CoGrammar

Datasets and DataFrames

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Data Science Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: Open Class Questions

Data Science Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support

- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting

- We would love your **feedback** on lectures: Feedback on Lectures

Lecture Objectives

- Learn how to read and manipulate data with the power of Pandas

Working With Datasets

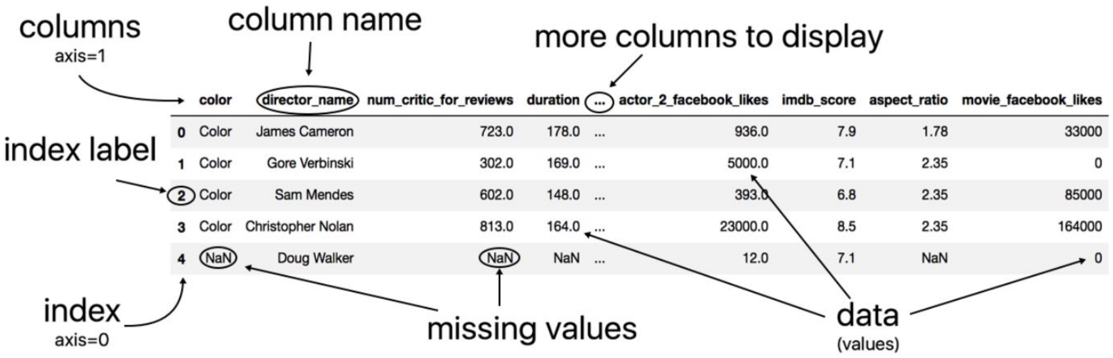
- ★ For context, when a dataset is mentioned, it is a collection of related data.
- ★ With datasets we can manipulate the data in a multitude of ways programmatically.
- ★ With the help of pandas DataFrames, we can effortlessly manipulate data to suit our needs.

Jupyter Notebook

- ★ From now on, we will be using Jupyter Notebook. Which can be described as follows from its official website (jupyter.org):
  - ○ “The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualisations and narrative text.”
  - ○ “Uses include data cleaning and transformation, numerical simulation, statistical modelling, data visualisation, machine learning and much more.”

Pandas DataFrame

★ **The pandas' library documentation defines a DataFrame as a "two-dimensional, size-mutable, with labelled rows and columns."**



```
   color  director_name      num_critic_for_reviews  duration  ...  actor_2_facebook_likes  imdb_score  aspect_ratio  movie_facebook_likes
0  Color  James Cameron      723.0                   178.0     ...  936.0                   7.9         1.78          33000
1  Color  Gore Verbinski     302.0                   169.0     ...  5000.0                  7.1         2.35          0
2  Color  Sam Mendes         602.0                   148.0     ...  393.0                   6.8         2.35          85000
3  Color  Christopher Nolan  813.0                   164.0     ...  23000.0                 8.5         2.35          164000
4  NaN    Doug Walker        NaN                     NaN       ...  12.0                    7.1         NaN           0
```

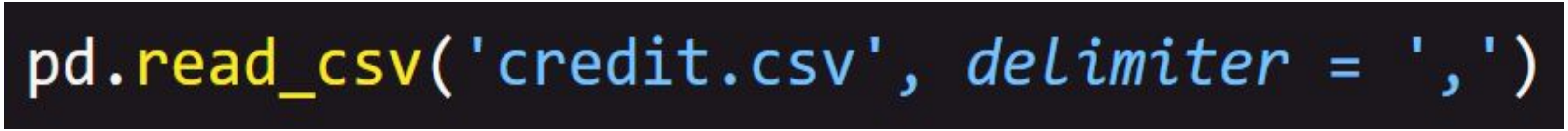
Anatomy of a DataFrame

Pandas DataFrame

- To simplify, think of a DataFrame as a table of data with the following characteristics (Lynn 2018) :
  - “There can be multiple rows and columns in the data
  - Each row represents a sample of data,
  - Each column contains a different variable that describes the samples (rows).
  - The data in every column is usually the same type of data - eg. numbers, strings, dates.
  - Usually, unlike an excel dataset, DataFrames avoid having missing values.

