Published in Towards AI

Amit Chauhan Follow

Jun 21, 2021 · 6 min read · ✦ · Listen

Save

Reading CSV(), Excel(), JSON () and HTML() File Formats in Pandas

Panda reads data from csv, txt, excel & more file formats

Photo by Blake Connally on Unsplash

What is Pandas?

Pandas is a Python library containing a bunch of capacities and specific information structures that have been intended to help Python developers to perform information examination errands in an organized manner.

Importing data is the most fundamental and absolute initial phase in any information-related work. The capacity to import the information accurately is a must have skill for every data scientist.

Data exists in many different forms, and not only should we know how to import various data formats but also how to analyze and manipulate the data to infer insights.

The majority of the things that pandas should do can be possible with fundamental Python, yet the gathered arrangement of pandas capacities and information structure makes the information examination assignments more reliable as far as punctuation and in this manner helps readability.

Specific highlights of pandas that we will be taking a look at over this and the few scenes include:

- Reading information stored in CSV documents
- Slicing and subsetting information in Dataframes (tables!)
- Dealing with missing information
- Reshaping information (long → wide, wide → long)
- Inserting and deleting columns from data structures
- Joining of datasets (after they have been stacked into Dataframes)

If you are asking why I compose pandas with a lower case 'p' because it is the name of the bundle and Python is case sensitive.

Let's now look at how panda reads data from csv, txt, excel & more file formats:

1. Load CSV files

CSV (comma-separated value) file is a common file format for transferring and storing data.

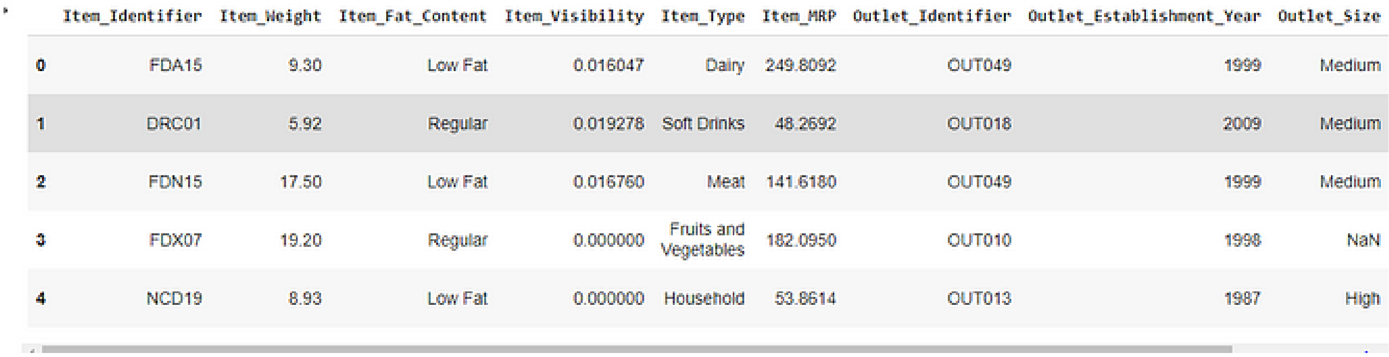
The capacity to read, write and manipulate data to and from CSV documents utilizing Python is vital expertise to dominate for any data scientist or business analysis

The essential interaction of loading data from a CSV document into a Pandas DataFrame (with all working out in a good way) is accomplished utilizing the "read_csv" work in Pandas.

```
# Loading the Pandas library with the alias as 'pd'
import pandas as pd

# Read data from file 'test.csv'
data = pd.read_csv("Test.csv")

# Check the first 5 lines of the loaded data
data.head(5)
```



	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High

A photo by Author

As should be obvious, read_csv takes the first row as the names for the columns. It is feasible to give different names to the columns. For this reason, we need to skip the first line by setting the parameter “header” to 0 and we need to assign a list of columns with the column names

On the plus side:

