2.0 - Pandas

Pandas is a library built on top of numpy, inheriting all of its features and adding much more. Pandas is packed with tools which sole purpose is to simplify data migration, validation and transformation.

Pynative has an excellent documentation guide explaining several key features in pandas.

2.1 - Pynative documentation & exercises

For the first set of exercises you will need to read the following document (with some exceptions, see 1.1.1 sections to read) → https://pynative.com/python-pandas-dataframe/

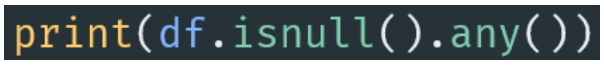
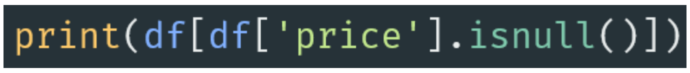
2.1.1 - Sections to read

A. Read all the sections until the headline “Insert columns”
B. Skip sections “Insert columns” through “DataFrame join”
C. Enter link https://pynative.com/python-pandas-exercise/
D. Download the automotive dataset, or get it from canvas
E. Do the exercises in the link from C.
F. For hints and additional information, see below (1.2. - Pynative hints & info)
G. For alternative solutions see canvas “Pynative alternative solutions”

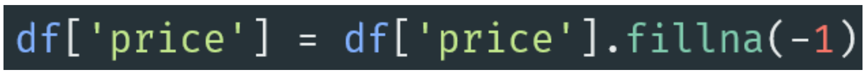
2.2 - Pynative hints & info

Exercises, referenced numerical relative to the document.

1. Sometimes an entity model is not enough to properly understand the data contained in a dataset. By utilizing the functions **head**, **tail**, and **info** one can get a concrete overview of the dataset.

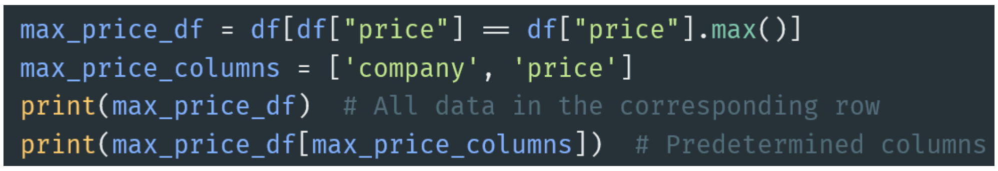
2. The solution that pynative suggests is a more brute-force approach since it simply replaces the value without any regards to what the implications of the missing values might be. A more analytical approach would be to first analyze if there are any missing values.
   
   ```
   print(df.isnull().any())
   ```
   Using this method we can observe that only the price column is missing values. Those of which we can easily print by printing all the null value rows in the column prices.
   
   ```
   print(df[df['price'].isnull()])
   ```
   a. Preferably at this point (in ETL) these rows should be flagged as not meeting specification requirements and stored in a seperate table

b. However, in this situation it suffices to mark them as “-1” since a cost can’t be negative.



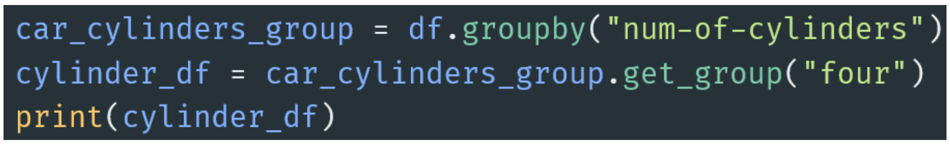
```python
df['price'] = df['price'].fillna(-1)
```

3. The solution they propose is based on extracting all rows in which the price corresponds to the max price found in the price column. Whilst their solution is a once-liner, it may be hard to decrypt, hence the piece below is exactly the same but split into multiple lines for readability.



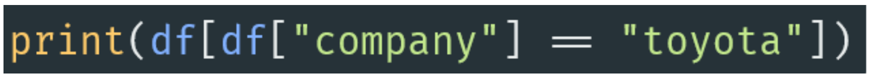
```python
max_price_df = df[df["price"] == df["price"].max()]
max_price_columns = ['company', 'price']
print(max_price_df)  # All data in the corresponding row
print(max_price_df[max_price_columns])  # Predetermined columns
```

4. Pynative proposes a straightforward method and resembles the SQL solution. E.g. If you wish to extract based on the number of cylinders in the car then we can use the same method.



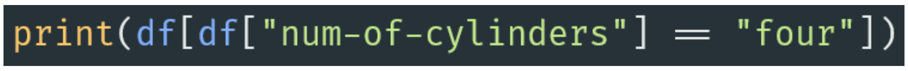
```python
car_cylinders_group = df.groupby("num-of-cylinders")
cylinder_df = car_cylinders_group.get_group("four")
print(cylinder_df)
```

However, there are shorthand commands to do the same operation.
Toyota company, shorthand



```python
print(df[df["company"] == "toyota"])
```

Car cylinders, shorthand



```python
print(df[df["num-of-cylinders"] == "four"])
```

5. Not much to mention here. I advice you to think about how you could do a value count for specific manufactures.

6. Nothing to add. The groupby makes it very easy to target specific groups of columns. First they group the rows by company and then extracts the max price of each company. You could swap max to average and get the company's average car selling values.
You could also install the matplotlib for python (data visualization) and present the max prices in a bar graph.

