

Lab 5

By the due date, upload this lab to the Learning Hub in the Activities -> Assignments -> Lab 5 Dropbox. **Late submissions are not accepted.**

Create a new Lab 5 project in Pycharm and create a script called **countries.py**. You will do all your coding in this script.

Copy the following code (download the file datafile.txt in the module 5) into your IDE to create three tuples. Do not change any of this code.

1. **countries_and_capitals:** this is a tuple of lists. Each element of this tuple is a list of two strings: a country and its capital city.
2. **countries:** this is a tuple of strings. Each element of this tuple is a string: the name of a country.
3. **capitals:** this is a tuple of strings. Each element of this tuple is a string: the name of a capital city.

Lab Instructions

Now that we have three tuples of data with which to work, your job is to write eight functions to meet the following requirements. Create functions with the following names, parameters, and return types. Note that all of these functions will work regardless of letter case (so 'A' is the same as 'a', for example. Do not alter the data in the tuples (given above).

Only use Python syntax we have covered so far. For any looping, use while loops. No marks will be given for using other types of looping or iteration.

Function Name	Parameters	Return
how_many_countries (1 mark)		Integer
get_name_of_longest_country (1 marks)		String
get_number_of_capitals_containing (1 marks)	substring	Integer
get_countries_and_capitals_that_start_with_same_letter (2 marks)		List of Strings
get_capital_of (2 marks)	country - whose capital to return	String or None
get_list_of_countries_with_this_many_letters_in_name (2 marks)	num_letters	List of Strings
get_capitals_and_countries_that_begin_and_end_with_sam letter (3 marks)		List of Strings (country names and then capital names)
print_countries_in_reverse_alphabetical_order (3 marks)		None

Create a **main** function in your countries.py script and test out your functions.

A sample run looks like this:

```
print(how_many_countries())
print(get_name_of_longest_country())
print(get_number_of_capitals_containing('e'))
print(get_number_of_capitals_containing('z'))
print(get_number_of_capitals_containing(''))
print(get_number_of_capitals_containing('an'))
print(get_countries_and_capitals_that_start_with_same_letter())
print(get_capital_of('canada')) # must use the countries_and_capitals list
print(get_capital_of('nEW zeALAND')) # must use the countries_and_capitals list
print(get_capital_of('xyz')) # must use the countries_and_capitals list
print(get_list_of_countries_with_this_many_letters_in_name(11))
print(get_list_of_countries_with_this_many_letters_in_name(15))
print(get_capitals_and_countries_that_begin_and_end_with_same_letter())
print_countries_in_reverse_alphabetical_order()
```

When you run the code above the correct solution will give the following results:

<code>print(how_many_countries())</code>
200
<code>print(get_name_of_longest_country())</code>
Congo, Democratic Republic of the
<code>print(get_number_of_capitals_containing('e'))</code>
75
<code>print(get_number_of_capitals_containing('z'))</code>
3 # Brazzaville, Zagreb, Vaduz
<code>print(get_number_of_capitals_containing(''))</code> # apostrophe

