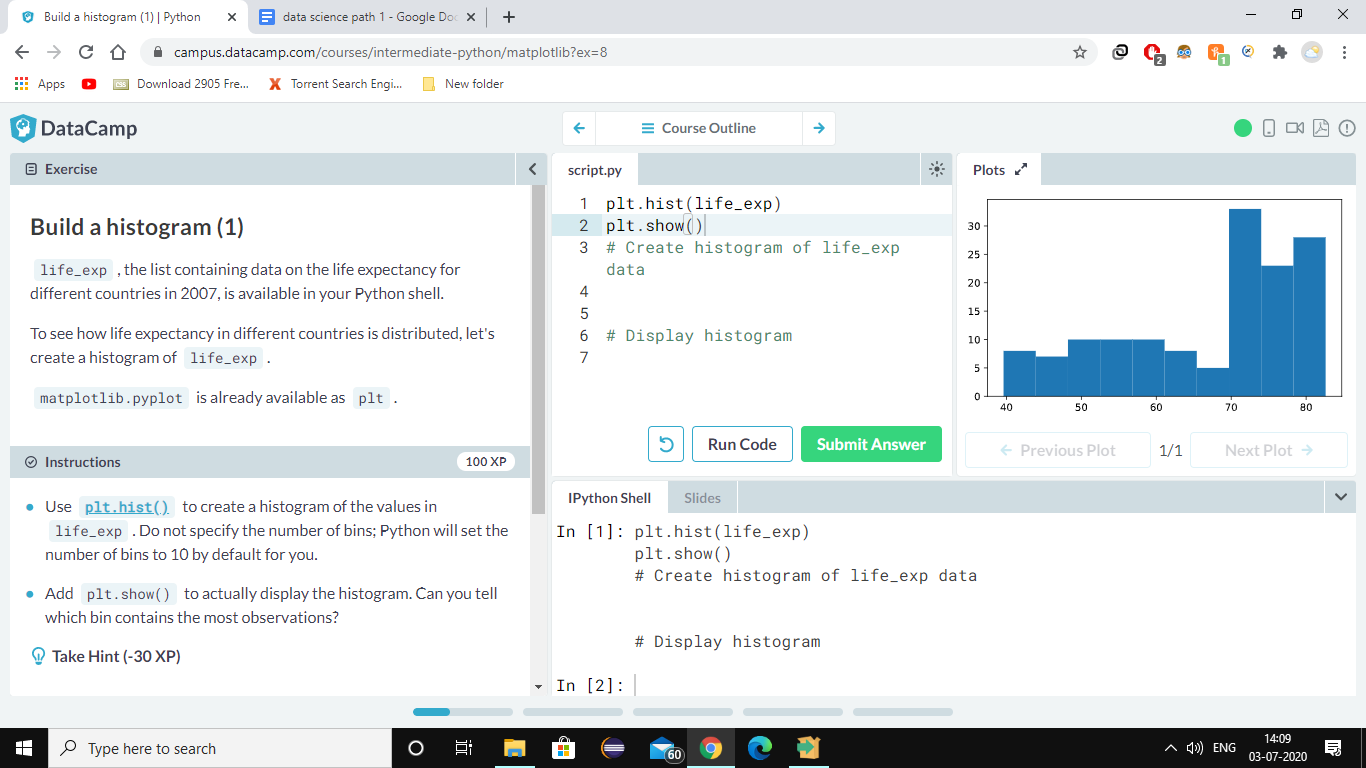
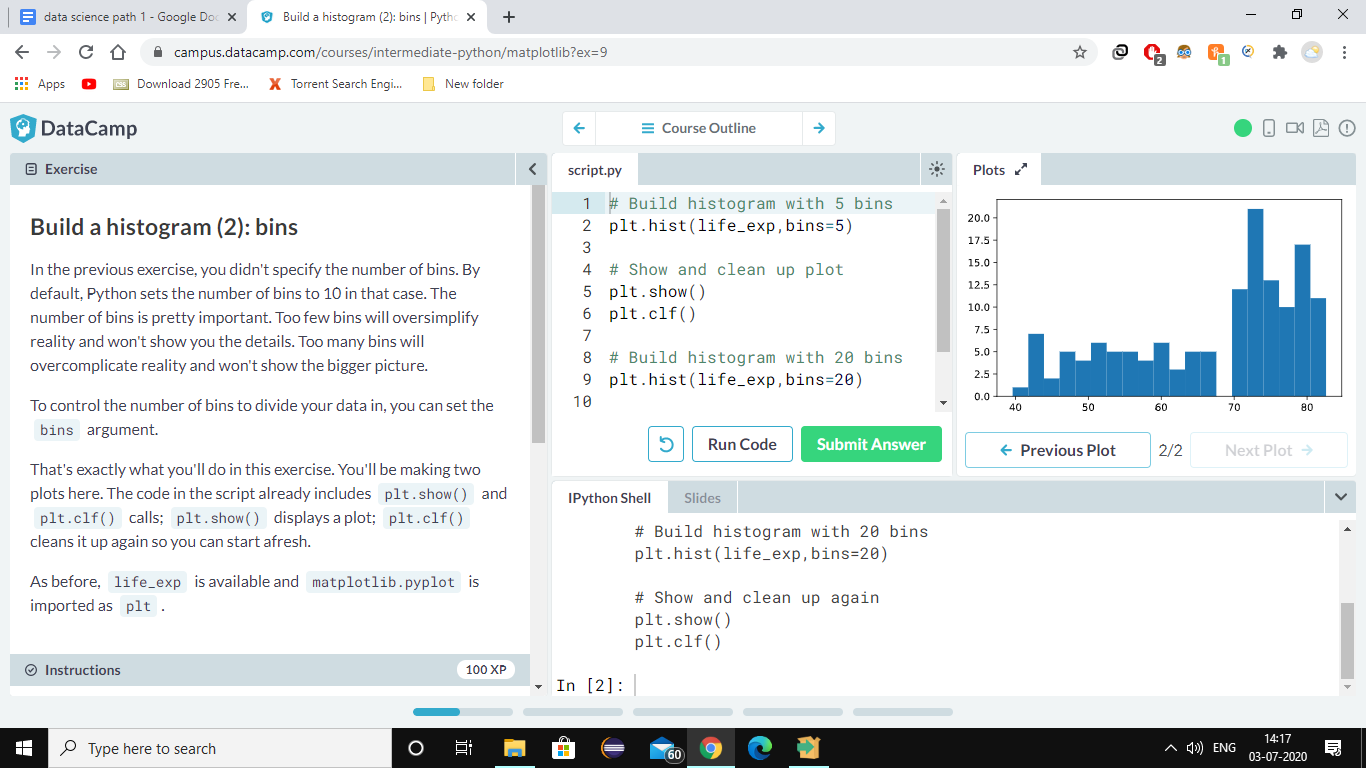
**Seeing histogram**

****

**plt.clf()**

****

**Here** code defines that the graph shown is from bin=20 not bin=5 as plt.clf() clear the the graph from bin=5.if you hide clf then it will give you both in one as overlapped

**QUES: You're a professor teaching Data Science with Python, and you want to visually assess if the grades on your exam follow a particular distribution. Which plot do you use?**

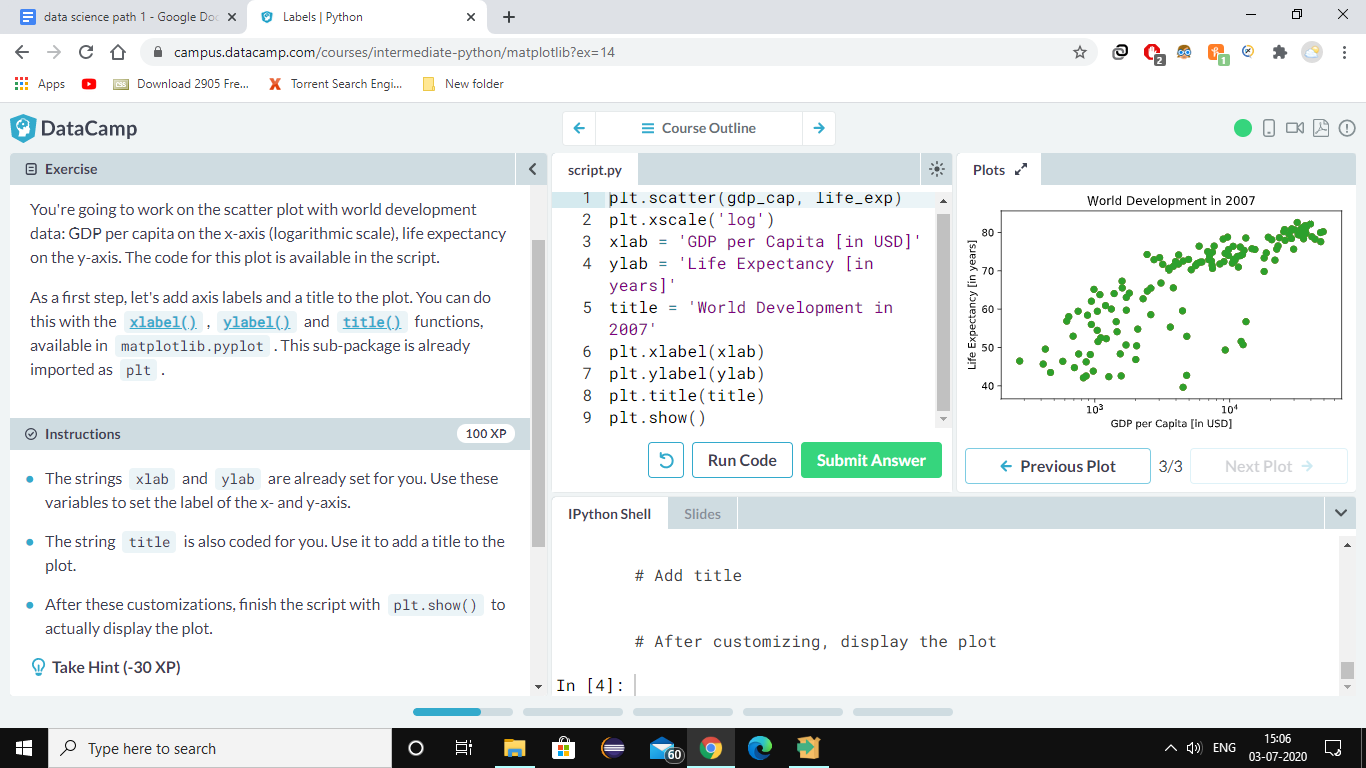
**SOL. histogram plot**

**QUES:You're a professor in Data Analytics with Python, and you want to visually assess if longer answers on exam questions lead to higher grades. Which plot do you use?**

