Twitter Thread by Sunrit Jana ■



Sunrit Jana ■
@JanaSunrise



Pandas is an amazing data analysis and manipulation library for python, Really popular when working with ML, Data science or more.

It has a robust data structure, Dataframe for manipulation and analyzing data.

Here's some tips to help you work better with pandas. Let's go! ↓

If you're not aware about what a Dataframe is, It's an optimized data structure for loading data, analysing it, manipulating data in it, and Mostly gathering insights.

It uses Cython backend which transpiles into C for optimized code.

Here's how a dataframe looks like,

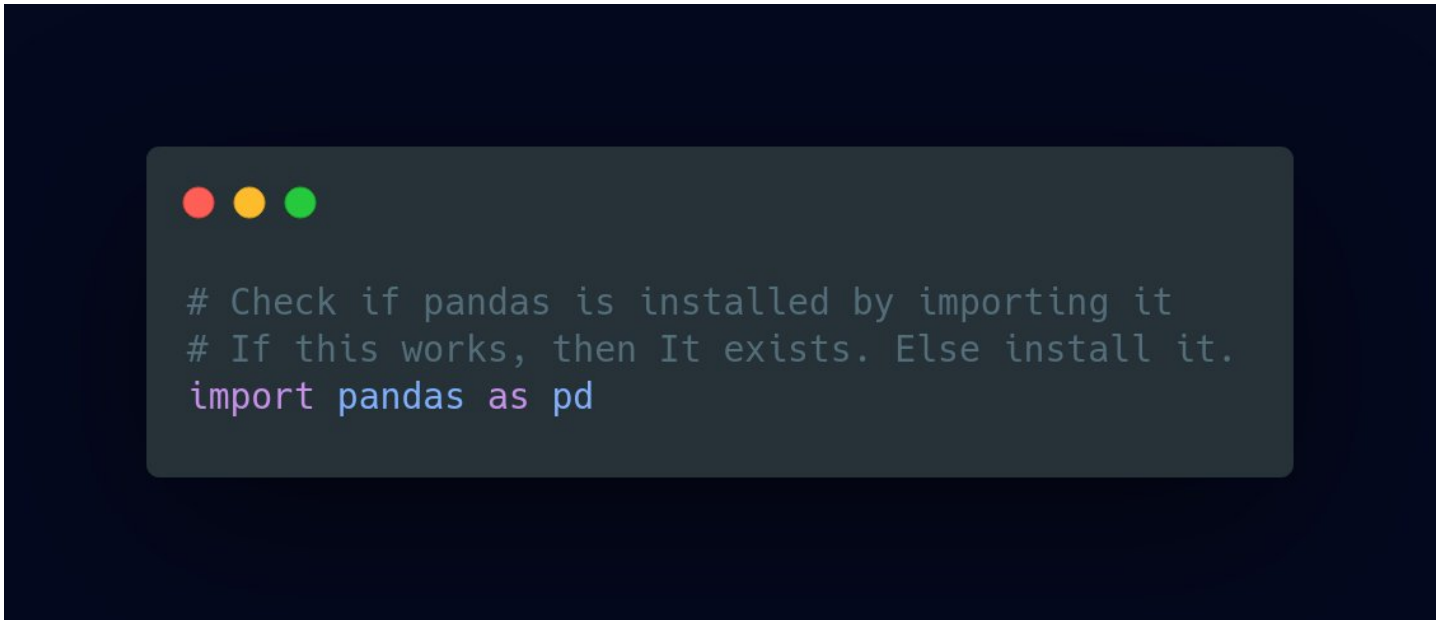
```
# Get the head of the Train Dataframe
train_df.head()
```

id	cat0	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	...	cont5	cont6	cont7	cont8	cont9	cont10	cont11	cont12	cont13	target
1	B	B	B	C	B	B	A	E	C	N	...	0.400361	0.160266	0.310921	0.389470	0.267559	0.237281	0.377873	0.322401	0.869850	8.113634
2	B	B	A	A	B	D	A	F	A	O	...	0.533087	0.558922	0.516294	0.594928	0.341439	0.906013	0.921701	0.261975	0.465083	8.481233
3	A	A	A	C	B	D	A	D	A	F	...	0.650609	0.375348	0.902567	0.555205	0.843531	0.748809	0.620126	0.541474	0.763846	8.364351
4	B	B	A	C	B	D	A	E	C	K	...	0.668980	0.239061	0.732948	0.679618	0.574844	0.346010	0.714610	0.540150	0.280682	8.049253
6	A	A	A	C	B	D	A	E	A	N	...	0.686964	0.420667	0.648182	0.684501	0.956692	1.000773	0.776742	0.625849	0.250823	7.972260

5 rows × 25 columns

Before we start, You need to ensure, you have pandas installed. If you don't, Do that before moving ahead!

