219114/115 Programming I

OOP programming laboratory 3

To do this lab, you must have first completed the OOP programming lab. 2. Click on the link below to do that lab before proceeding to do this lab.

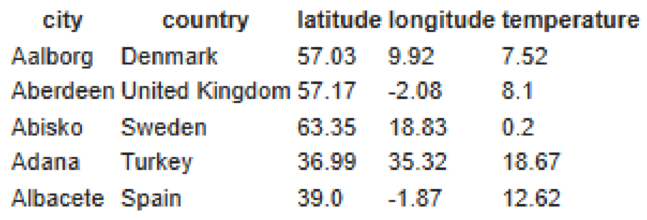
oop_lab_2

Commit 4

In processing data, we are more likely than not to encounter a situation which involves multiple tables. These tables live in some kind of database. In later courses, you will learn more about this, but for this lab, let our database be a list of tables.

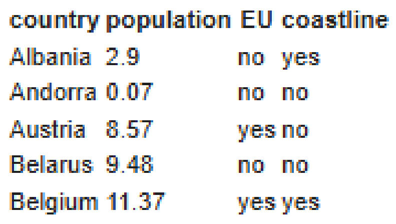
Two tables can be joined together if they have a common attribute (also called key). The following shows how cities and countries tables can be joined on a common key that is ‘country’.

cities



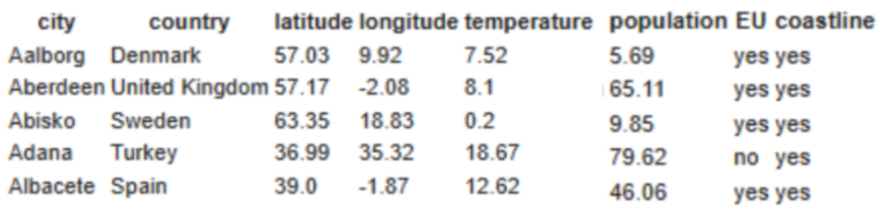
city	country	latitude	longitude	temperature
Aalborg	Denmark	57.03	9.92	7.52
Aberdeen	United Kingdom	57.17	-2.08	8.1
Abisko	Sweden	63.35	18.83	0.2
Adana	Turkey	36.99	35.32	18.67
Albacete	Spain	39.0	-1.87	12.62

countries



country	population	EU	coastline
Albania	2.9	no	yes
Andorra	0.07	no	no
Austria	8.57	yes	no
Belarus	9.48	no	no
Belgium	11.37	yes	yes

cities_countries_joined



city	country	latitude	longitude	temperature	population	EU	coastline
Aalborg	Denmark	57.03	9.92	7.52	5.69	yes	yes
Aberdeen	United Kingdom	57.17	-2.08	8.1	65.11	yes	yes
Abisko	Sweden	63.35	18.83	0.2	9.85	yes	yes
Adana	Turkey	36.99	35.32	18.67	79.62	no	yes
Albacete	Spain	39.0	-1.87	12.62	46.06	yes	yes

For our Python code, a row in the table is actually a dictionary, e.g,
```
{'city': 'Augsburg', 'country': 'Germany', 'latitude': '48.35',
'longitude': '10.90', 'temperature': '4.54'}
```

And the table itself is a list of dictionaries, e.g., a filtered table that shows all the cities in Spain with temperature above 16°C:
```
[{'city': 'Algeciras', 'country': 'Spain', 'latitude': '36.13',
'longitude': '-5.47', 'temperature': '17.38'}, {'city': 'Cartagena',
'country': 'Spain', 'latitude': '37.60', 'longitude': '-0.98',
'temperature': '17.32'}, {'city': 'Granada', 'country': 'Spain',
'latitude': '37.16', 'longitude': '-3.59', 'temperature': '16.33'},
{'city': 'Huelva', 'country': 'Spain', 'latitude': '37.25',
'longitude': '-6.93', 'temperature': '17.09'}, {'city': 'Marbella',
'country': 'Spain', 'latitude': '36.52', 'longitude': '-4.88',
'temperature': '17.19'}, {'city': 'Valencia', 'country': 'Spain',
'latitude': '39.49', 'longitude': '-0.40', 'temperature': '16.02'}]
```

We want to modify the code in Commit 3 so that it has a database (DB) class where tables can be inserted and searched for. In addition, we also want to add the operation to join two tables in the Table class.

Get the starting code from the following link:

https://gist.github.com/parujr/fce6082cce20fdf11f4e21e753e1cd0a

Get the additional dataset Countries.csv from the following link:

Countries.csv

Your job is to complete the missing code and make sure that all the given test code runs successfully. A successful run produces the result shown in the following link:

https://gist.github.com/parujr/9c1211508dc2e8e897ed40b6ec40a2c2

Once you are done with the implementation, create a README.md file that includes:

- Lab Overview - What is this lab all about?
- Project Structure - How files are organized
- Design Overview - Detailed explanation of each class, detailing attributes and key methods
- How to test and run your code

Make the final commit with a meaningful message and push your changes to the remote Github repository.

What to submit

- Show your final commit at your remote Github repository to the TAs so they can check you off
- Put the link to the Github repository to Google Classroom for grading