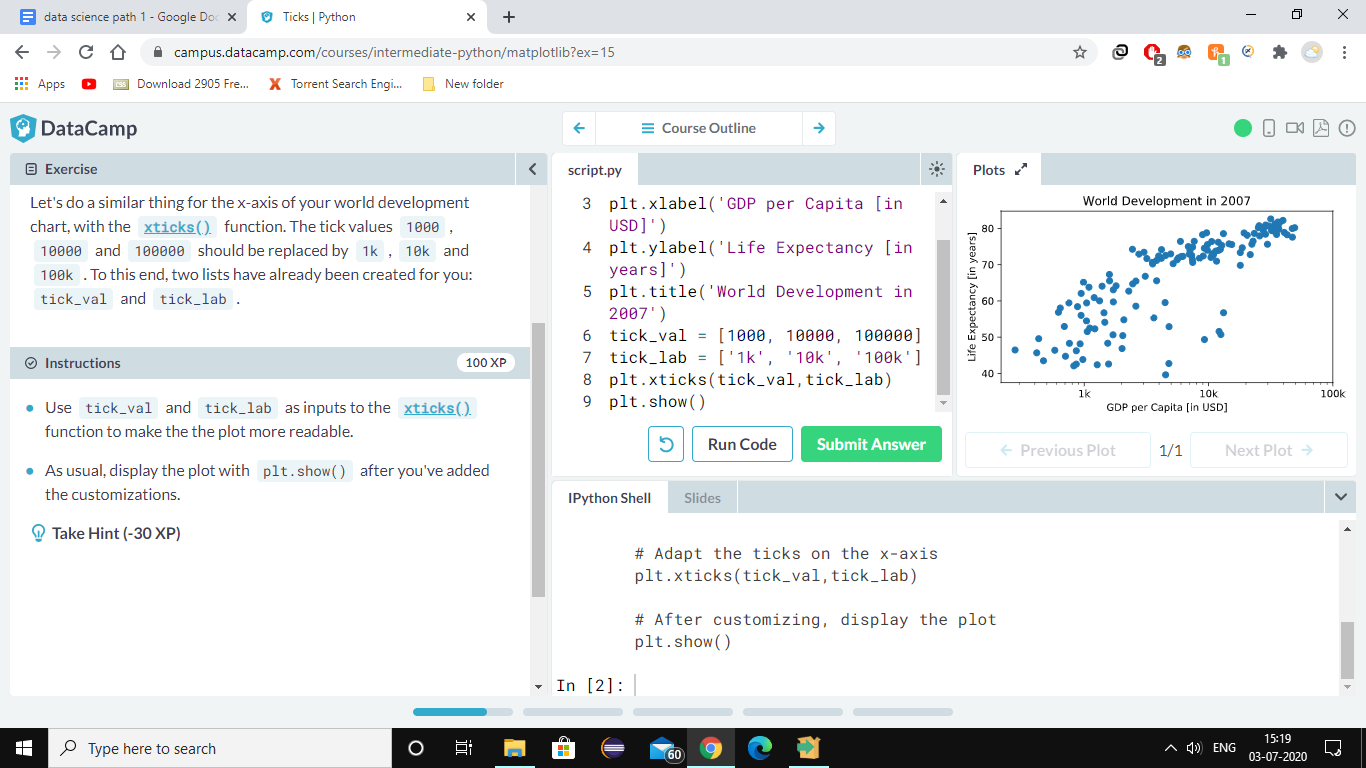
**SOL. scatter plot**

**CUSTOMIZATION**

Xlabel,ylabel,title



USE ticks to shorten your work



PROGRAM

**There is a list pop loaded in your workspace. It contains population numbers for each country expressed in millions. You can see that this list is added to the scatter method, as the argument s, for size.**

**# Import numpy as np**

import numpy as np

**# Store pop as a numpy array: np\_pop**

np\_pop=np.array(pop)

**# Double np\_pop**

np\_pop=np\_pop\*2

**# Update: set s argument to np\_pop**

plt.scatter(gdp\_cap, life\_exp, s = np\_pop)

**# Previous customizations**

plt.xscale('log')

plt.xlabel('GDP per Capita [in USD]')

plt.ylabel('Life Expectancy [in years]')

plt.title('World Development in 2007')

plt.xticks([1000, 10000, 100000],['1k', '10k', '100k'])

**# Display the plot**

plt.show()

How did we make the list col you ask? The Gapminder data contains a list continent with the continent each country belongs to. A dictionary is constructed that maps continents onto colors:

dict = {

'Asia':'red',

'Europe':'green',

'Africa':'blue',

'Americas':'yellow',

'Oceania':'black'

}

Here we have added some additional customization which is **in bold**

# Scatter plot

plt.scatter(x = gdp\_cap, y = life\_exp, s = np.array(pop) \* 2, **c = col, alpha = 0.8**)

# Previous customizations

plt.xscale('log')

plt.xlabel('GDP per Capita [in USD]')

plt.ylabel('Life Expectancy [in years]')

plt.title('World Development in 2007')

plt.xticks([1000,10000,100000], ['1k','10k','100k'])

**# Additional customizations**

**plt.text(1550, 71, 'India')**

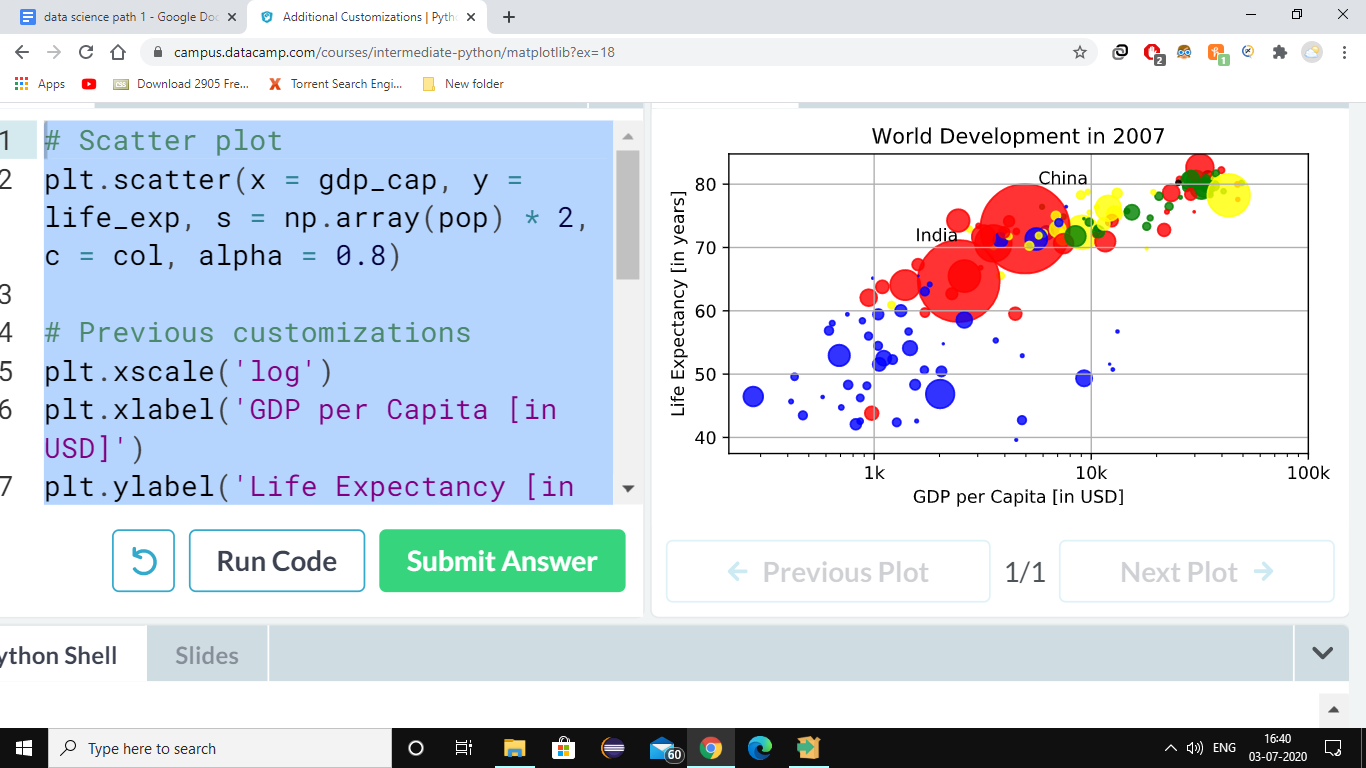
**plt.text(5700, 80, 'China')**

**# Add grid() call**

**plt.grid(True)**

# Show the plot

plt.show()



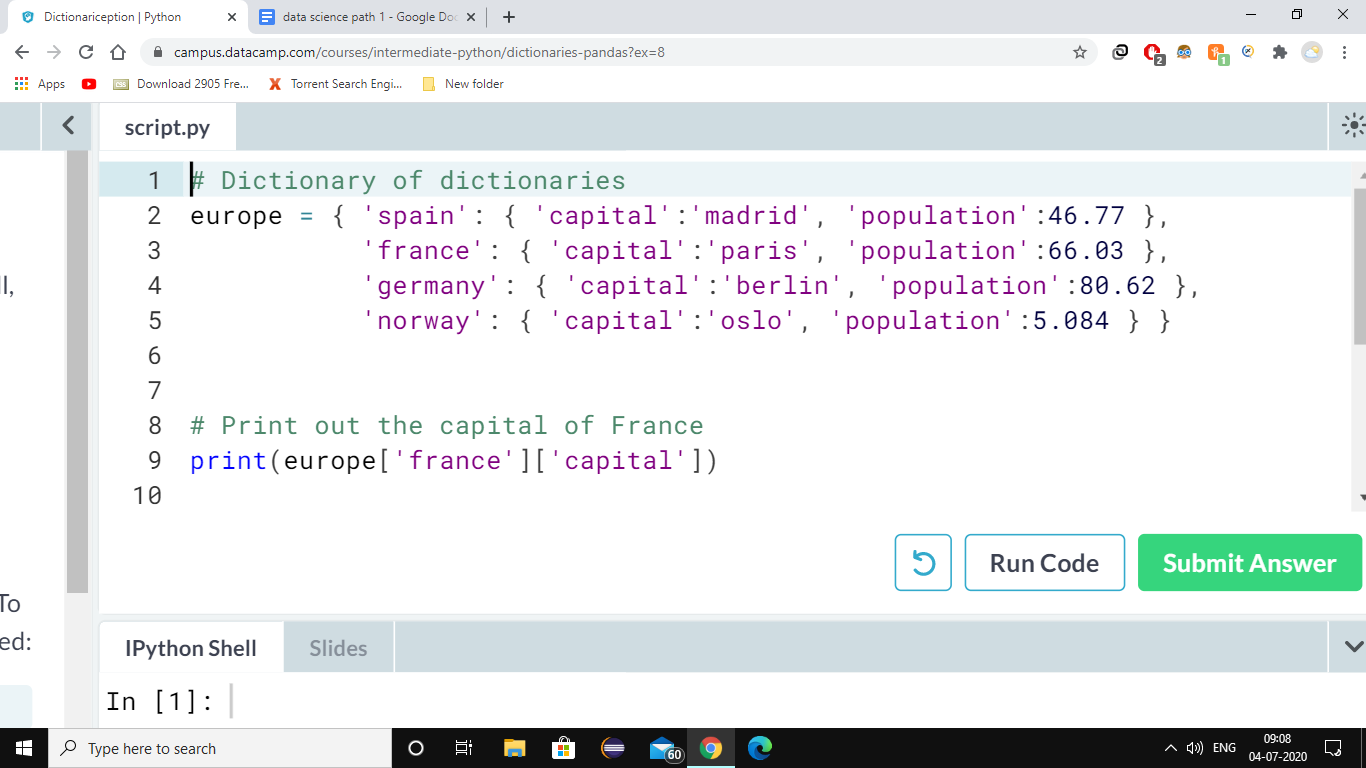
**Dictionary** have keys and values within the curly brackets which can be written as