Here are the tips, Let's go!



1/ Convert PD series to Dataframe

We all have struggled, when we deal with pandas series. It's always easier to work with Dataframes, rather than series. Here is how you can convert series to dataframe easily.



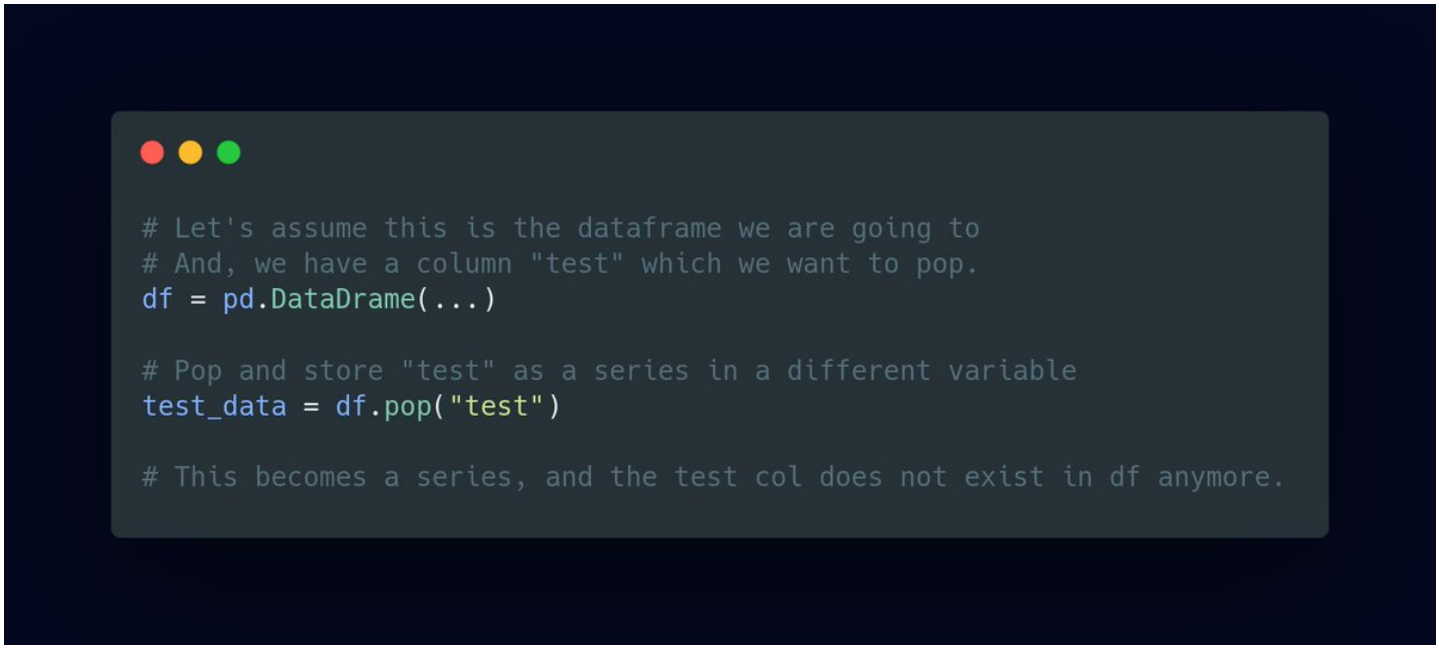
2/ How to create dummy Dataframe for testing

We always need dataframes for testing and analysing normally, if we do not have data ready. Here is how you can use Pandas API to generate different types of data.