Working With Datasets

- You can read data from a .csv (comma separated values) file into a DataFrame using the read_csv() function.



```
pd.read_csv('credit.csv', delimiter = ',')
```

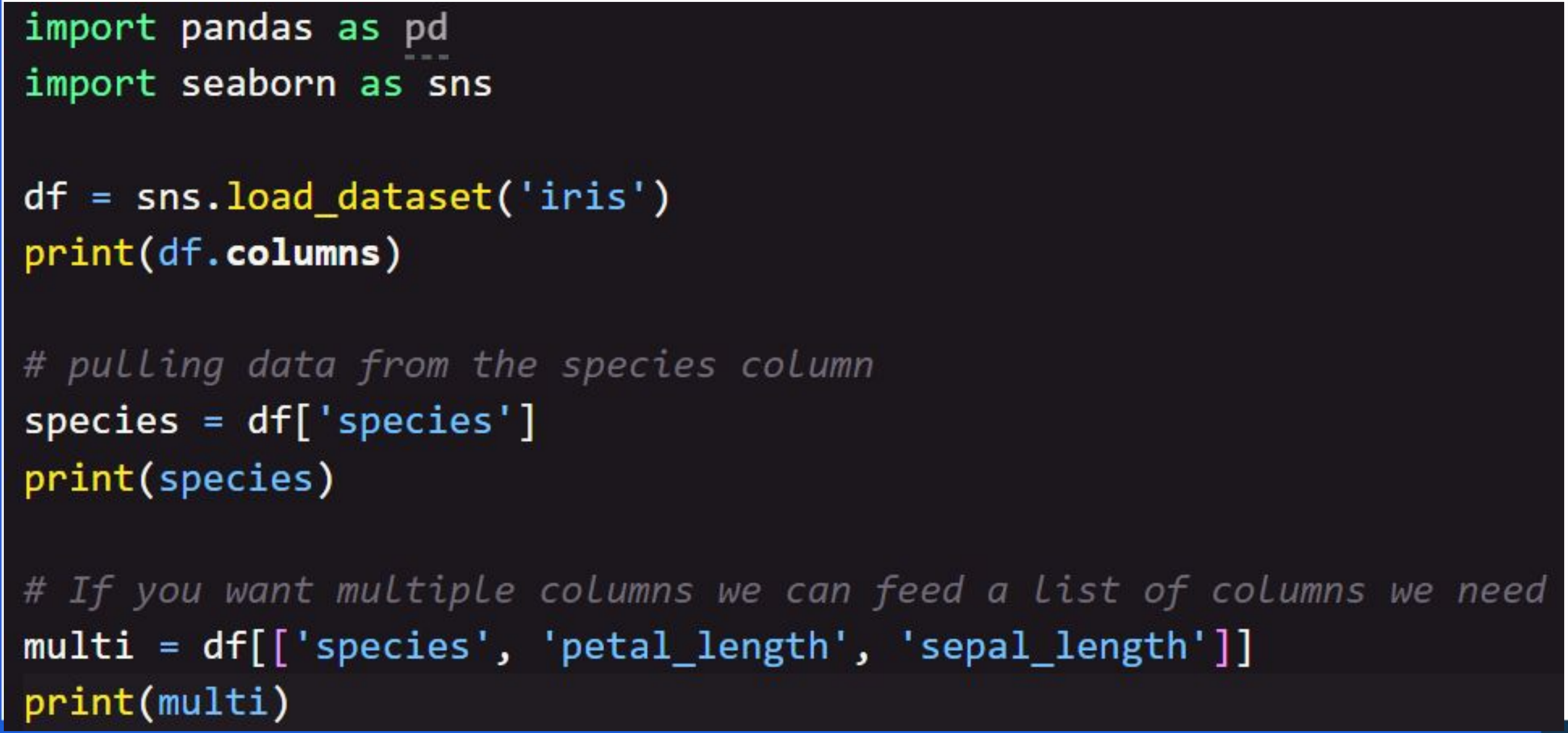
- There are other functions that can be used to read data from other sources into DataFrames, such as : read_excel(), read_sql() to name a few.

Selecting Columns in Pandas

★ There are a multitude of ways to specify columns in Pandas. A simplified way would be to use dictionary notation for specific columns.

★ We could think of DataFrames can be thought of as dictionaries : keys would be the column name and values would be the values within the column.