**dict={‘keys1’:’values1’,keys2’:’values2’}**

* To extract all the elements i.e keys as well as values use **print(dict) o/p={’keys1’:’values1’,’keys2’:’values2’}**
* To extract only keys use method **keys()** with the used variable like

**dict.keys()**

* To get a particular value like values2 then **print(dict[‘keys2’]) o/p=values2**
* To get the verification whether the values nd key you have entered is there or not the use ‘in’ to verify like **‘keys2’ in dict o/p True**
* Want to update a key then use **dict[‘keys2’]=’values3’**
* Removal use **del(dict[‘keys2’])**
* **chaining**



**DATA FRAMES**

**PROCEDURE**

* **Import pandas as pd.**
* **Use the pre-defined lists to create a dictionary called my\_dict. There should be three key value pairs:**
  + **key 'country' and value names.**
  + **key 'drives\_right' and value dr.**
  + **key 'cars\_per\_cap' and value cpc.**
* **Use** [**pd.DataFrame()**](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html) **to turn your dict into a DataFrame called cars.**
* **Print out cars and see how beautiful it is.**

**# Pre-defined lists**

names = ['United States', 'Australia', 'Japan', 'India', 'Russia', 'Morocco', 'Egypt']

dr = [True, False, False, False, True, True, True]

cpc = [809, 731, 588, 18, 200, 70, 45]

**# Import pandas as pd**

import pandas as pd

**# Create dictionary my\_dict with three key:value pairs: my\_dict**

my\_dict={'country':names,'drives\_right':dr,'cars\_per\_cap':cpc}

**# Build a DataFrame cars from my\_dict: cars**

cars=pd.DataFrame(my\_dict)

**# Print cars**

print(cars)

**# Definition of row\_labels**

row\_labels = ['US', 'AUS', 'JPN', 'IN', 'RU', 'MOR', 'EG']

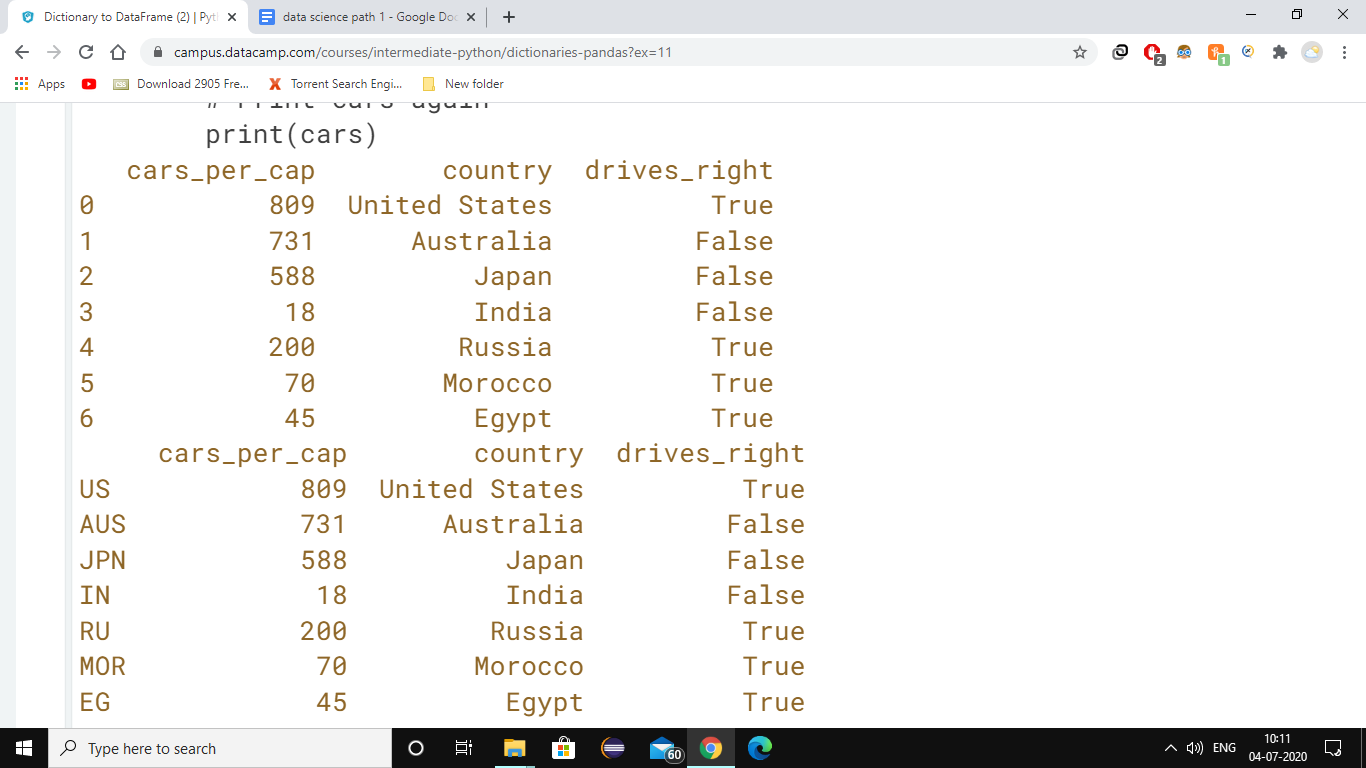
**# Specify row labels of cars**

cars.index=row\_labels

**# Print cars again**

print(cars)

**O/P**



To import CSV data into Python as a Pandas DataFrame you can use [**read\_csv()**](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html).

Let's explore this function with the same cars data from the previous exercises. This time, however, the data is available in a CSV file, named cars.csv. It is available in your current working directory, so the path to the file is simply 'cars.csv'.

* **To import CSV files you still need the pandas package: import it as pd.**
* **Use** [**pd.read\_csv()**](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html) **to import cars.csv data as a DataFrame. Store this dataframe as cars.**
* **Print out cars**

**# Import pandas as pd**

import pandas as pd

**# Import the cars.csv data: cars**

cars=pd.read\_csv(**'cars.csv'**

**,index\_col**)

**# Print out cars**

print(cars)

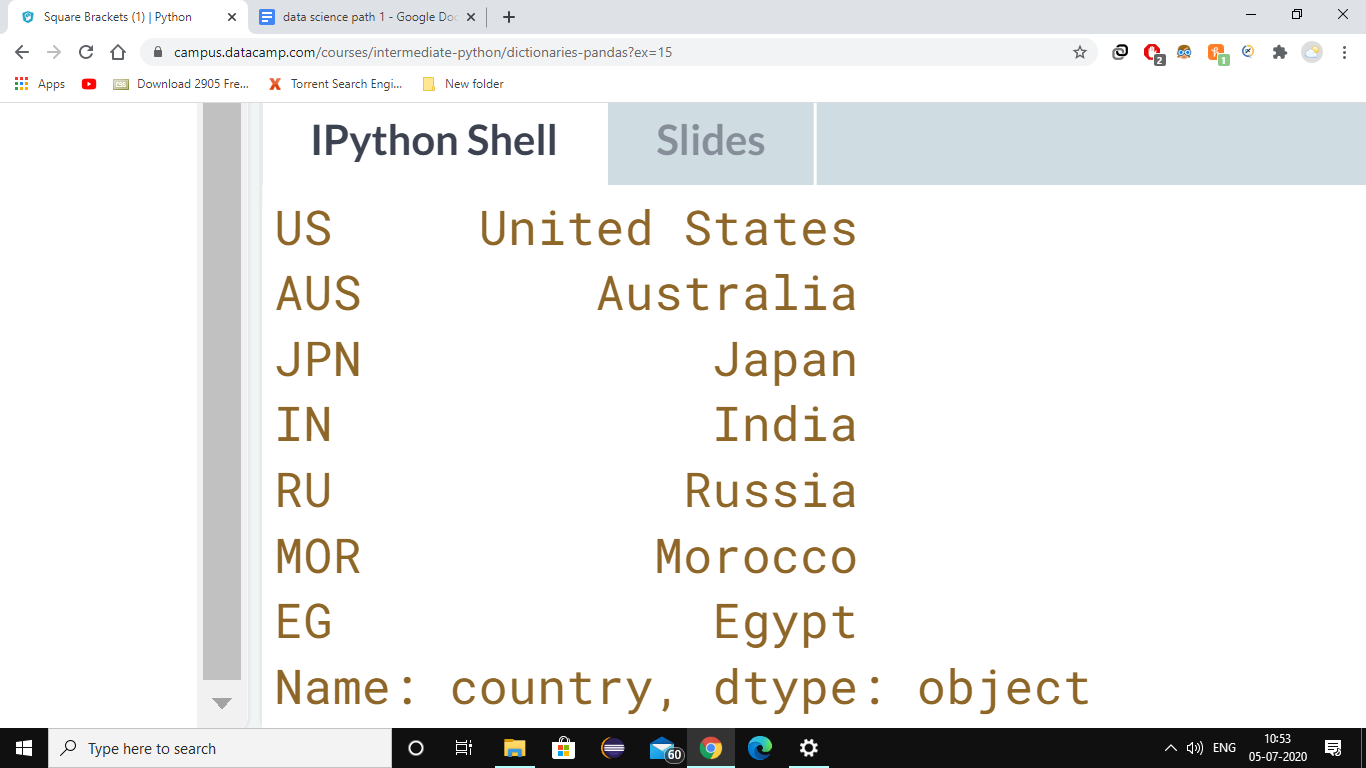
**# Import cars data**

import pandas as pd

cars = pd.read\_csv('cars.csv', index\_col = 0)