3/ Remove a column from the current DF and Convert it as separate series

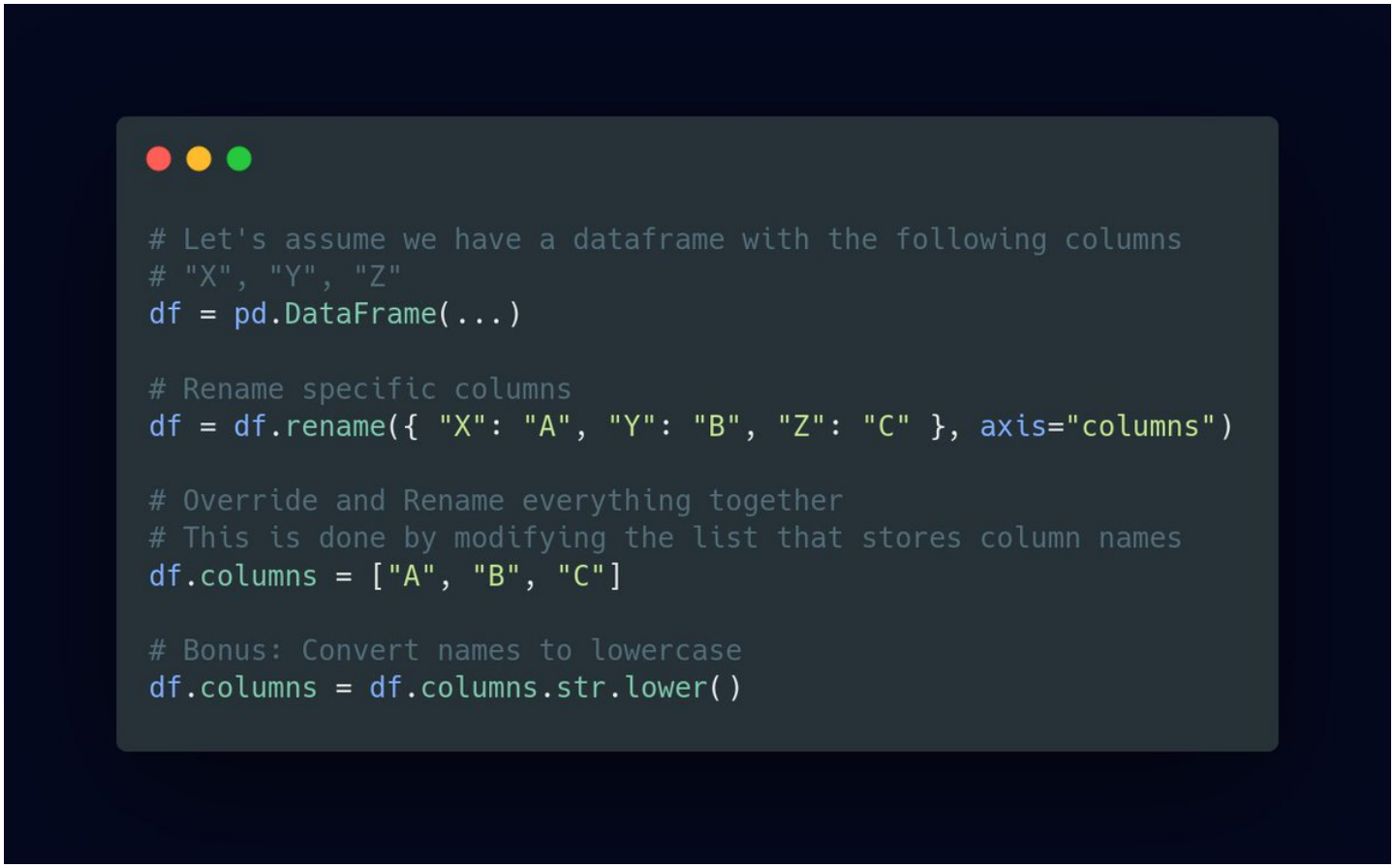
We sometimes need to remove a column from dataframe, and use it separately for other use. Here is how you can do that!



```python
# Let's assume this is the dataframe we are going to
# And, we have a column "test" which we want to pop.
df = pd.DataDrame(...)

# Pop and store "test" as a series in a different variable
test_data = df.pop("test")

# This becomes a series, and the test col does not exist in df anymore.
```

4/ Easily renaming columns

In order to rename specific columns, or Override everything, or just change things a bit easily, Here is how you can do renaming in 2 major ways, along with a tip to lowercase column names.



```python
# Let's assume we have a dataframe with the following columns
# "X", "Y", "Z"
df = pd.DataFrame(...)

# Rename specific columns
df = df.rename({ "X": "A", "Y": "B", "Z": "C" }, axis="columns")

# Override and Rename everything together
# This is done by modifying the list that stores column names
df.columns = ["A", "B", "C"]

# Bonus: Convert names to lowercase
df.columns = df.columns.str.lower()
```

