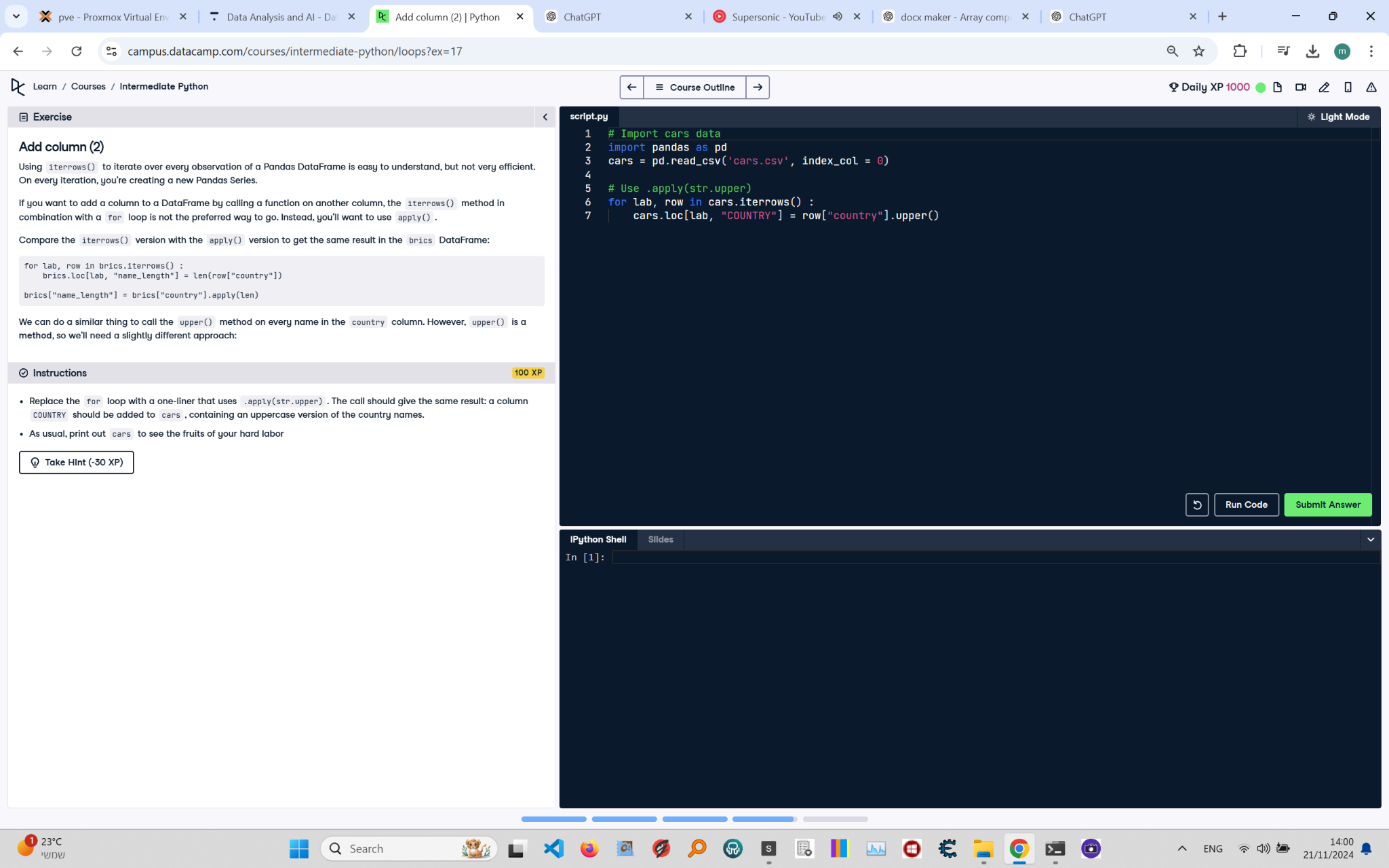
# Add Column (2) to DataFrame in Python



\*\*Question:\*\*

Using `iterrows()` to iterate over every observation of a Pandas DataFrame is easy to understand, but not very efficient. On every iteration, you're creating a new Pandas Series.  
  
If you want to add a column to a DataFrame by calling a function on another column, the `iterrows()` method in combination with a `for` loop is not the preferred way to go. Instead, you'll want to use `apply()`.  
  
We can do a similar thing to call the `upper()` method on every name in the `country` column. However, `upper()` is a method, so we'll need a slightly different approach:  
  
```python  
brics['name\_length'] = brics['country'].apply(len)  
```  
  
\*\*Instructions:\*\*  
1. Replace the `for` loop with a one-liner that uses `apply(str.upper)`. The call should give the same result: a column `COUNTRY` should be added to `cars`, containing an uppercase version of the country names.  
2. As usual, print out `cars` to see the fruits of your hard labor.

\*\*Answer:\*\*

Here is the Python code that solves the problem:

# Import pandas  
import pandas as pd  
  
# Load the cars DataFrame  
cars = pd.read\_csv('cars.csv', index\_col=0)  
  
# Add a new column 'COUNTRY' using apply and str.upper  
cars['COUNTRY'] = cars['country'].apply(str.upper)  
  
# Print the updated DataFrame  
print(cars)

\*\*Explanation:\*\*

1. \*\*Importing pandas\*\*: The `pandas` library is imported to handle the DataFrame.  
2. \*\*Loading the DataFrame\*\*: The `cars` DataFrame is loaded from a CSV file (`cars.csv`) with the first column set as the index using `index\_col=0`.  
3. \*\*Adding a new column\*\*:  
 - The `apply()` method is used on the `country` column to apply the `str.upper` function.  
 - This converts each value in the `country` column to uppercase and stores it in a new column named `COUNTRY`.  
4. \*\*Printing the updated DataFrame\*\*: Finally, the updated DataFrame is printed to verify the addition of the new column.