**# Print out country column as Pandas Series**

print(cars['country'])



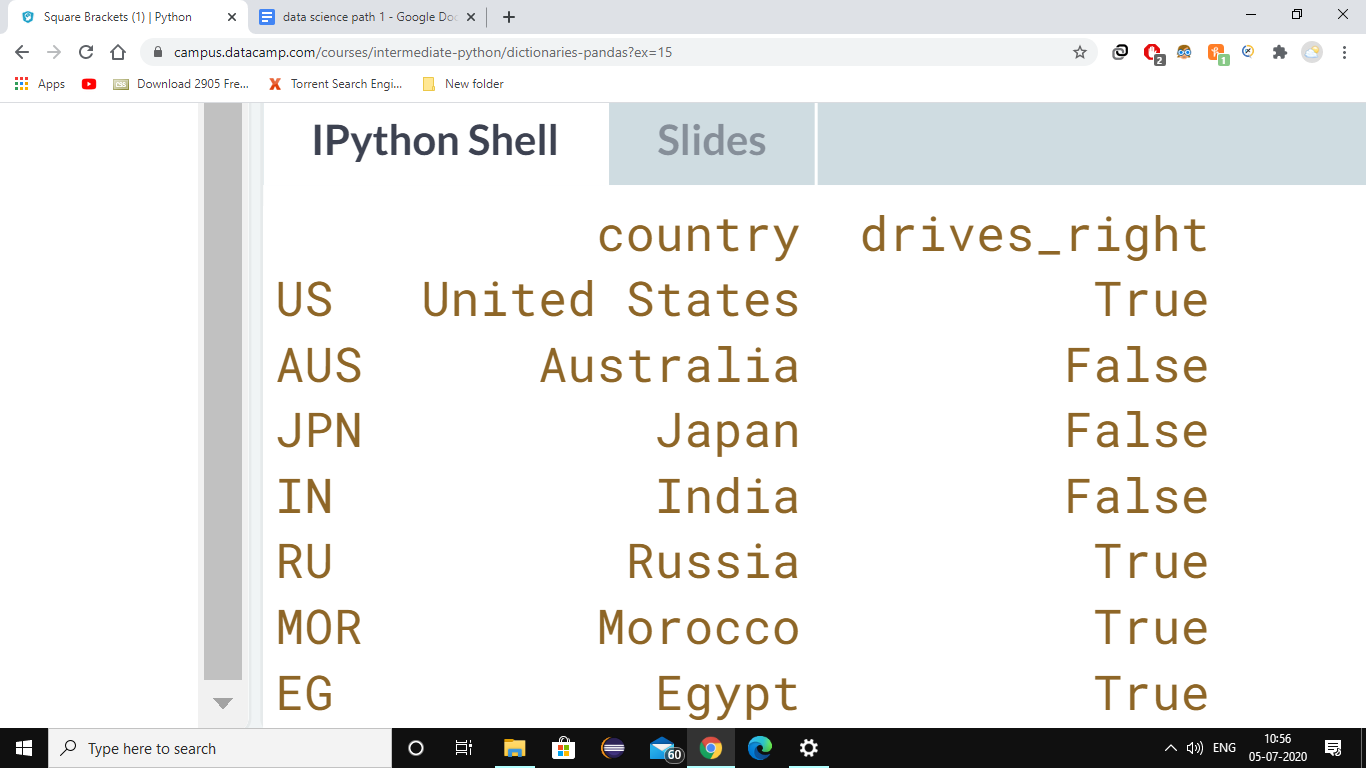
**# Print out country column as Pandas DataFrame**

print(cars[['country']])

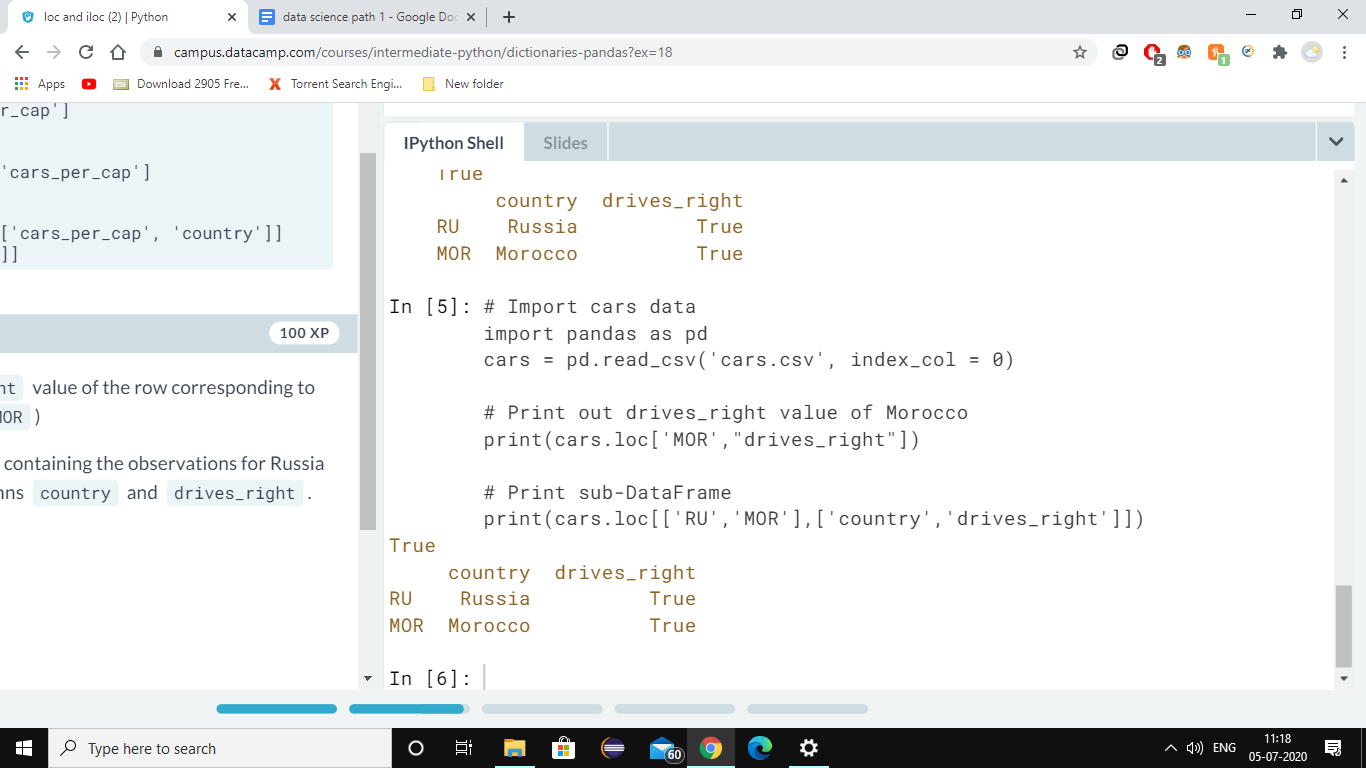


**# Print out DataFrame with country and drives\_right columns**

print(cars[['country','drives\_right']])



**loc** or **iloc**

****

**Selecting columns from loc or iloc**

**# Import cars data**

import pandas as pd

cars = pd.read\_csv('cars.csv', index\_col = 0)

**# Print out drives\_right column as Series**

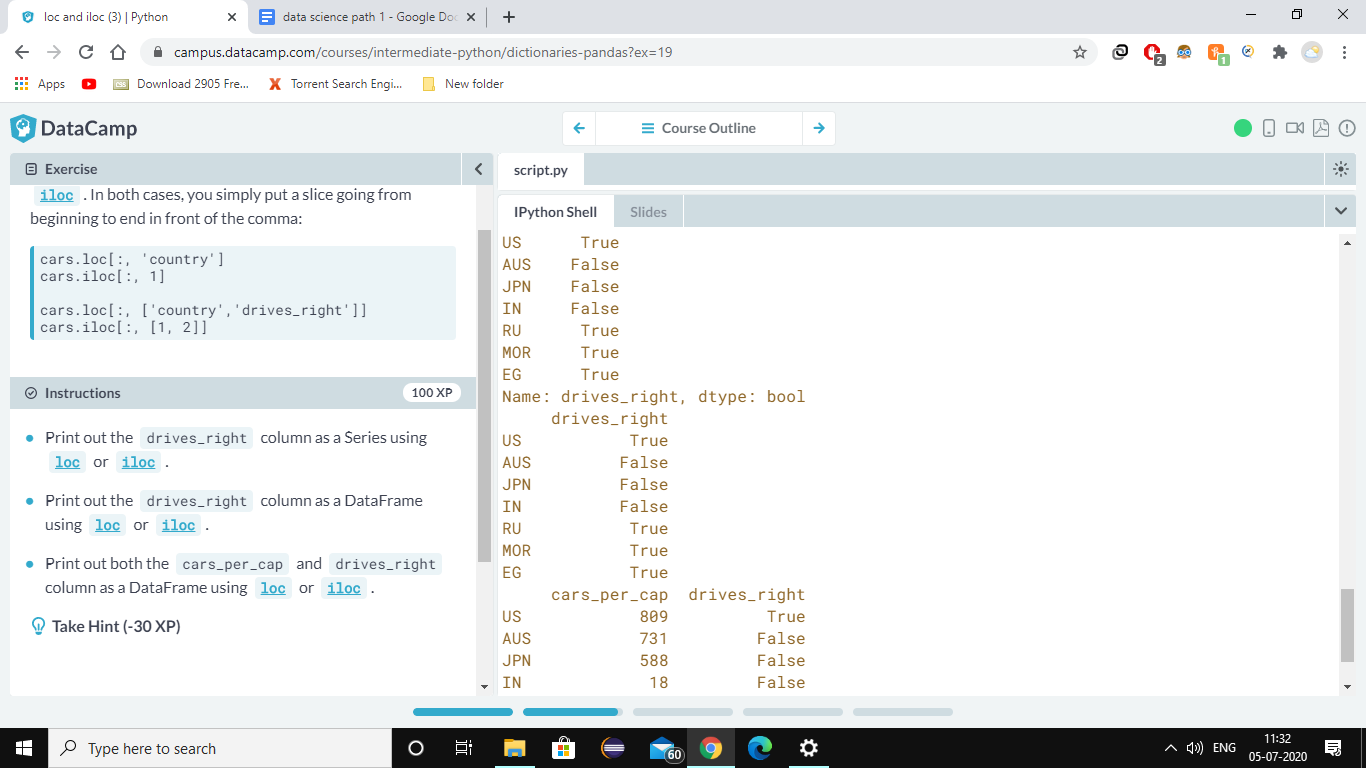
print(cars.loc[:,'drives\_right'])