5/ Add prefix / suffix to column names

Sometimes, We need to add specific naming patterns to the columns names in a dataframe, such as adding `_train` to end, or `test_` in front. Here's the easiest way to do that!



```
# Let's assume we have a dataframe "df"
df = pd.DataFrame(...)

# Add prefix of "test_" to columns
df.add_prefix("test_")

# Or, Add suffix of "_train"
df.add_suffix("_train")
```

6/ Reversing a Dataframe in different orders

Sometimes, We have the need to reverse a dataframe, either by columns, or rows. This is the perfect way to do it!



7/ Fixing data types and Making columns as needed

Sometimes columns are in the wrong data type which needs to be fixed, and We need to create a column from existing ones for several purpose.

An easy way to do it, is this,



```python
# Change the value of an column
# Assume we want to convert "id" to string
df["id"] = df["id"].astype("str")

# Convert multiple columns type
# For example, "id" to string and "phoneNumber" to int
df.astype({ "id": "str", "phoneNumber": "int" })

# Remove the extra spaces from "name" column
df["name"] = df["name"].astype("str").str.strip()

# Extract the Date from "date" column, and convert it into datetime
# Then assign it to a new column "datetime"
df["datetime"] = pd.to_datetime(df["date"])
```

8/ Selecting columns by data type

There are several instances, where we need to select column by data type. For example, processing and cleaning categorical and numerical variables separately. Here is how to select them differently, really easily!

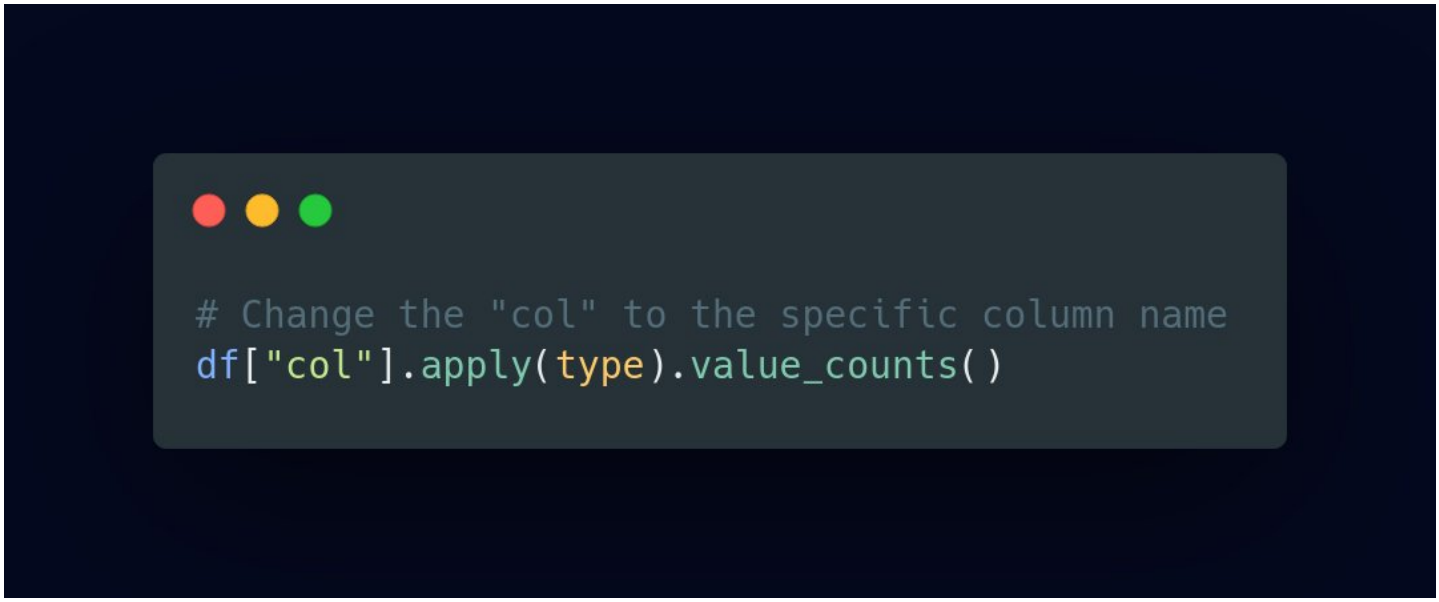


```python
# Select all the columns with datatype of number
df.select_dtypes(include="number")

# Select all columns with datatype of Number, Category and Object
df.select_dtypes(include=["number", "category", "object"])

# Select all columns which are *NOT* Datetime or timedelta
df.select_dtypes(exclude=["datetime", "timedelta"])
```

9/ Check the datatypes in a column, when annotated as "object"

When a column is given the type of "object", It may or may not have multiple types of data. Here is an easy way to check

that!



