MODULE 05 - Piscine Python for Data Science

Pandas: working with Dataframes

Summary: This day will help you acquire skills with Pandas.

Contents

Chapter I

Foreword

Fun facts about pandas:

- Pandas are big eaters – every day they fill their tummies for up to 12 hours, shifting up to 12 kilograms of bamboo

- Unlike most other bears, pandas do not hibernate. When winter approaches, they head lower down their mountain homes to warmer temperatures, where they continue to chomp away on bamboo

- Sadly, these beautiful bears are endangered, and it's estimated that only around 1,000 remain in the wild

- On average, pandas poo 40 times a day

- According to legend, the panda was once an all-white bear. When a small girl tried to save a panda cub from being attacked by a leopard, the leopard killed the girl instead. Pandas came to her funeral wearing armbands of black ashes. As they wiped their eyes, hugged each other, and covered the ears, they smudged the black ashes

- A panda's entire mating process takes only about two or three days. Once they have mated, females chase the males out of their territory and raise their cubs on their own

- A giant panda usually gives birth to a single cub. Sometimes twins are born, but when this happens, the mother typically ignores the weaker cub. She does not have enough energy to care for two cubs

- A giant panda's face is cute, but it is not chubby. It gets its shape from massive cheek muscles

- Keeping even a single panda in a zoo is expensive. A panda costs five times more to keep than the next most expensive animal, an elephant

- If you think that any of these is relevant to the library Pandas, please do not. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals

Chapter II

Instructions

- Use this page as the only reference. Do not listen to any rumors and speculations about how to prepare your solution.

- Here and further we use Python 3 as the only correct version of Python.

- The python files for python exercises (module01, module02, module03) must have a block in the end: if ___name___ == '___main___'.

- Pay attention to the permissions of your files and directories.

- To be assessed your solution must be in your GIT repository.

- Your solutions will be evaluated by your piscine mates.

- You should not leave in your directory any other file than those explicitly specified by the exercise instructions. It is recommended that you modify your .gitignore to avoid accidents.

- When you need to get precise output in your programs, it is forbidden to display a precalculated output instead of performing the exercise correctly.

- Have a question? Ask your neighbor on the right. Otherwise, try with your neighbor on the left.

- Your reference manual: mates / Internet / Google.

- Remember to discuss on the Intra Piscine forum.

- Read the examples carefully. They may require things that are not otherwise specified in the subject.

- And may the Force be with you!

Chapter III

Specific instructions of the day

- Use Jupyter Notebook to work with your code

- For each major subtask in the list of any exercise (black bullets), your ipynb file should have an h2 heading to help your peer easily navigate in your code

- No imports allowed, except those explicitly mentioned in the section "Authorized functions" of the title block of each exercise

- You can use any built-in function, if it is not prohibited in the exercise

- Save and load all the required data in the subfolder data/

Chapter IV

Exercice 00 : Load and save

	Exercise 00
	Load and save
Turn-in directory : *ex00/*	
Files to turn in : `load_and_save.ipynb`	
Allowed functions : `import pandas as pd`	

Congratulations! Five days and one rush are behind your back. They were dedicated to building good fundamental skills in working with Python. Now you know all basic data types, useful built-in functions, how to work with a virtual environment, how to use OOP, how to use logging, how to parse data from websites, and how to send messages in a Slack channel. In the following week, we will be more focused on data science. This day will be full of Pandas exercises. Pandas is one of the most popular and useful libraries in the field. You will love it too!

In this exercise, you will need to load the log file (put it in the directory data in the root directory of the day) into a dataframe, change the delimiter, and save it to another file.

The task is:

- read_csv:

  - filter the rows with index 2 and 3 using skiprows argument, we know that these observations were fake

  - filter the last 2 rows from the footer using skipfooter argument, we know that these observations were fake too

  - assign the following names to the column: datetime, user

  - use datetime as the index column

- rename datetime to date_time

- to_csv:

  - use ';' as the delimiter

○ save it to a file with the name feed-views-semicolon.log

What you need to achieve as the result of read_csv is below:

```
In [3]: df.head()
   Out[3]:
             user
  datetime
2020-04-17 12:01:08.463179 artem
2020-04-17 12:01:23.743946 artem
2020-04-17 12:35:52.735016 artem
2020-04-17 12:36:21.401412 oksana
2020-04-17 12:36:22.023355 oksana



In [4]: df.tail()
   Out[4]:
             user
  datetime
2020-05-21 16:36:40.915488 ekaterina
2020-05-21 17:49:36.429237 maxim
2020-05-21 18:45:20.441142 valentina
2020-05-21 23:03:06.457819 maxim
2020-05-21 23:23:49.995349 pavel
```

Think about how many lines of code you would have needed to write, if you had had to do it without Pandas

Chapter V

Exercice 01 : Basic operations

	Exercise 01
	Basic operations
Turn-in directory : *ex01/*	
Files to turn in : `basic_operations.ipynb`	
Allowed functions : `import pandas as pd`	

We are sure that you understand that it is not all the things that Pandas can do for you. Let us go deeper and wider.