**# Print out drives\_right column as DataFrame**

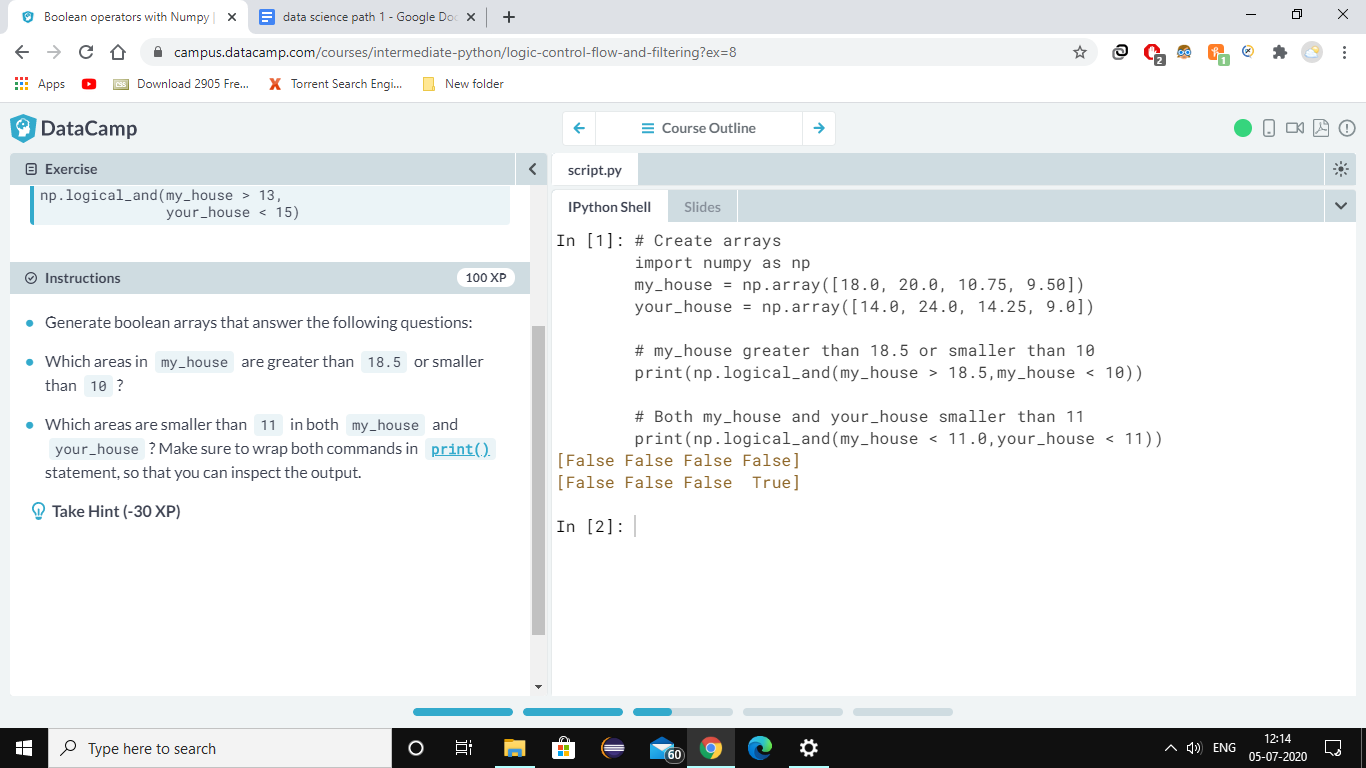
print(cars.iloc[:,[2]])

**# Print out cars\_per\_cap and drives\_right as DataFrame**

print(cars.iloc[:,[0,2]])



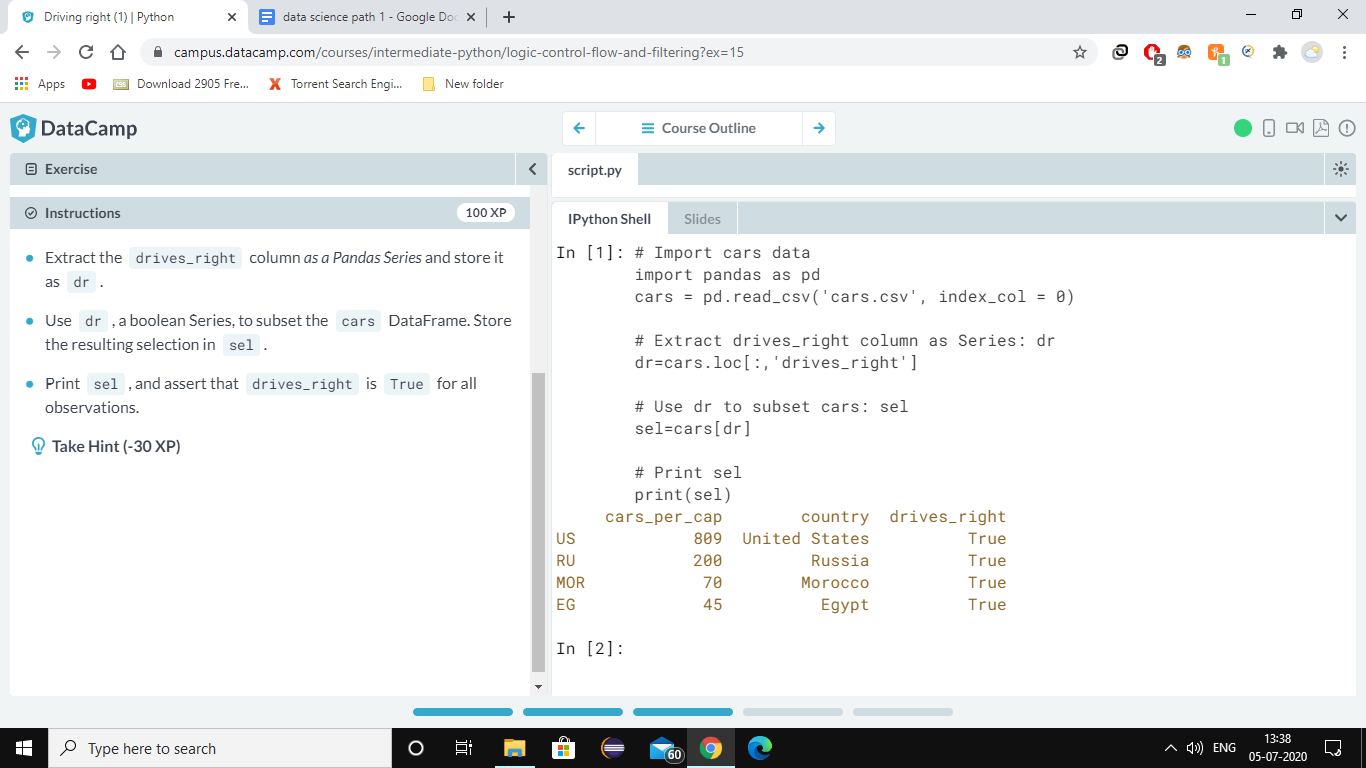
# **Boolean operators with Numpy**



**Ques. Driving right**

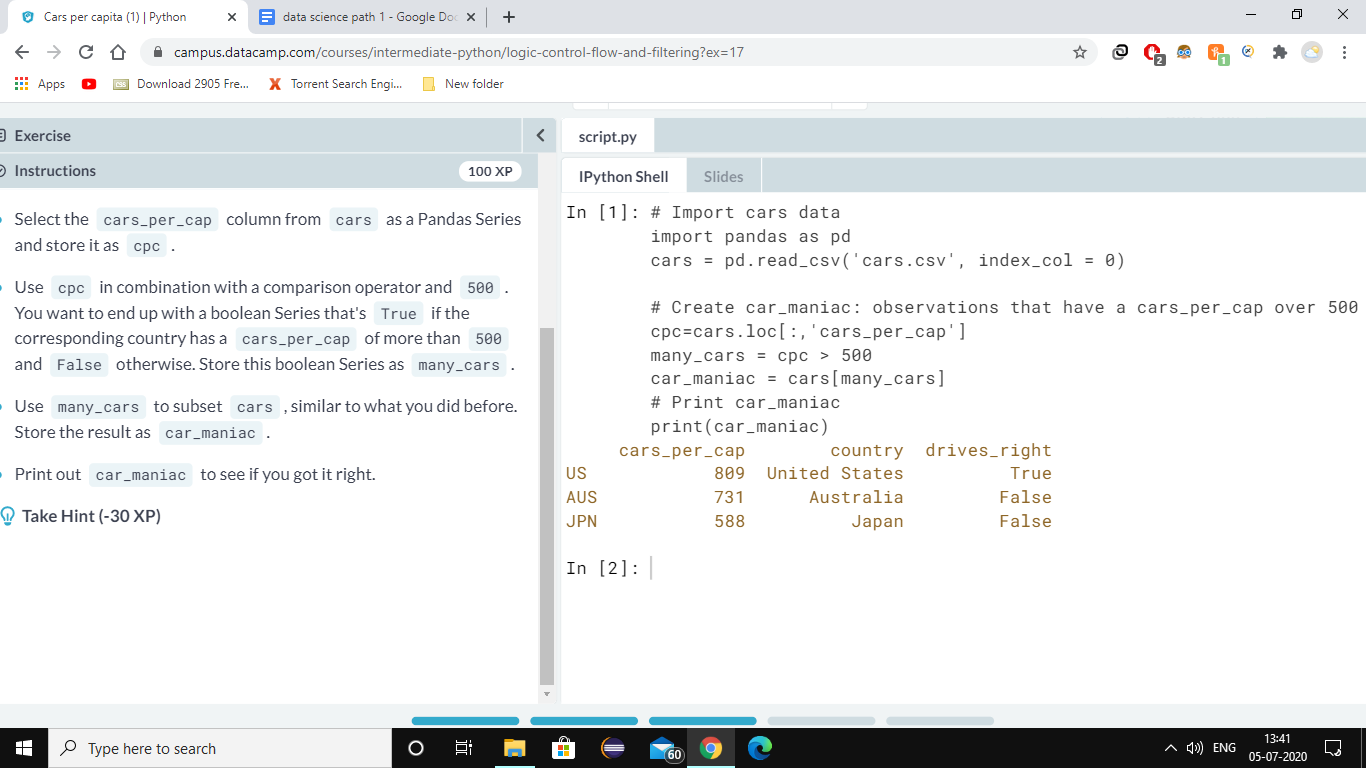
**To filter observations from a DataFrame based on boolean arrays. Let's start simple and try to find all observations in cars where drives\_right is True.**