10/ Converting columns from Continuous to categorical

There are a lot of instances, where we need to organize continuous variable into categories. Such as age column can be organized by Giving categories.

0-18: Child
18-66: Adult
66-100: Seniors

Here's how to do it,



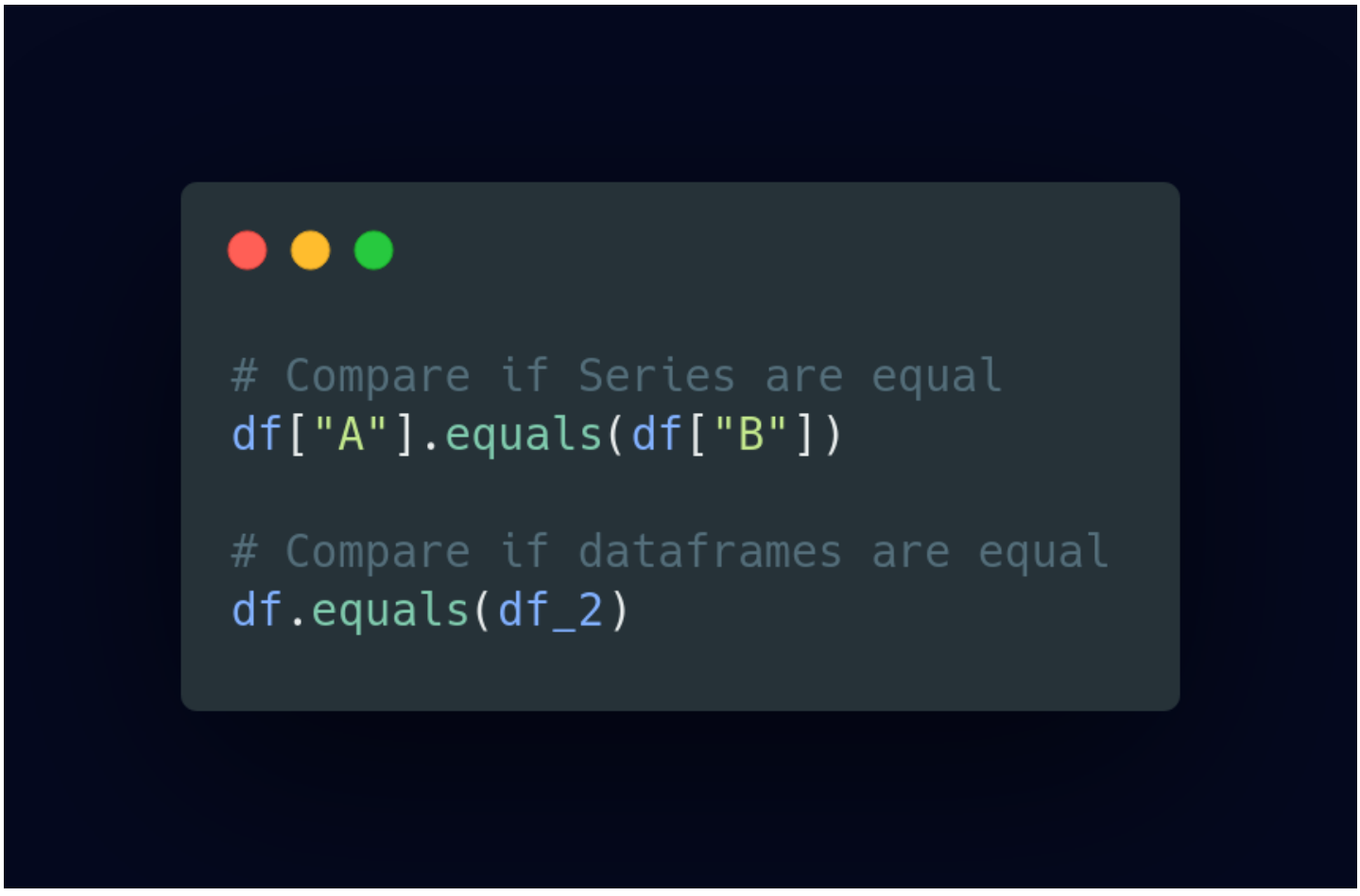
11/ Combine dates spread across multiple columns

Sometimes, When we have Date, Month and Year as different columns, It can get pretty hard, since it's Time-Series data. In order to combine them, Here is how that can be done.