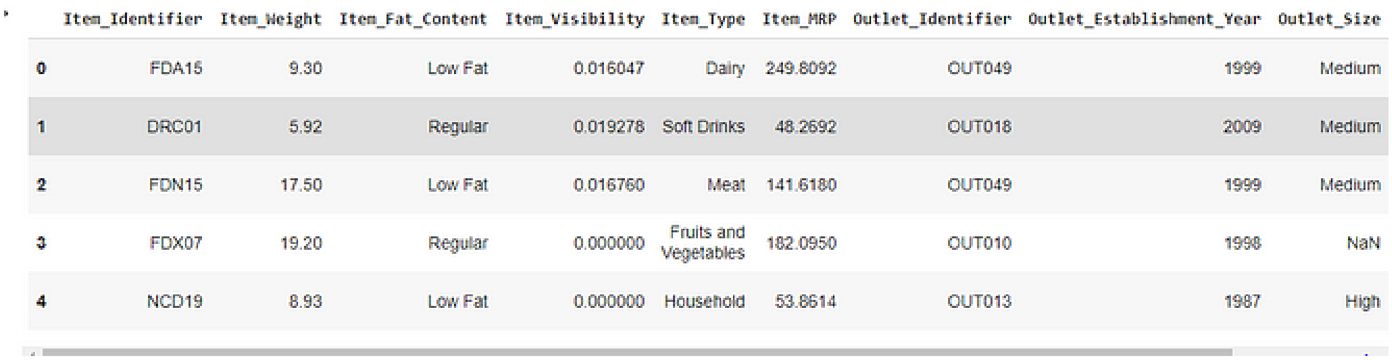
- CSV design is widespread and the information can be stacked by practically any software.
- CSV records are easy to understand and troubleshoot with a fundamental text editor
- CSV records are quick to create and load into memory before analysis.

2. Reading Excel Files

To read the excel file, we need to use *read_excel*.

```
# Read data from file 'filename.csv'
train2 = pd.read_excel("Train_BigMart.xlsx")

# Check the first 5 lines of the loaded data
train2.head(5)
```



	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High

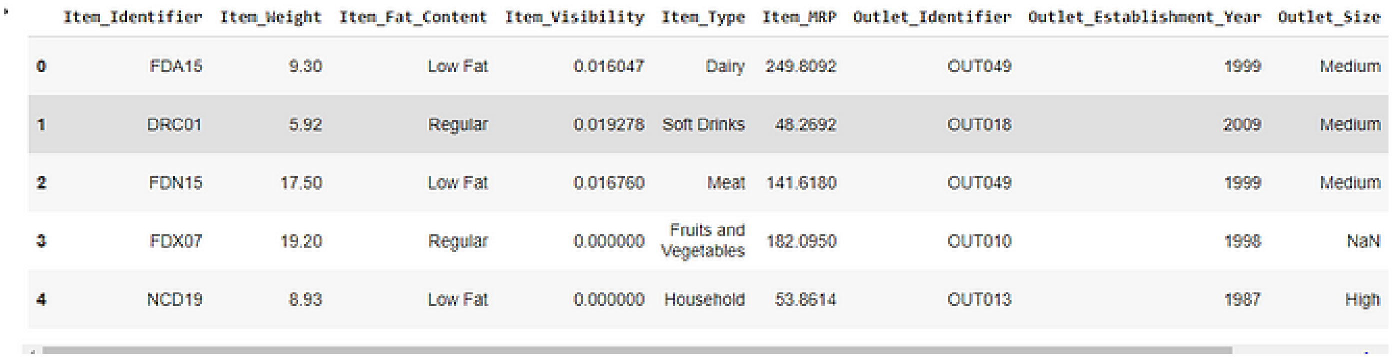
A photo by Author

If the document “BigMart.xlsx” contains two sheets, we can read that using the same read_excel. A complete Excel document, which can consist of many sheets, can be read like this:

```
#Reading Multiple sheets of excel
Excel = pd.ExcelFile("Train_Test_BigMart.xlsx")

# Creating two different data frames for the Excel files
df1 = pd.read_excel(Excel,"Train_BigMart")
df2 = pd.read_excel(Excel,"Test_BigMart")

# Preview the first 5 lines of the loaded data (sheet 1 - df1)
df1.head(5)
```