**drives\_right is a boolean column, so you'll have to extract it as a Series and then use this boolean Series to select observations from cars.**

****

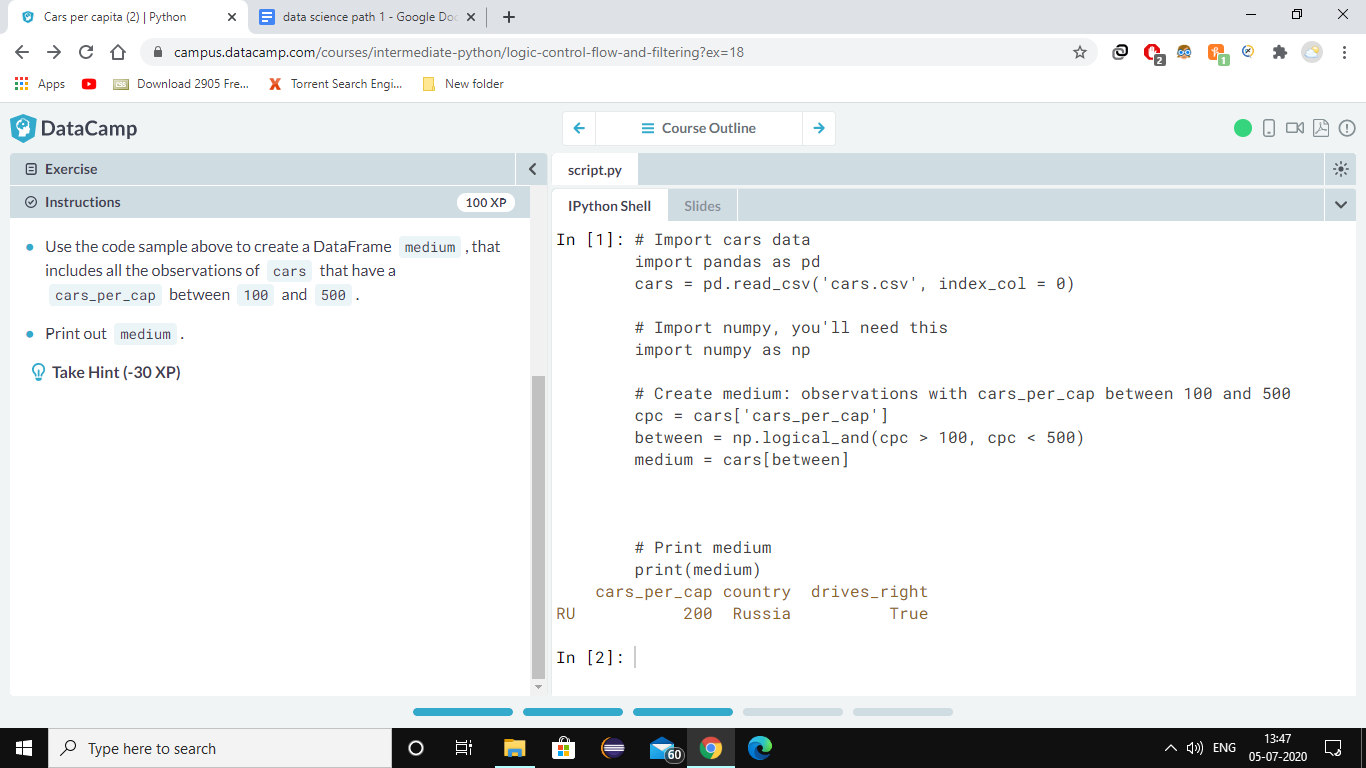
**Ques cars per capita**

**Find out which countries have a high *cars per capita* figure. In other words, in which countries do many people have a car, or maybe multiple cars.**

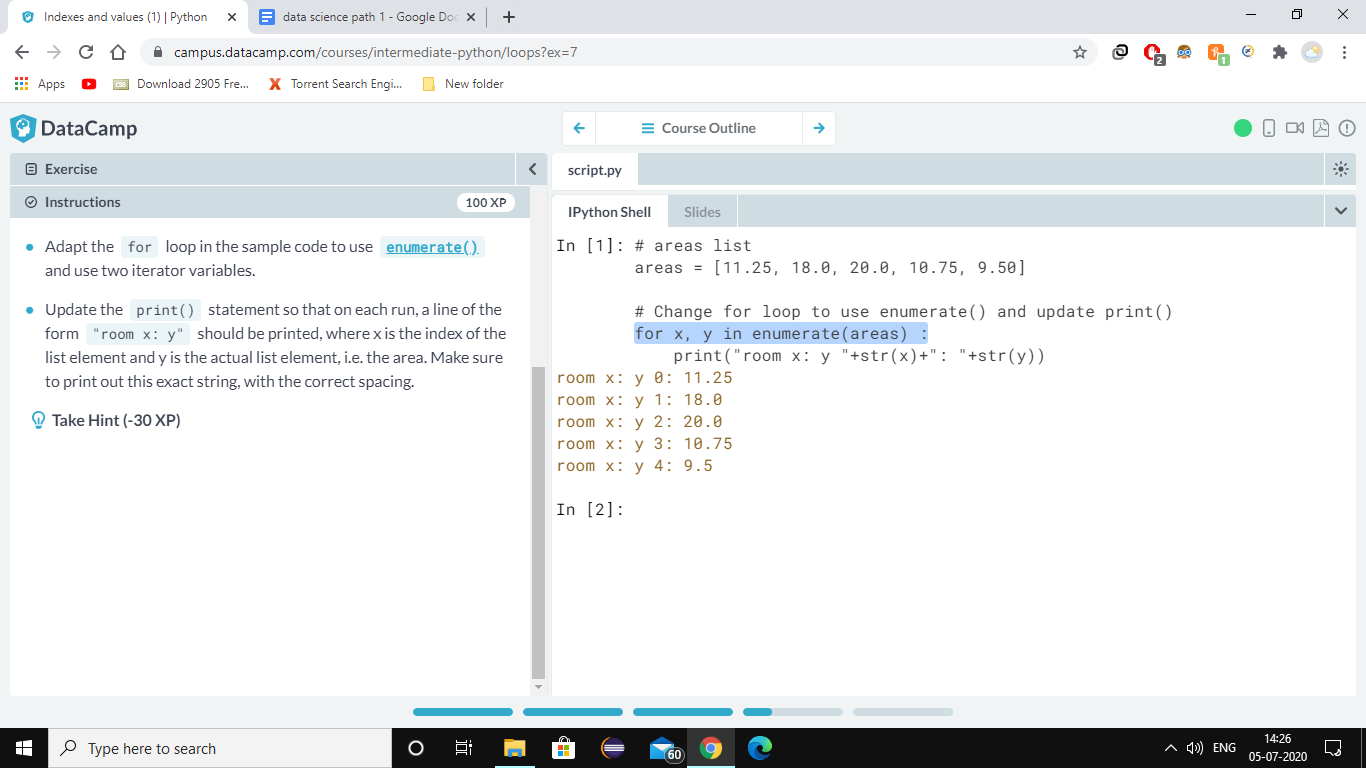
****

**Ques**

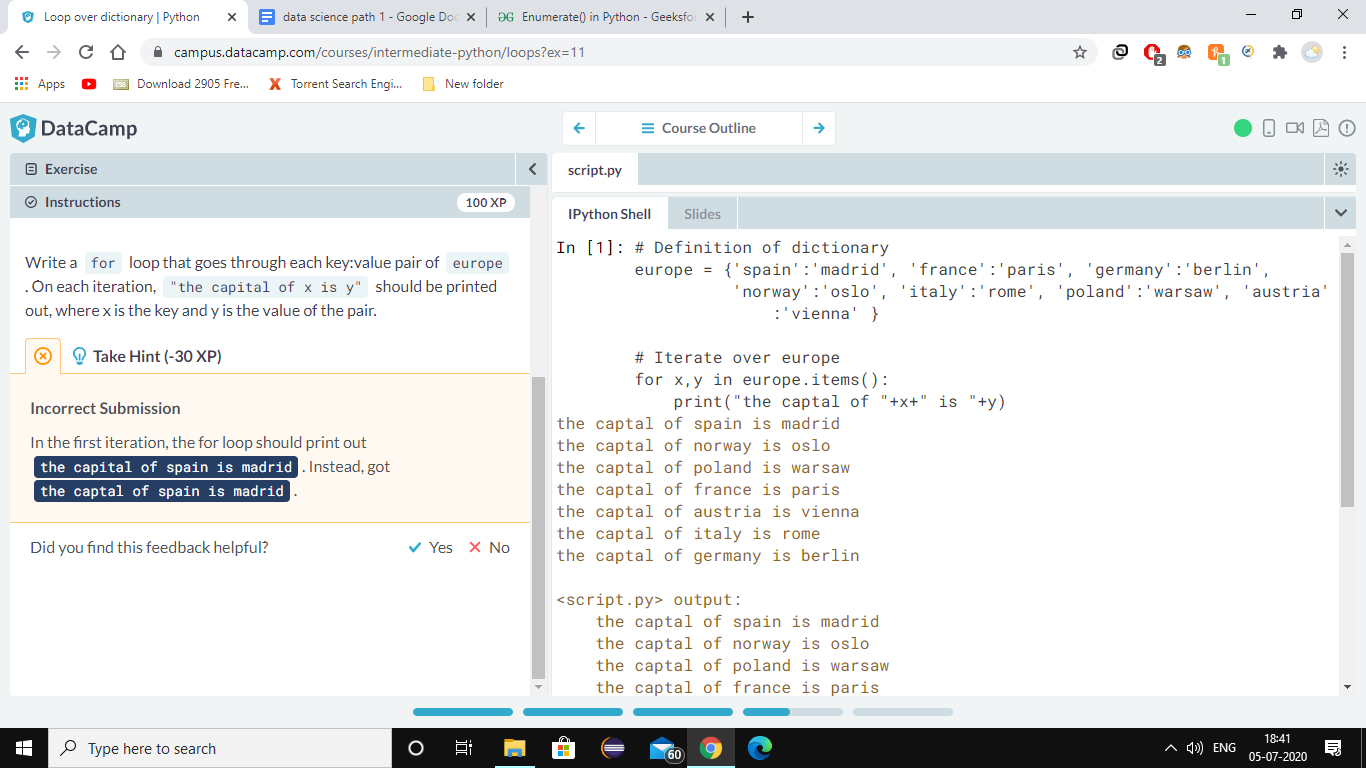
**Select observations that have a cars\_per\_cap between 10 and 80. Try out these lines of code step by step to see what's happening.**