12/ Comparing if 2 series / dataframes are same

There are a lot of times, When we need to compare between pandas Series, or Dataframes, and Check if they're same. Here is the right way to do it!



13/ Plotting using Pandas

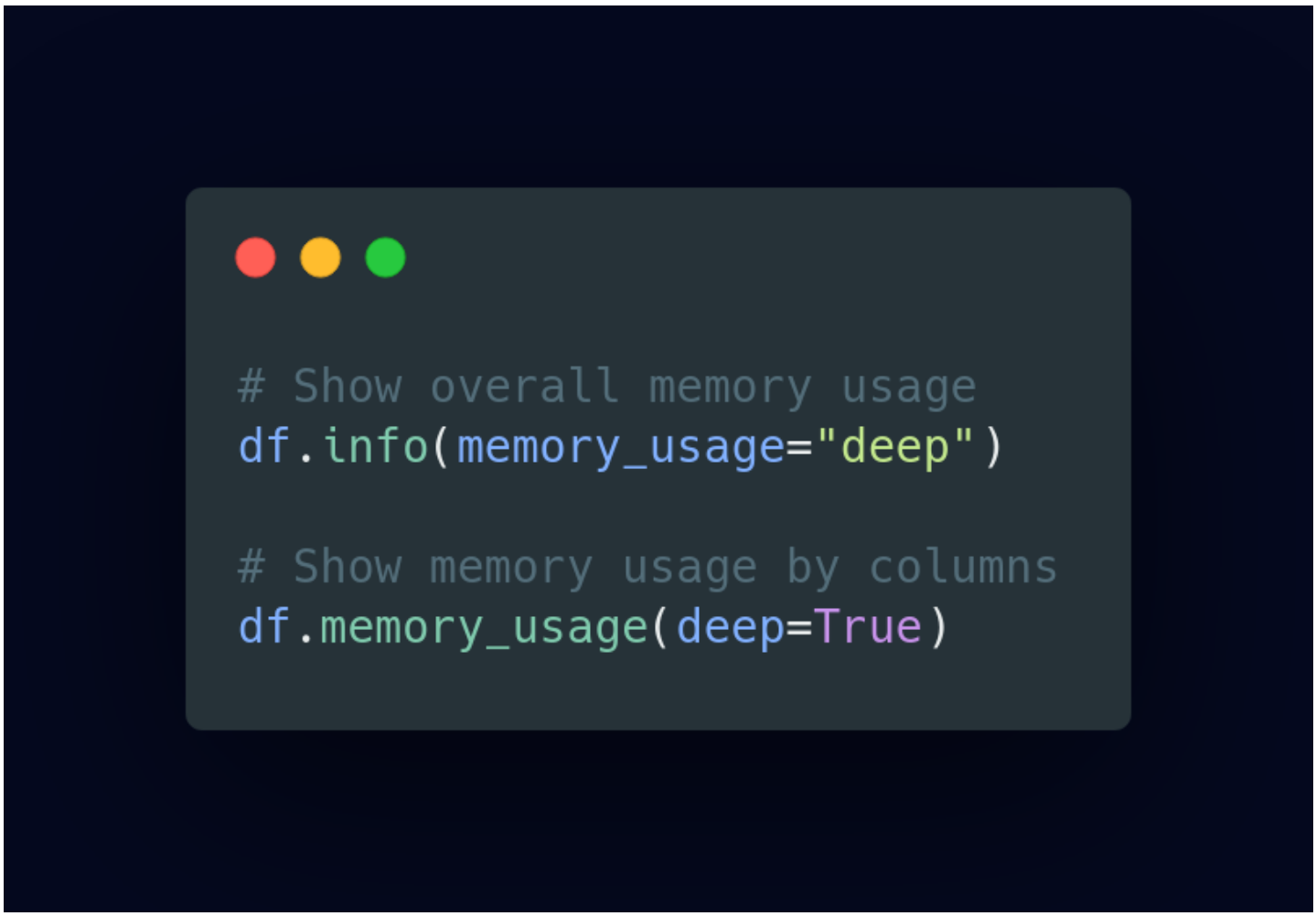
Pandas provides a REALLY EASY way to perform plotting and create graphs combined with matplotlib library. Matplotlib's a story, for another day.

