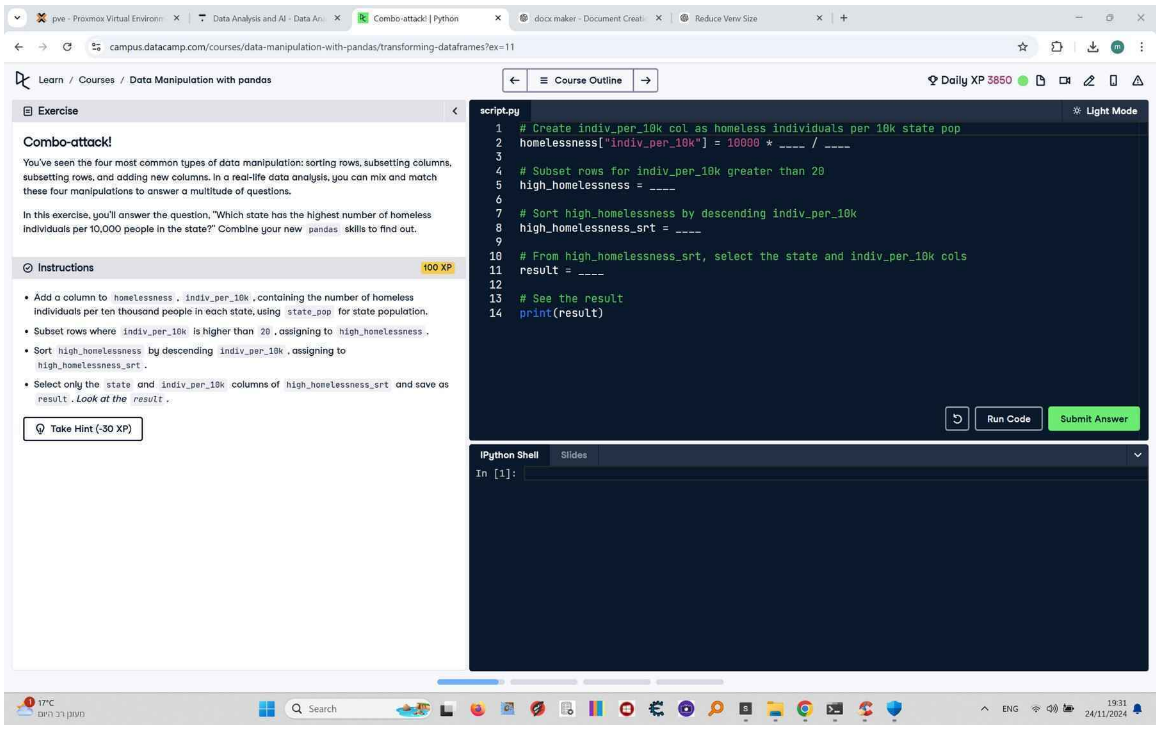
# Combo-Attack! Combining Multiple Data Manipulations

This document includes the question, the solution, and a breakdown of the code provided in the screenshot.

## Uploaded Screenshot

Below is the screenshot of the task:



Learn / Courses / Data Manipulation with pandas

Exercise

### Combo-attack!

You've seen the four most common types of data manipulation: sorting rows, subsetting columns, subsetting rows, and adding new columns. In a real-life data analysis, you can mix and match these four manipulations to answer a multitude of questions.

In this exercise, you'll answer the question, "Which state has the highest number of homeless individuals per 10,000 people in the state?" Combine your new `pandas` skills to find out.

Instructions 100 XP

- Add a column to `homelessness`, `indiv_per_10k`, containing the number of homeless individuals per ten thousand people in each state, using `state_pop` for state population.
- Subset rows where `indiv_per_10k` is higher than `20`, assigning to `high_homelessness`.
- Sort `high_homelessness` by descending `indiv_per_10k`, assigning to `high_homelessness_srt`.
- Select only the `state` and `indiv_per_10k` columns of `high_homelessness_srt` and save as `result`. *Look at the* `result`.

script.py

```python
# Create indiv_per_10k col as homeless individuals per 10k state pop
homelessness["indiv_per_10k"] = 10000 * ____ / ____

# Subset rows for indiv_per_10k greater than 20
high_homelessness = ____

# Sort high_homelessness by descending indiv_per_10k
high_homelessness_srt = ____

# From high_homelessness_srt, select the state and indiv_per_10k cols
result = ____

# See the result
print(result)
```

```
In [1]:
```

## Question

1. Add a column to `homelessness`, `indiv_per_10k`, containing the number of homeless individuals per 10,000 people in each state, using `state_pop` for state population.
2. Subset rows where `indiv_per_10k` is higher than 20, assigning to `high_homelessness`.
3. Sort `high_homelessness` by descending `indiv_per_10k`, assigning to `high_homelessness_srt`.
4. Select only the `state` and `indiv_per_10k` columns of `high_homelessness_srt` and save as `result`. View the `result`.

## Answer

# Create indiv_per_10k col as homeless individuals per 10k state pop
homelessness['indiv_per_10k'] = 10000 * homelessness['individuals'] / homelessness['state_pop']

```
# Subset rows for indiv_per_10k greater than 20
high_homelessness = homelessness[homelessness['indiv_per_10k'] > 20]

# Sort high_homelessness by descending indiv_per_10k
high_homelessness_srt = high_homelessness.sort_values('indiv_per_10k',
ascending=False)

# From high_homelessness_srt, select the state and indiv_per_10k cols
result = high_homelessness_srt[['state', 'indiv_per_10k']]

# See the result
print(result)
```

## Code Explanation

# Explanation of the code:

1. `homelessness['indiv_per_10k'] = 10000 * homelessness['individuals'] / homelessness['state_pop']`: Adds a new column `indiv_per_10k` to the `homelessness` DataFrame, calculating the number of homeless individuals per 10,000 people in each state.

2. `homelessness[homelessness['indiv_per_10k'] > 20]`: Subsets the rows where `indiv_per_10k` is greater than 20 and assigns the result to `high_homelessness`.

3. `high_homelessness.sort_values('indiv_per_10k', ascending=False)`: Sorts the `high_homelessness` DataFrame in descending order based on the `indiv_per_10k` column and assigns it to `high_homelessness_srt`.

4. `high_homelessness_srt[['state', 'indiv_per_10k']]`: Selects the `state` and `indiv_per_10k` columns from the `high_homelessness_srt` DataFrame and assigns the result to `result`.

5. `print(result)`: Prints the resulting DataFrame to verify the calculations and filtering.