****

**Enumerate!**

****

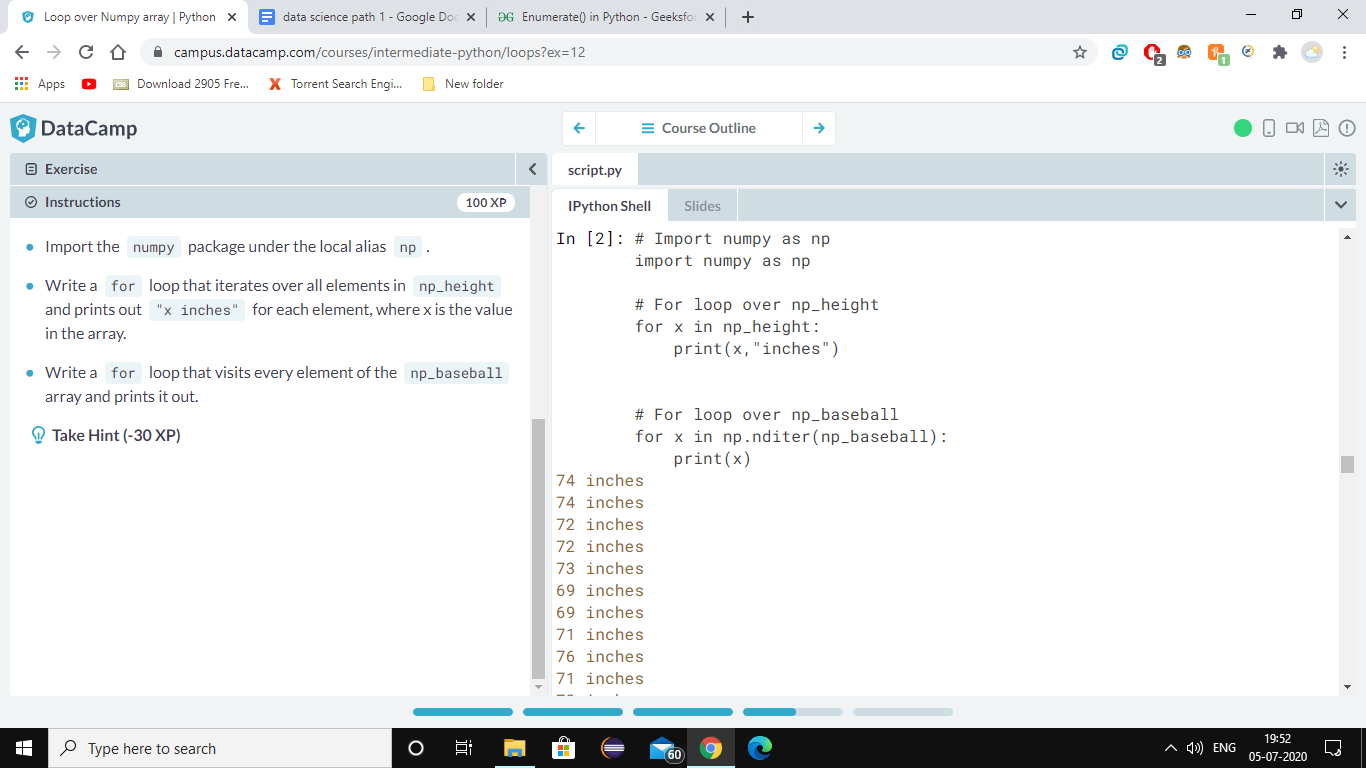
**For n Dictionary**

****

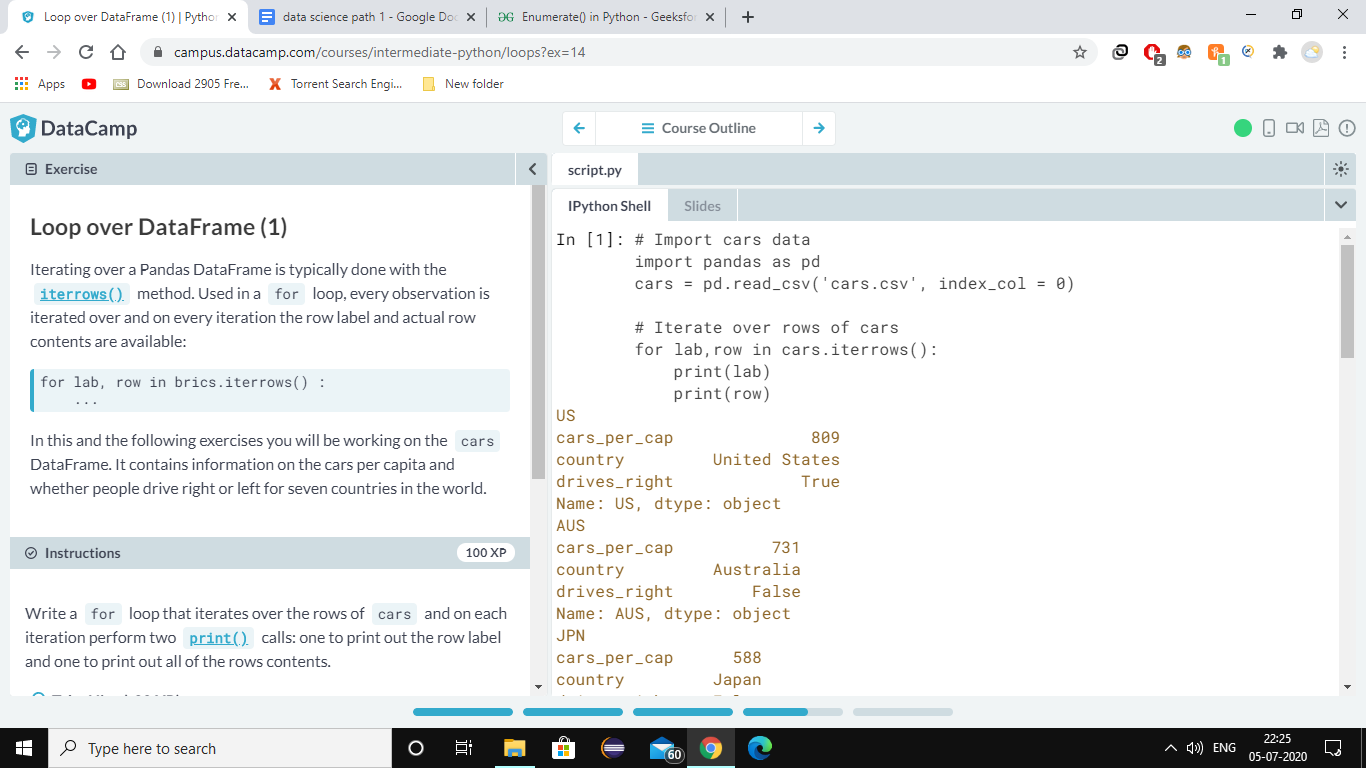
**For n Numpy array**

**For multidimensional array u need this syntax for printing array**

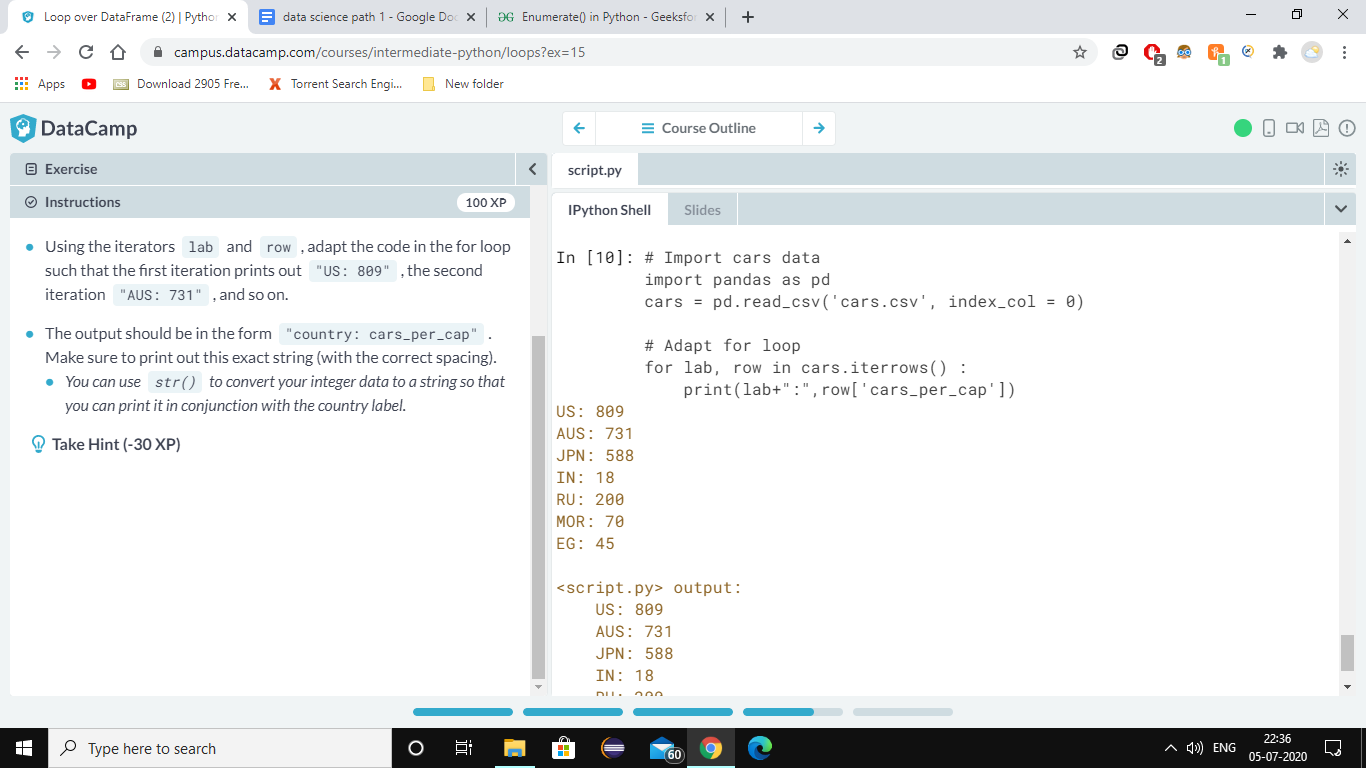
**Syntax: for x in np.nditter(xyz):**



For n dataframe

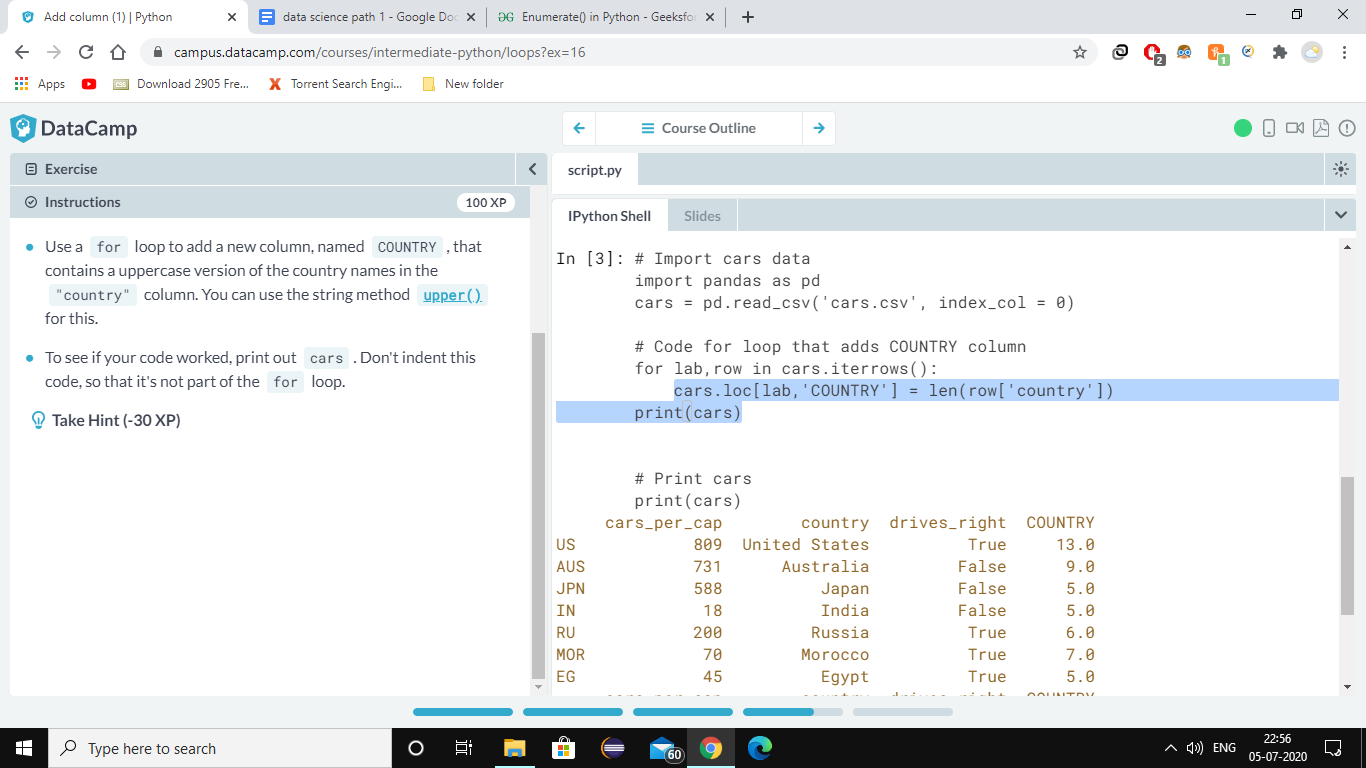


**Print selected outcomes**

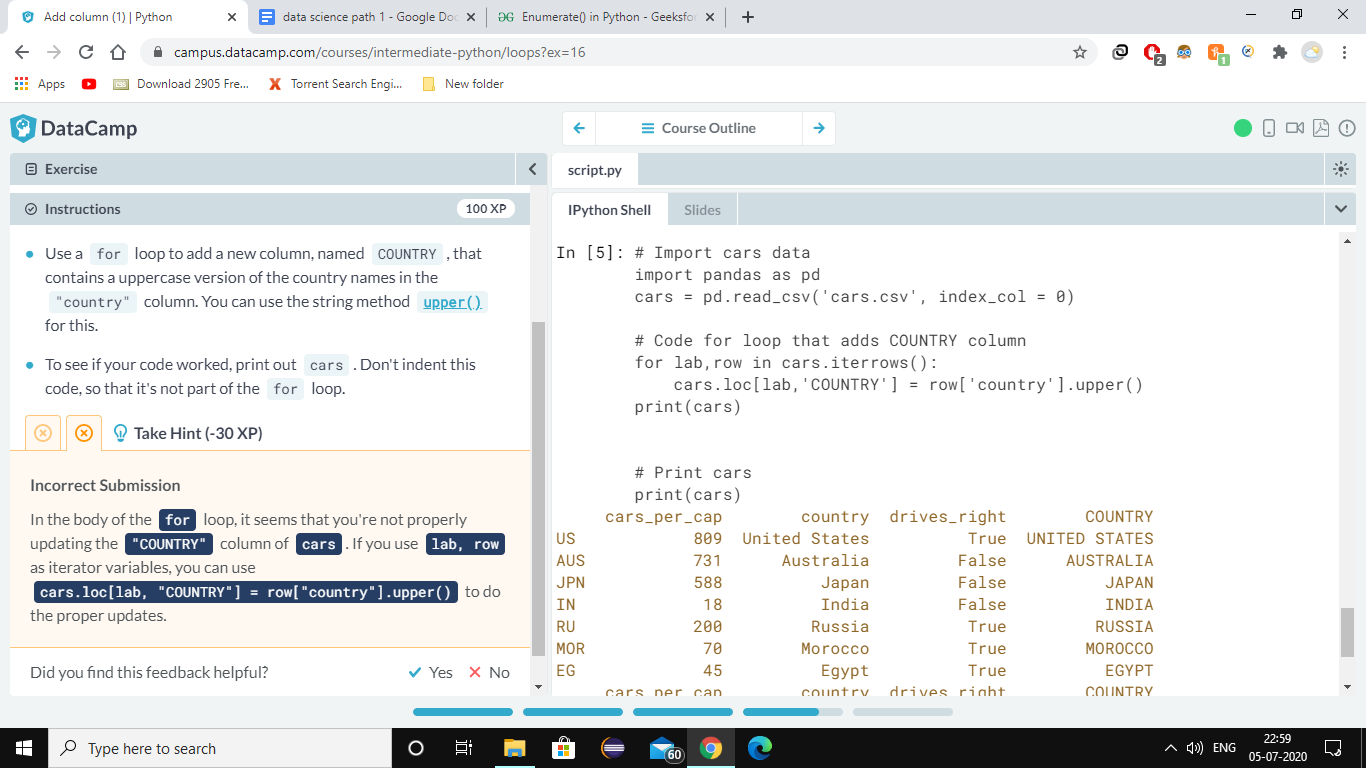