In this exercise, you will work with the same log of the users who visited a page and their timestamps.

- create a dataframe views with two columns: datetime and user by reading feed-views.log

  - convert the datetime to the datetime64[ns] Dtype

  - extract the year, month, day, hour, minute, second from the values of that column to the new columns

- create the new column daytime

  - you need to assign the particular daytime value if an hour is in the particular interval, for example, afternoon if the hour is larger than 11 and less or equal to 17

  - 0.00 – 03.59 night, 04.00 – 06.59 early morning, 07.00 – 10.59 morning, 11.00 – 16.59 afternoon, 17.00 – 19.59 early evening, 20.00 – 23.59 evening

  - use method cut to solve this subtask

  - assign the column user as the index

- calculate the number of elements in your dataframe

  - use the method count()

- ○ calculate the number of elements in each daytime category using the method value_counts()

- sort values in your dataframe by hour, minute and second ascendingly (simultaneously and not one by one)

- calculate the minimum, maximum for the hours and the mode for the daytime categories

  - ○ calculate the maximum of hour for the rows where the daytime is night

  - ○ calculate the minimum of hour for the rows where the daytime is morning

  - ○ additionally to this, find out who visited the the page at those hours (one example out of it)

  - ○ calculate the mode for the hour and daytime

- show the 3 smallest hours in the morning and the corresponding usernames and the 3 largest hours and the usernames using nsmallest() and nlargest()

- use the method describe() to get the basic statistics for the columns

  - ○ to find out what the most popular interval of visiting the page is, calculate the interquartile range for the hour by extracting values from the result of the describe() method and store it in the variable iqr

Chapter VI

Exercice 02 : Preprocessing

	Exercise 02
	Preprocessing
Turn-in directory : *ex02/*	
Files to turn in : `preprocessing.ipynb`	
Allowed functions : `import pandas as pd`	

One day you will train machine learning models (not further than this week), but most of them require the data to be clean: without duplicates, without missing values, enriched. Pandas is a great tool to do this. It gives you the tools not only to perform descriptive analysis and understand your data better but to preprocess it too. That is what you are going to do in this exercise.

- download and read the csv file and make ID the index column

- count the number of observations using method count()

- drop the duplicates taking into account only the following columns: CarNumber, Make_n_model, Fines

  - between the two equal observations, you need to choose the last

  - check again the number of observations

- work with missing values

  - check how many missing values are in each column

  - drop all the columns that have more than 500 missing values using the argument thresh, check how many missing values are in each column

  - replace all the missing values in the Refund column by the previous value in this column for that cell, use the argument method, check how many missing values are in each column

  - replace all the missing values in the Fines column by the mean value of this column (exclude NA/null values when computing the mean value), check how many missing values are in each column

- split and parse the make and model

  - use the method apply both for splitting and for extracting the values to the new columns Make and Model

  - drop the column Make_n_model

  - save the dataframe in the JSON file auto.json in the format below:

```
[{"CarNumber":"Y163O8161RUS","Refund":2.0,"Fines":3200.0,"Make":"Ford","Model":"Focus"},{"CarNumber":"
    E432XX77RUS","Refund":1.0,"Fines":6500.0,"Make":"Toyota","Model":"Camry"}]
```

Chapter VII

Exercice 03 : Selects and aggregations

	Exercise 03
	Selects and aggregations
Turn-in directory : *ex03/*	
Files to turn in : `selects_n_aggs.ipynb`	
Allowed functions : `import pandas as pd`	

Ok, great. Now we have our data cleaned: all the duplicates are dropped, all the missing values are deleted, columns reorganized to be more convenient for analysis. Now we can go further. We have a lot of questions that need to be answered.

- load the JSON-file that you created in the previous exercise into a dataframe

  - set CarNumber as the index column

- make the following selects

  - display the rows only where the fines are more than 2,100

  - display the rows only where the fines are more than 2,100 and the refund equals 2

  - display the rows only where the models are from the list: ['Focus', 'Corolla']

  - display the rows only where the car number is from the list: ['Y7689C197RUS', '92928M178RUS', '7788KT197RUS', 'H115YO163RUS', 'X758HY197RUS']

- make the aggregations with the make and the model

  - display the median fines grouped by the make

  - display the median fines grouped by the make and the model

  - display the number of fines grouped by the make and the model in order to understand if we can trust the median values

- display the minimum and the maximum fines grouped by the make and the model in order to better understand the variance

- display the standard deviation of the fines grouped by the make and the model in order to better understand the variance

- make the aggregations with the car number

  - display the car numbers grouped by the number of the fines in the descending order, we want to find those who most often violated the law

  - select from the initial dataframe all the rows corresponding to the top-1 car number, we want to zoom in a little bit

  - display the car numbers grouped by the sum of the fines in the descending order, we want to find those who paid the most

  - select from the initial dataframe all the rows corresponding to the top-1 car number, we want to zoom in a little bit

  - display the table that answers the question: are there any car number that was connected to different models?