Here is the link to official docs for tutorial on that: https://t.co/Embc3jwD8u

Let's see how to!

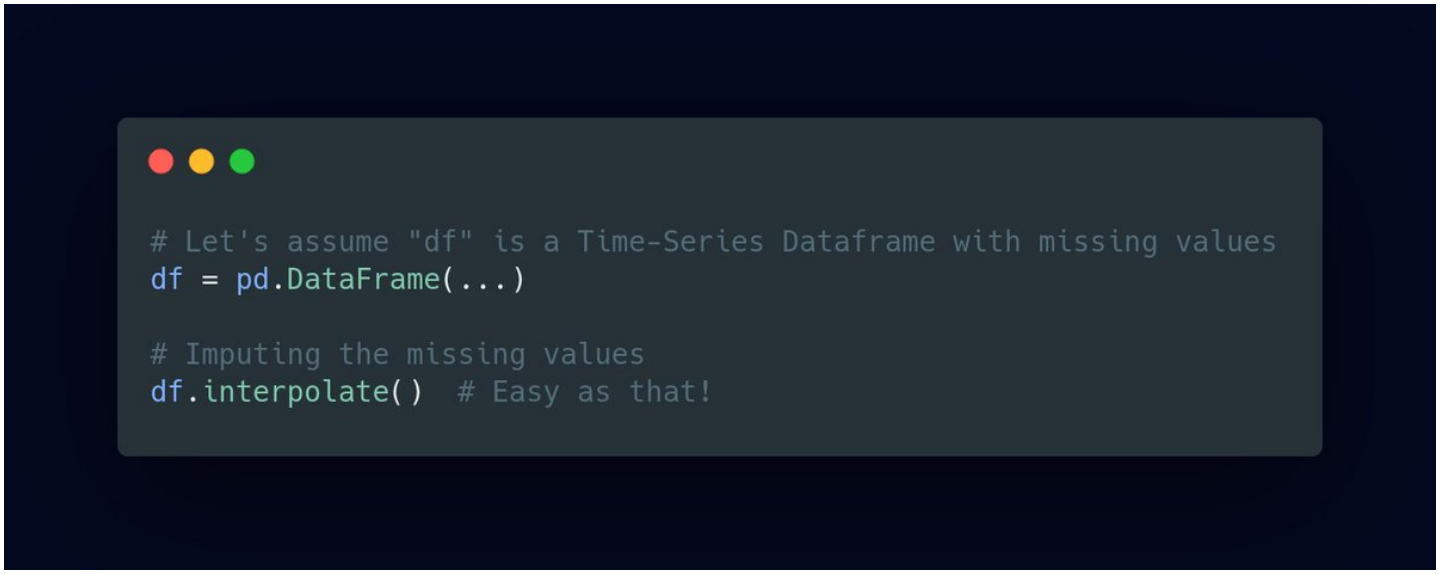
14/ Get memory usage of dataframe

A lot of instances, Large data when loaded, take up a lot of memory. Here is an easy and quick way to analyse the memory usage.



15/ Impute missing values in Time-Series data

Real world, does have a lot of missing data. Time-Series is a one type of data, which has that too! Pandas makes it really easy to impute and fill them. Here's how to



16/ Shuffle rows in Pandas

Shuffling rows is really useful, especially when randomizing the dataset before Splitting and feeding into model to ensure randomness. Here's the easiest way to do.



17/ Split the dataset into 2 Random sets

When working with ML or Data science, Usually we need to split data into 2 sets, Training and Test. Sometimes even validation.
Here is how you can do that using Pandas.