****

**Remember** for the , &+ using inside the print method + is used when u want to add two strings or integer ,but it will not work in integer and string, until we parse integer to string using **str()** function. For , it will just for printing different kind of data types at once.

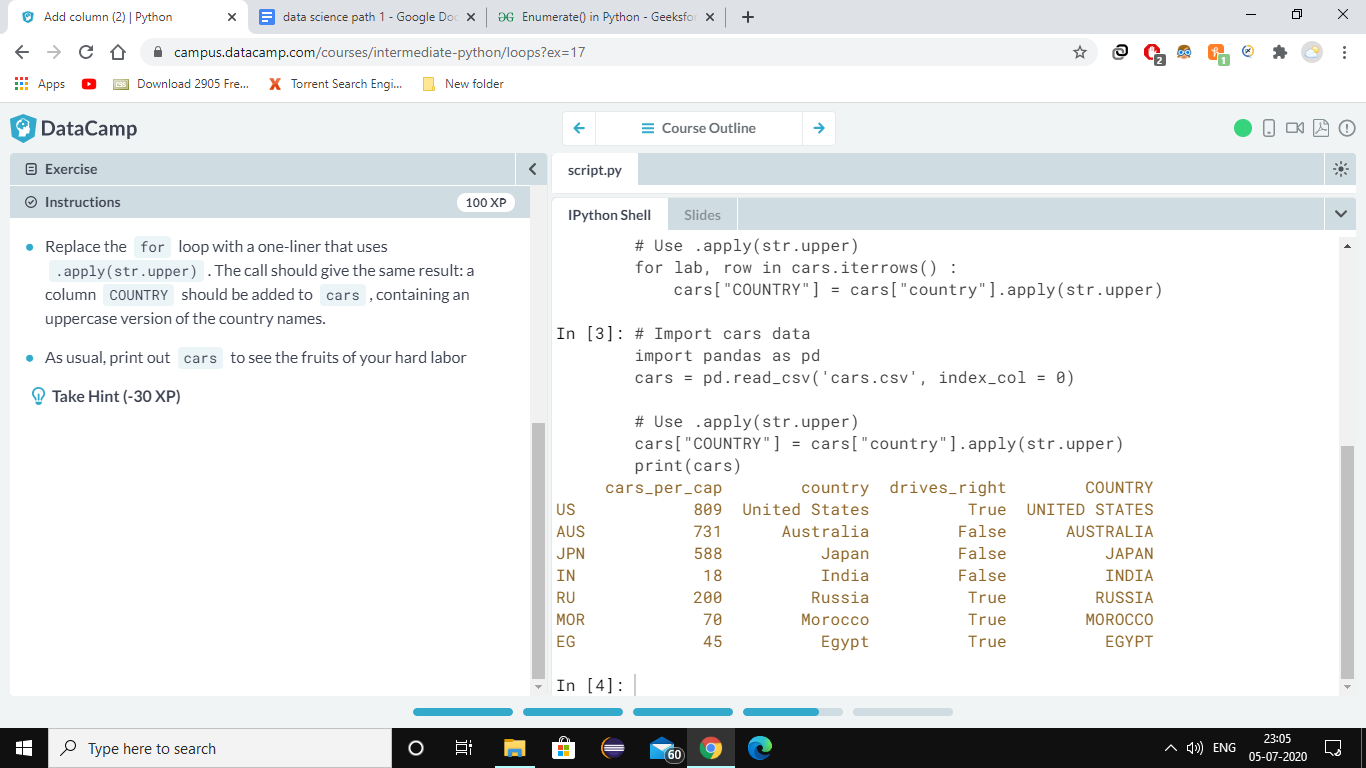
**ADD** new column



NOW new column in upper case



**ONE LINER**

****

[**https://youtu.be/lP1hb4BFAWA**](https://youtu.be/lP1hb4BFAWA) **krish naik road map**