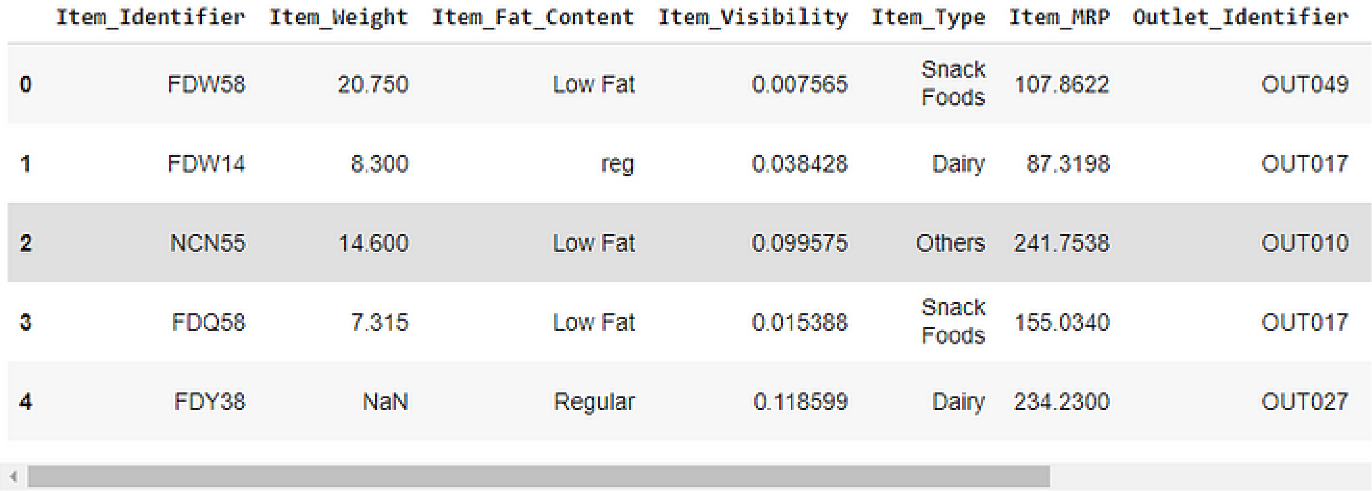


	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High

A photo by Author

```
# Check the first 5 lines of the loaded data (sheet 2 - df2)

df2.head(5)
```



	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT017
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT027

A photo by Author

The entire Excel file is loaded during the ExcelFile() call. This merely saves us from having to read the same file each time we want to access a new sheet.

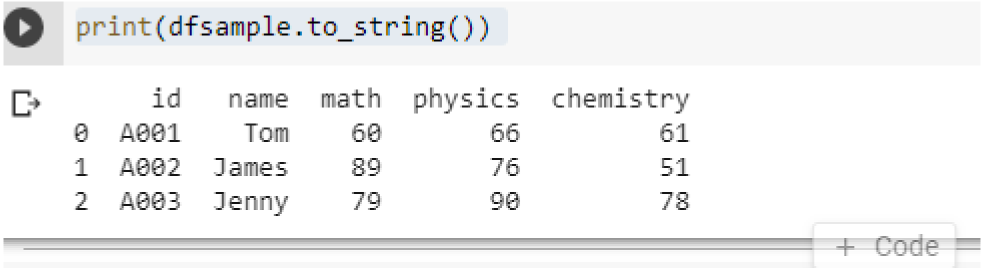
3. Reading JSON

A JSON record is a document that stores simple data structures and objects in JavaScript Object Notation (JSON) design. it is a standard information exchange design. It is used for communicating information between a web application and a worker. JSON documents are lightweight, text-based, user-friendly, and can be altered utilizing a text editor.