PS: Scikit learn can do this too using train_test_split



18/ Ignoring warnings easily [PYTHON]

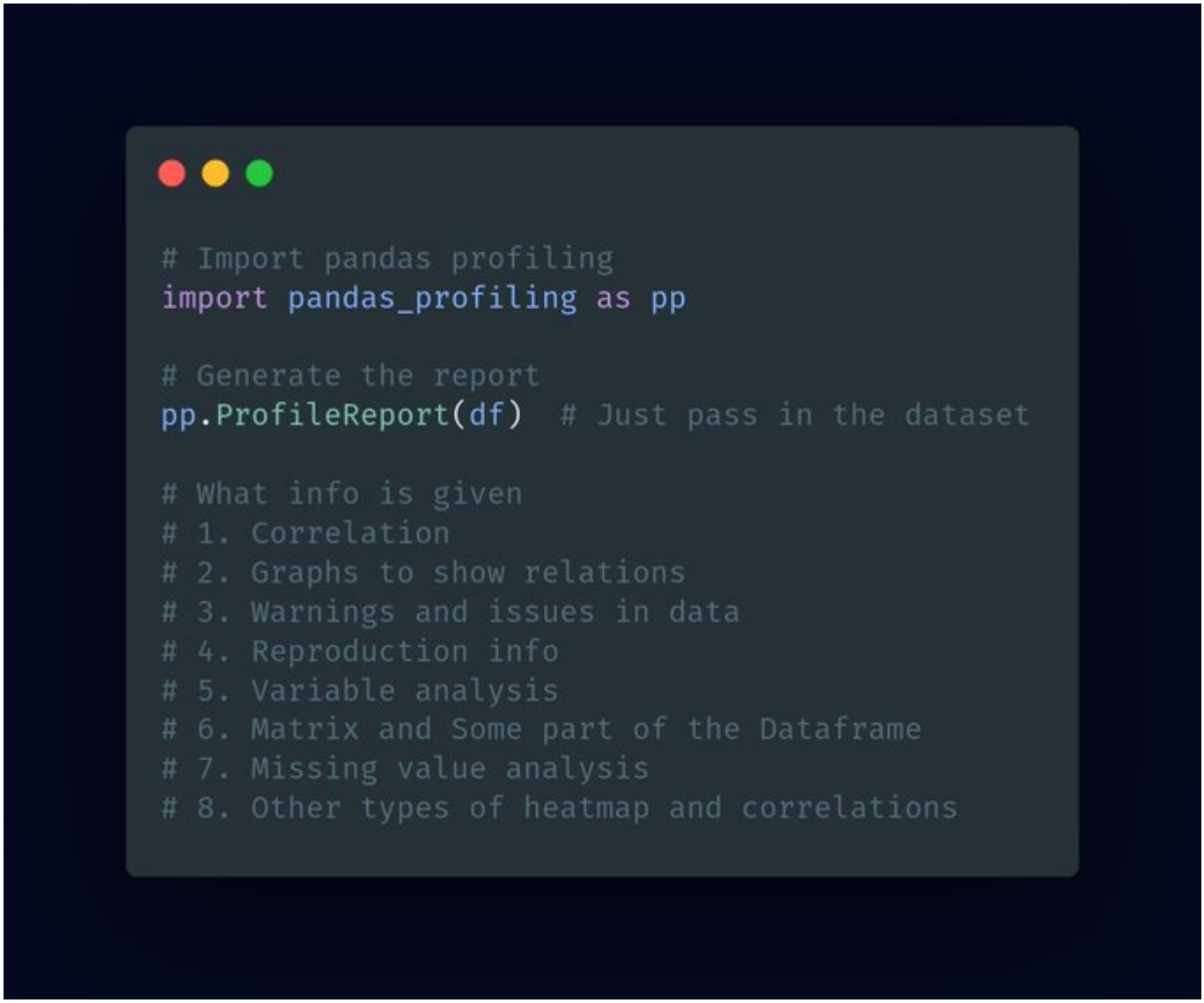
Warnings are usually redundant, except some cases. So, Here is how you can directly ignore them from Python.



BONUS, Auto EDA!

There are some libraries which can perform auto EDA, gather insights automatically from the dataset, and Display it! This is effective when you have done a lot of EDA, Analyze rest of it easily, and quickly.

We'll be using Pandas profiling. Here's how to do it.



You have reached the end! ■

Really enjoyed sharing these useful functions in Pandas with you! More amazing content, Coming soon, next time!

Let me know, what you think! And If you're feeling generous, Do retweet to share with everyone else!

It's time for me to leave.
Don't forget to follow me at @JanaSunrise, I keep making such amazing content!

If you don't know me, I'm Sunrit & I make amazing content on ML, Python, Startups & More! ■

Stay safe and take care! Thank you for everything, See you next time ❤■ Bye!