Example



```python
import pandas as pd
import seaborn as sns

df = sns.load_dataset('iris')
print(df.columns)

# pulling data from the species column
species = df['species']
print(species)

# If you want multiple columns we can feed a list of columns we need
multi = df[['species', 'petal_length', 'sepal_length']]
print(multi)
```

CoGrammar

Built-in DataFrame Methods

★ Here is a list of common built-in functions in Pandas for such things :

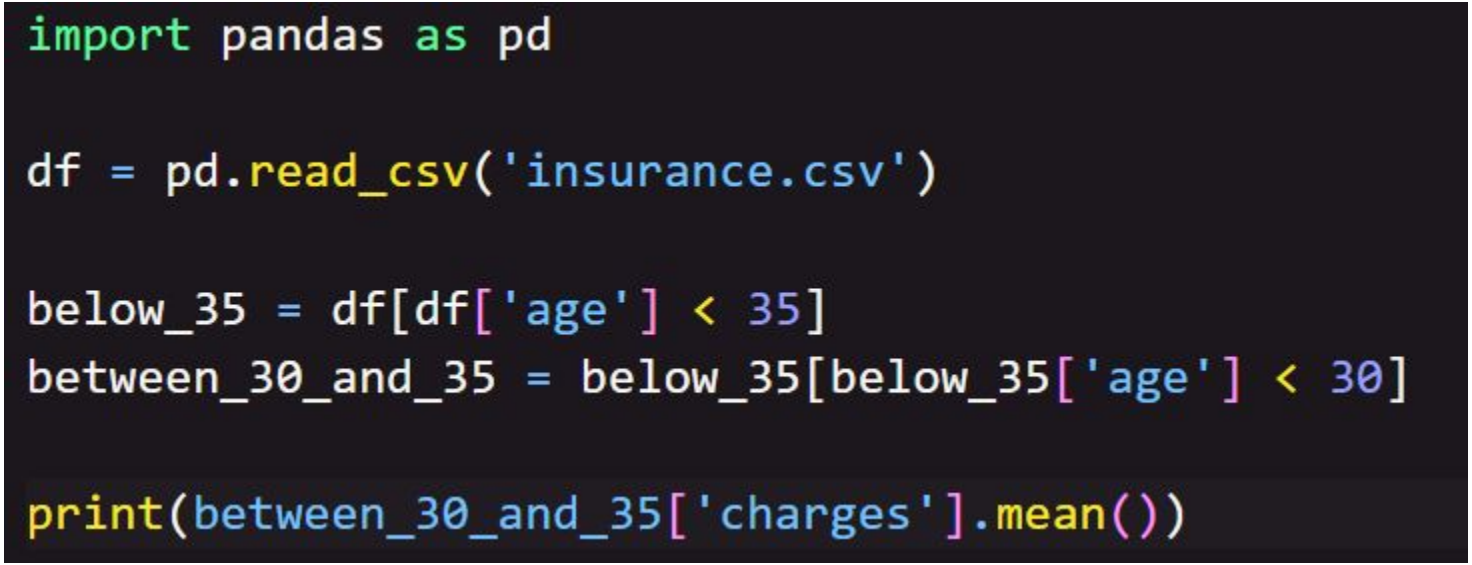
- ○ mean() - Computes the mean for each column.
- ○ min() - Computes the minimum for each column.
- ○ max() - Computes the maximum for each column.
- ○ std() - Computes the standard deviation for each column.
- ○ var() - Computes the variance for each column.
- ○ nunique() - Computes the number of unique values in each column.

Grouping in Pandas

★ Data analysis can get a bit complicated at times, and some more advanced functionality might be needed.

★ For instance, we need the average the insurance charges for all individuals between the age of 30 to 35.

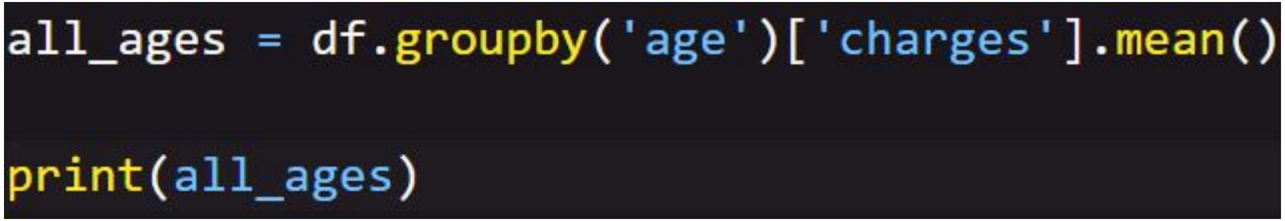
Example



```
import pandas as pd

df = pd.read_csv('insurance.csv')

below_35 = df[df['age'] < 35]
between_30_and_35 = below_35[below_35['age'] < 30]

print(between_30_and_35['charges'].mean())
```

Working With Datasets

- While the previous example works nicely, what if we wanted to average the charges for every age group?
- It can be done with the same syntax, but will take quite a few lines of code.
- Luckily, pandas has something for us to make this possible using the least lines of code possible :



```
all_ages = df.groupby('age')['charges'].mean()

print(all_ages)
```

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

CoGrammar

Thank you for joining!