5
<pre>print(get_number_of_capitals_containing('an'))</pre> <p>35</p>
<pre>print(get_countries_and_capitals_that_start_with_same_letter())</pre> <p># put a hyphen between country and capital, as follows</p> <p>['Algiers - Algeria', 'Andorra la Vella - Andorra', 'Bridgetown - Barbados', 'Brussels - Belgium', 'Belmopan - Belize', 'Brasilia - Brazil', 'Bandar Seri Begawan - Brunei', 'Djibouti - Djibouti', 'Guatemala City - Guatemala', 'Georgetown - Guyana', 'Kuwait City - Kuwait', 'Luxembourg - Luxembourg', 'Male - Maldives', 'Majuro - Marshall Islands', 'Mexico City - Mexico', 'Monaco - Monaco', 'Maputo - Mozambique', 'No official capital - Nauru', 'Niamey - Niger', 'Panama City - Panama', 'Port Moresby - Papua New Guinea', 'San Marino - San Marino', 'Sao Tome - Sao Tome and Principe', 'Singapore - Singapore', 'Seoul - South Korea', 'Stockholm - Sweden', 'Taipei - Taiwan', 'Tunis - Tunisia', 'Vatican City - Vatican City']</p>
<pre>print(get_capital_of('canada')) # case insensitive</pre> <p>Ottawa</p>
<pre>print(get_capital_of('nEW zeALAND')) #case insensitive</pre> <p>Wellington</p>
<pre>print(get_capital_of('xyz'))</pre> <p>None</p>
<pre>print(get_list_of_countries_with_this_many_letters_in_name(11))</pre> <p>['Afghanistan', 'El Salvador', 'Netherlands', 'New Zealand', 'North Korea', 'Philippines', 'Saint Lucia', 'South Korea', 'South Sudan', 'Switzerland']</p>
<pre>print(get_list_of_countries_with_this_many_letters_in_name(15))</pre> <p>['Myanmar (Burma)', 'Solomon Islands']</p>

```
print(get_capitals_and_countries_that_begin_and_end_with_same_letter())
# first list all the countries then all the capitals, as follows
```

```
['Albania', 'Algeria', 'Andorra', 'Angola', 'Antigua and Barbuda', 'Argentina', 'Armenia',
'Australia', 'Austria', 'Central African Republic', 'Saint Kitts and Nevis', 'Saint Vincent and
the Grenadines', 'Seychelles', 'Solomon Islands', 'Andorra la Vella', "Saint John's", 'Asmara',
'Addis Ababa', 'Accra', "Saint George's", 'Nur-Sultan', 'Abuja', 'Oslo', 'Warsaw', 'Apia',
'Ankara', 'Tashkent']
```

```
print_countries_in_reverse_alphabetical_order()
```

```
('Zimbabwe', 'Zambia', 'Yemen', 'Wales', 'Vietnam', 'Venezuela', 'Vatican City', 'Vanuatu',
'Uzbekistan', 'Uruguay', 'United States', 'United Kingdom', 'United Arab Emirates', 'Ukraine',
'Uganda', 'Tuvalu', 'Turkmenistan', 'Turkey', 'Tunisia', 'Trinidad and Tobago', 'Tonga',
'Togo', 'Thailand', 'Tanzania', 'Tajikistan', 'Taiwan', 'Syria', 'Switzerland', 'Sweden',
'Suriname', 'Sudan', 'Sri Lanka', 'Spain', 'South Sudan', 'South Korea', 'South Africa',
'Somalia', 'Solomon Islands', 'Slovenia', 'Slovakia', 'Singapore', 'Sierra Leone', 'Seychelles',
'Serbia', 'Senegal', 'Scotland', 'Saudi Arabia', 'Sao Tome and Principe', 'San Marino', 'Samoa',
'Saint Vincent and the Grenadines', 'Saint Lucia', 'Saint Kitts and Nevis', 'Rwanda', 'Russia',
'Romania', 'Qatar', 'Portugal', 'Poland', 'Philippines', 'Peru', 'Paraguay', 'Papua New Guinea',
'Panama', 'Palau', 'Pakistan', 'Oman', 'Norway', 'Northern Ireland', 'North Macedonia
(Macedonia)', 'North Korea', 'Nigeria', 'Niger', 'Nicaragua', 'New Zealand', 'Netherlands',
'Nepal', 'Nauru', 'Namibia', 'Myanmar (Burma)', 'Mozambique', 'Morocco', 'Montenegro',
'Mongolia', 'Monaco', 'Moldova', 'Mexico', 'Mauritius', 'Mauritania', 'Marshall Islands',
'Malta', 'Mali', 'Maldives', 'Malaysia', 'Malawi', 'Madagascar', 'Luxembourg', 'Lithuania',
'Liechtenstein', 'Libya', 'Liberia', 'Lesotho', 'Lebanon', 'Latvia', 'Laos', 'Kyrgyzstan', 'Kuwait',
'Kosovo', 'Kiribati', 'Kenya', 'Kazakhstan', 'Jordan', 'Japan', 'Jamaica', 'Italy', 'Israel', 'Ireland',
'Iraq', 'Iran', 'Indonesia', 'India', 'Iceland', 'Hungary', 'Honduras', 'Haiti', 'Guyana', 'Guinea-
Bissau', 'Guinea', 'Guatemala', 'Grenada', 'Greece', 'Ghana', 'Germany', 'Georgia', 'Gambia',
'Gabon', 'France', 'Finland', 'Fiji', 'Federated States of Micronesia', 'Ethiopia', 'Eswatini
(Swaziland)', 'Estonia', 'Eritrea', 'Equatorial Guinea', 'England', 'El Salvador', 'Egypt',
'Ecuador', 'East Timor', 'Dominican Republic', 'Dominica', 'Djibouti', 'Denmark', 'Czech
Republic (Czechia)', 'Cyprus', 'Cuba', 'Croatia', "CÃ'te d'Ivoire (Ivory Coast)", 'Costa Rica',
'Congo, Republic of the', 'Congo, Democratic Republic of the', 'Comoros', 'Colombia',
'China', 'Chile', 'Chad', 'Central African Republic', 'Cape Verde', 'Canada', 'Cameroon',
'Cambodia', 'Burundi', 'Burkina Faso', 'Bulgaria', 'Brunei', 'Brazil', 'Botswana', 'Bosnia and
Herzegovina', 'Bolivia', 'Bhutan', 'Benin', 'Belize', 'Belgium', 'Belarus', 'Barbados',
'Bangladesh', 'Bahrain', 'Bahamas', 'Azerbaijan', 'Austria', 'Australia', 'Armenia', 'Argentina',
'Antigua and Barbuda', 'Angola', 'Andorra', 'Algeria', 'Albania', 'Afghanistan')
```

Note that your instructor may test with other code too...not just the sample run listed above. The sample run listed above is intended only to show whether your code appears to be running as expected.