Chapter VIII

Exercice 04 : Enrichment and transformations

	Exercise 04
	Enrichment and transformations

Turn-in directory : *ex*04/
Files to turn in : `enrichment.ipynb`
Allowed functions : `import pandas as pd, import numpy as np, import requests`

Cool. But the more data you have the better analysis you can conduct. Let us enrich our initial dataset.

- read the JSON file that you saved in ex02

    ○ one of the columns has the float type, so let us define the format of it in pandas using pd.options.display.float_format: floats should be displayed with two decimals

    ○ there are missing values in Model, do not do anything with them

- enrich the dataframe using a sample from that dataframe

    ○ create a sample with 200 new observations with the random_state = 21

        ∗ the sample should not have new combinations of the car number, make and model – so the whole dataset will be consistent in these terms

        ∗ the refund and fines do not have any restrictions, you can randomly take any value from these columns and use it towards any car number

    ○ concatenate the sample with the initial dataframe to a new dataframe concat_rows

- enrich the dataframe concat_rows by a new column with generated data

    ○ create a series with the name Year using random integers from 1980 to 2019

- o use np.random.seed(21) before generating the years

- o concatenate the series with the dataframe and name it fines

- enrich the dataframe by the data from another dataframe

  - o create a new dataframe with the car numbers and their owners

    - ∗ get the most popular surnames in the US

    - ∗ create a new series with the surnames (they should not have special characters like commas, brackets, etc.) from the data you gathered, the count should be equal to the number of the unique car numbers using sample (use random_state = 21)

    - ∗ create the dataframe owners with 2 columns: CarNumber and SURNAME

  - o append to the fines dataframe 5 more observations (come up with your own ideas of CarNumber, etc.)

  - o delete from the owners dataframe last 20 observations and add 3 new observations (they are not the same as those you add to the fines dataframe)

  - o join both dataframes:

    - ∗ the new dataframe should have only the car numbers that exist in both dataframes

    - ∗ the new dataframe should have all the car numbers that exist in both dataframes

    - ∗ the new dataframe should have only the car numbers from the fines dataframe

    - ∗ the new dataframe should have only the car numbers from the owners dataframe

- create a pivot table from the fines dataframe, it should look like this (the values are the sums of the fines), but only with all the years (the values can be different for you):

Make	Model	Year 1980	1981	1982	1983	1984
Dodge	Neon	nan	nan	nan	nan	nan
Ford	Focus	89,194.59	266,783.76	107,283.76	147,289.17	106,000.00
	Mondeo	nan	nan	46,200.00	nan	nan
Honda	CRV	nan	nan	nan	nan	nan
Hyundai	Solaris	nan	nan	nan	nan	nan
Lada	Kalina	nan	nan	nan	nan	nan
Merc	S300	nan	nan	nan	nan	nan
Skoda	Octavia	13,794.59	1,900.00	8,894.59	nan	1,300.00
Toyota	Camry	12,000.00	nan	1,000.00	8,594.59	1,000.00
	Corolla	nan	6,800.00	nan	12,800.00	nan
Volkswagen	Golf	20,800.00	8,594.59	5,000.00	200.00	nan
	Jetta	nan	1,000.00	nan	nan	nan
	Passat	900.00	12,500.00	nan	1,100.00	8,594.59
	Touareg	nan	nan	nan	nan	nan

- save both dataframes fines and owners to csv files without index

Chapter IX

Exercice 05 : Pandas optimizations

	Exercise 05
	Pandas optimizations
Turn-in directory : *ex05/*	
Files to turn in : `optimizations.ipynb`	
Allowed functions : `import pandas as pd, import gc`	

We are coming back to the idea of code efficiency. By now you know the basics of the Pandas. It is time to know some cool stuff that most of the Pandas users do not use and do not know about.

- read the fines.csv that you saved it the previous exercise

- iterations: in all the following subtasks you need to calculate fines/refund*year for each row and create a new column with the calculated data and measure the time using the magic command %%timeit in the cell

  - loop: write a function that iterates through the dataframe using for i in range(0, len(df)), iloc and append() to a list, assign the result of the function to a new column in the dataframe

  - do it using iterrows()

  - do it using apply() and lambda function

  - do it using Series objects from the dataframe

  - do it as in the previous subtask but with the methods .values

- indexing: measure the time using the magic command %%timeit in the cell

  - get a row for a specific CarNumber, for example, 'O136HO197RUS'

  - set the index in your dataframe with CarNumber

  - again, get a row for the same CarNumber

- downcasting:
  - run df.info(memory_usage='deep'), pay attention to the Dtype and the memory usage
  - make a copy() of your initial dataframe into another dataframe optimized
  - downcast from float64 to float32 for all the columns
  - downcast from int64 to the smallest numerical dtype possible
  - run info(memory_usage='deep') for your new dataframe, pay attention to the Dtype and the memory usage
- categories:
  - change the object type columns to the type category
  - check the memory usage this time, it probably has a decrease 2-3 times comparing to the initial dataframe
- memory clean
  - using %reset_selective and the library gc clean the memory only of your initial dataframe