

Setting Multi-Level Indexes

The screenshot shows a web browser window displaying a DataCamp course page. The browser's address bar shows the URL: `campus.datacamp.com/courses/data-manipulation-with-pandas/slicing-and-indexing-dataframes?ex=4`. The page title is "Setting multi-level indexes | Py". The course navigation bar includes "Learn / Courses / Data Manipulation with pandas" and a "Course Outline" button. The main content area is titled "Exercise" and "Setting multi-level indexes". It contains a text block explaining multi-level indexes, a "pandas is loaded as pd. temperatures is available." note, and a list of instructions. The instructions are: 1. Set the index of 'temperatures' to the 'country' and 'city' columns, and assign this to 'temperatures_ind'. 2. Specify two country/city pairs to keep: 'Brazil' / 'Rio De Janeiro' and 'Pakistan' / 'Lahore', assigning to 'rows_to_keep'. 3. Print and subset 'temperatures_ind' for 'rows_to_keep' using '.loc[]'. A "Take Hint (-30 XP)" button is below the instructions. On the right, a code editor shows a Python script:

```
1 # Index temperatures by country & city
2 temperatures_ind = ____
3
4 # List of tuples: Brazil, Rio De Janeiro & Pakistan, Lahore
5 rows_to_keep = [____]
6
7 # Subset for rows to keep
8 print(temperatures_ind.____)
```

 Below the code editor is an "IPython Shell" with "In [1]:" and a "Run Code" button. The bottom of the screen shows a Windows taskbar with the date 25/11/2024 and time 16:56.

Indexes can also be made out of multiple columns, forming a multi-level index (sometimes called a hierarchical index). There is a trade-off to using these.

The benefit is that multi-level indexes make it more natural to reason about nested categorical variables. For example, in a clinical trial, you might have control and treatment groups. Then each test subject belongs to one or another group, and we can say that a test subject is nested inside the treatment group. Similarly, in the temperature dataset, the city is located in the country, so we can say a city is nested inside the country.

The main downside is that the code for manipulating indexes is different from the code for manipulating columns, so you have to learn two syntaxes and keep track of how your data is represented.

pandas is loaded as `pd`. `temperatures` is available.

Final Answer

```
# Index temperatures by country & city
temperatures_ind = temperatures.set_index(["country", "city"])
```

```
# List of tuples: Brazil, Rio De Janeiro & Pakistan, Lahore
rows_to_keep = [("Brazil", "Rio De Janeiro"), ("Pakistan", "Lahore")]

# Subset for rows to keep
print(temperatures_ind.loc[rows_to_keep])
```