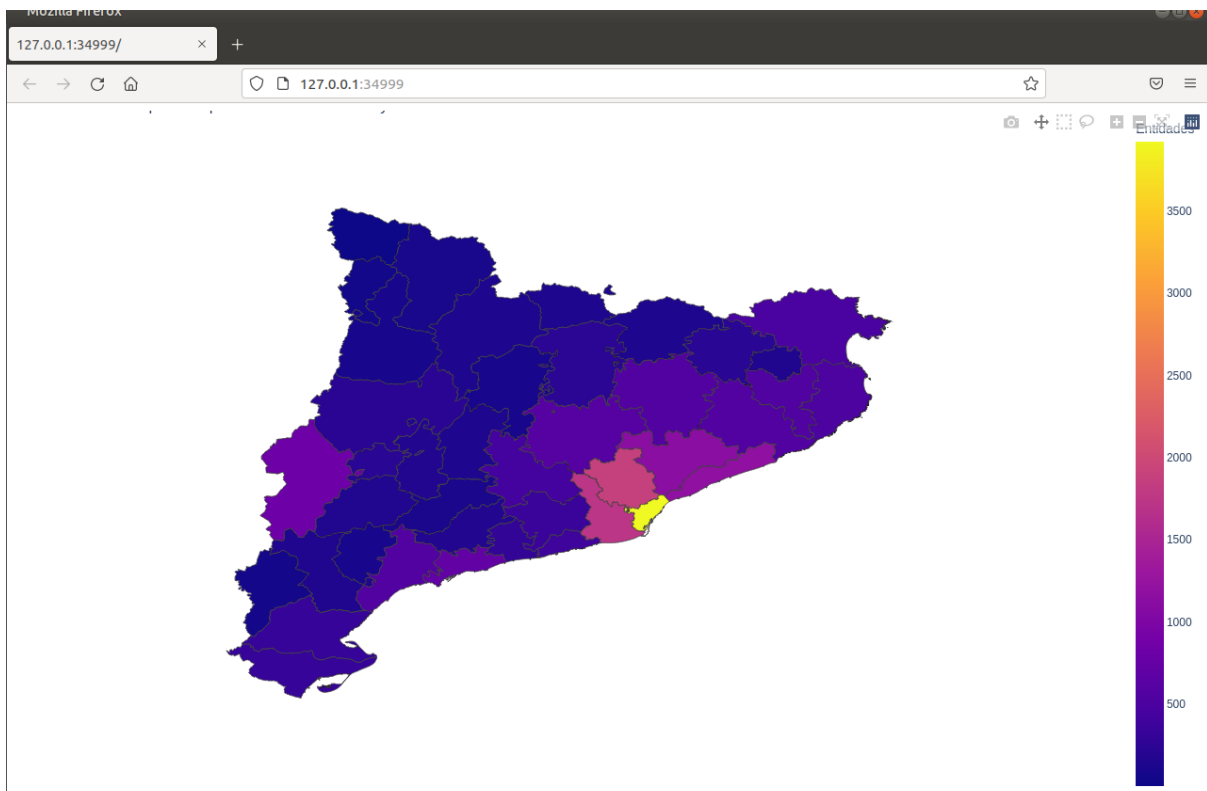
We can see that Barcelonès counts as if it had 0 entities, and the other ones have the amount it should.



Exercise 9:

```
Which exercise you want to execute?  
9
```

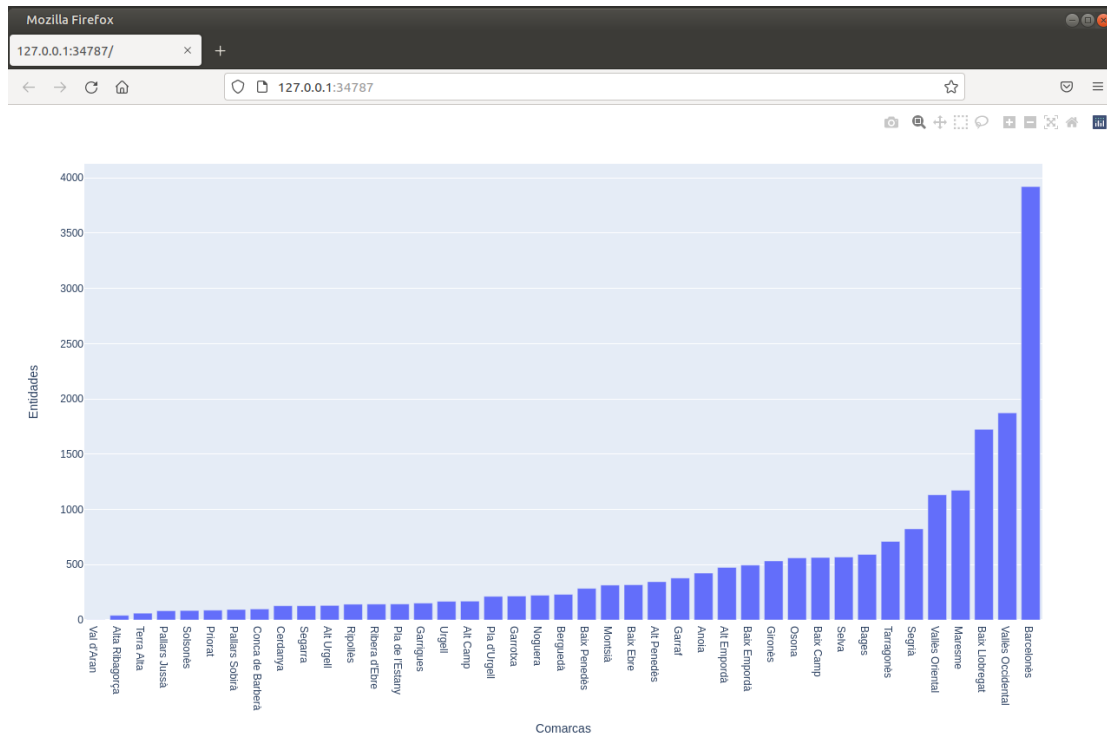
We see that this time every comarca has its correct amount of entities



Exercise 10:

```
Which exercise you want to execute?  
10
```

It will pop up a histogram where we can see clearly how many entities has every comarca, we can hover every bar to see the exact number



Exercise 11:

```
Which exercise you want to execute?  
10  
Want to try another option?  
no  
Okay, bye  
(lab2-lgj) Sergio9322LAB2_PYTHON_SergioSanchez_GerardTorrent
```

We are asked after every execution if we want to try another option, if our answer is not “yes” it will exit the program.

```
Want to try another option?  
yes  
Which exercise you want to execute?
```

If we say “yes” we can choose another exercise.