To read a JSON file via Pandas, we can use the read_json() method.

```
dfsample = pd.read_json("sample.json")

# use to_string() to print the whole DataFrame.
print(dfsample.to_string())
```

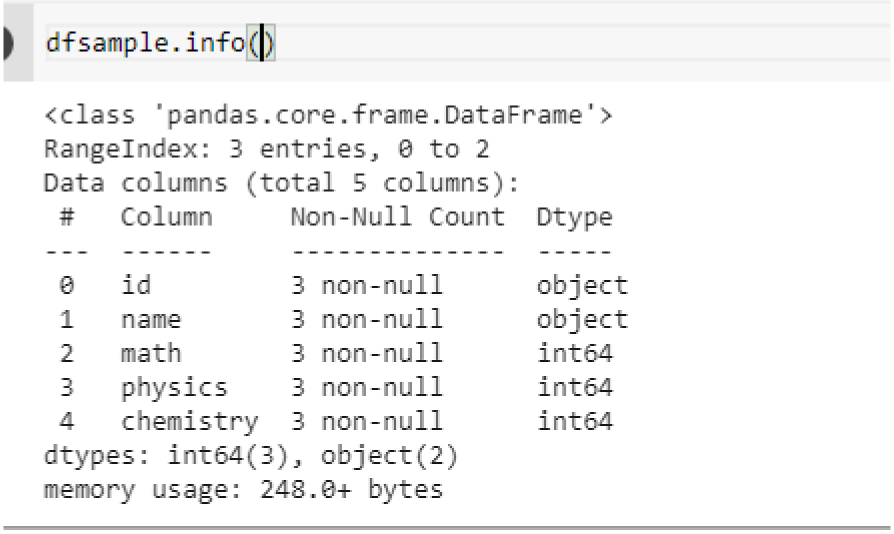


```
print(dfsample.to_string())
```

```
     id   name  math  physics  chemistry
0  A001    Tom    60       66         61
1  A002  James    89       76         51
2  A003  Jenny    79       90         78
```

A photo by Author

The result looks cool. Let’s take a look at the data types using df.info(). By default, numerical columns are assigned to numeric types, for example, the math, physics, and chemistry columns have been assigned int64.



```
dfsample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   id         3 non-null      object
 1   name       3 non-null      object
 2   math       3 non-null      int64 
 3   physics    3 non-null      int64 
 4   chemistry  3 non-null      int64 
dtypes: int64(3), object(2)
memory usage: 248.0+ bytes
```