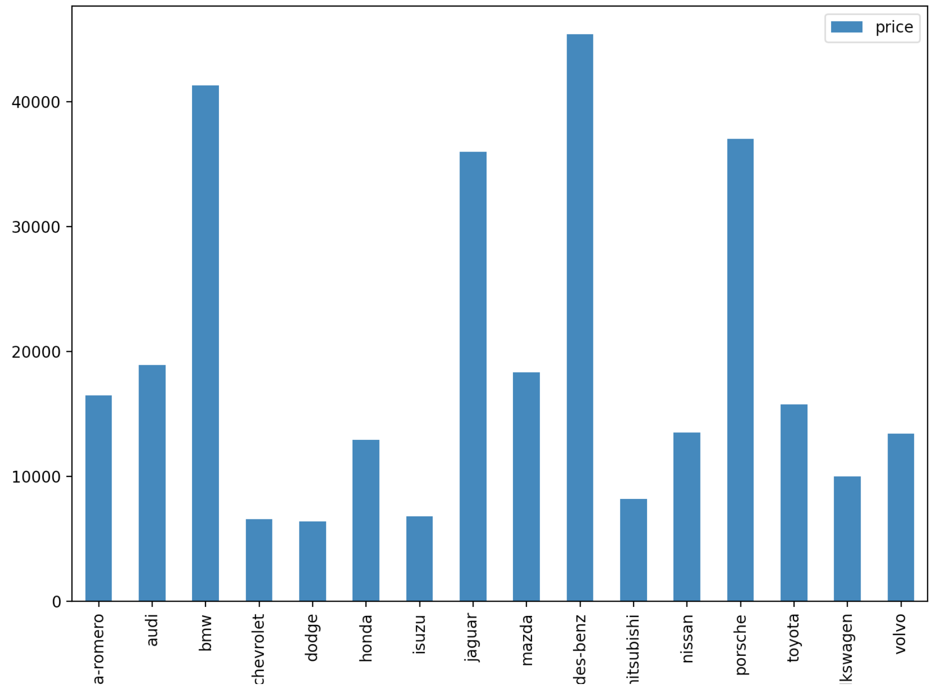
7. Code to create a bar plot

```
import matplotlib.pyplot as plt

# read from file ...
company_group = df.groupby("company")
max_price_df = company_group["company", "price"].max()

max_price_df.plot(kind="bar", x="company", y="price")
plt.show()
```

Output



8. The method is extremely similar to the previous solution, but with different columns.

9. For this exercise I would refer to pandas official documentation
   https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sort_values.html

10. Refering to the official documentation once more
    https://pandas.pydata.org/docs/reference/api/pandas.concat.html

11. There are two methods of merging. Ordered, or not ordered
    https://pandas.pydata.org/docs/reference/api/pandas.merge.html
    https://pandas.pydata.org/docs/reference/api/pandas.merge_ordered.html
    What is the difference between concat and merge in pandas?
    When is merging preferred over concat? Can you create another example from the exercise?

For a more extensive tutorial reading: https://pynative.com/python/pandas/

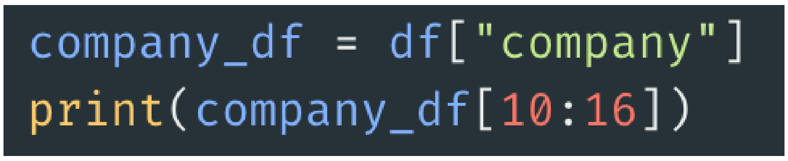
2.3 - Validating and transforming

Oh no, someone let their kid play next to the computer with the automobile csv open!

A. Download the file ruined_automobile_data.csv from canvas.

B. Inspect the damage by using the same method as in 2.2B

C. Inspect the rows surrounding the damaged lines using python.
   a. Either do it by fetching each individual index, e.g.



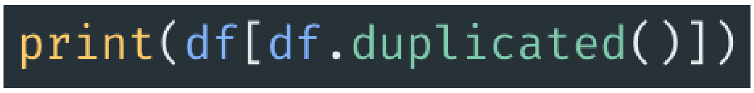
```python
company_df = df["company"]
print(company_df[10:16])
```

   b. Look in the actual file (cheating)
   c. Or create a loop that prints the rows surrounding the damaged lines

D. Notice any pattern? hint: you can use the pandas method *fillna* to replace the missing values. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html

E. Perform the same operation for all the damaged columns that you observed in 2.3B

F. Figure out if there are any duplicates and remove them using https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.duplicated.html
To "See" the duplicates you can use this piece of code, however, you need the documentation to "deal" (remove) the duplicates



```python
print(df[df.duplicated()])
```

G. Finally, verify that some values are according to the data requirements

   Manufactures - alfa-romero, audi, bmw, chevrolet, dodge, honda, isuzu, jaguar, mazda, mercedes-benz, mitsubishi, nissan, porsche, toyota, volkswagen, volvo

   Styles - convertible, hatchback, sedan, wagon, hardtop

   Hint: the function isin can be helpful to find out if selected rows contains certain values
   https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.isin.html
   https://stackoverflow.com/questions/19960077/how-to-filter-pandas-dataframe-using-in-and-not-in-like-in-sql