A photo by Author

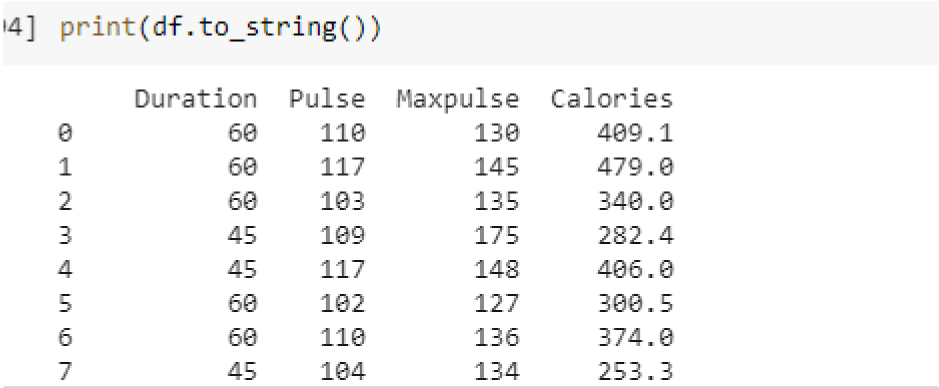
Load JSON from URL

To load JSON from an URL (API), use this code:

```
# Insert the URL you want to get the data from

URL = '<https://www.w3schools.com/python/pandas/data.js>'
df = pd.read_json(URL)

# use to_string() to print the dataFrame
print(df.to_string())
```



```
4] print(df.to_string())
```

```
    Duration  Pulse  Maxpulse  Calories
0         60    110       130     409.1
1         60    117       145     479.0
2         60    103       135     340.0
3         45    109       175     282.4
4         45    117       148     406.0
5         60    102       127     300.5
6         60    110       136     374.0
7         45    104       134     253.3
```

A photo by Author

Same as reading from a local file, it returns a DataFrame, and numerical columns are assigned numeric types by default.

Loading HTML Data

HTML is a Hypertext Markup Language that is majorly used for created web

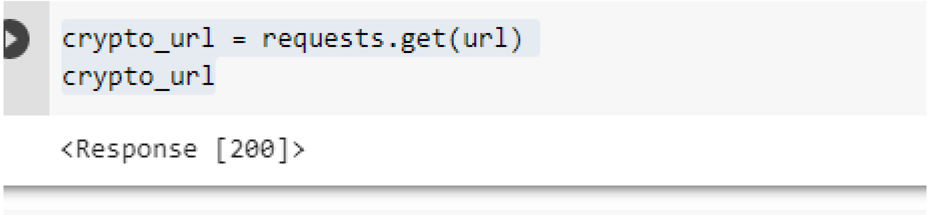
`pandas` uses `read_html()` to read the HTML document.

So, whenever we pass an HTML to pandas and expect it to output a nice looking dataFrame, we should make sure the HTML page has a table in it!

we will be using a Cryptocurrency website as an HTML dataset. it has various crypto coins on it and has various details about each crypto

```
import requests
url = '<https://www.worldcoinindex.com/>'

crypto_url = requests.get(url)
crypto_url
```



```
crypto_url = requests.get(url)
crypto_url
```

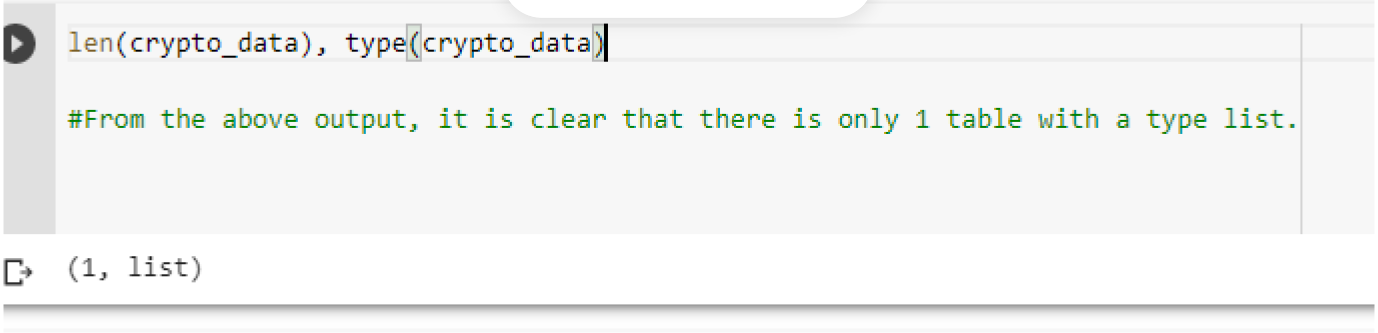
```
<Response [200]>
```

A photo by Author

Here, we defined the URL and then using requests.get() we sent a request to that URL and received a response as an acknowledgment [200] which means that we were able to connect with that web server.

Finally, we will pass crypto_url.text to the pd.read_html() function which will return you a list of dataframes where each element in that list is a table (dataframe) the cryptocurrency webpage has in it.

```
#print the length and the type of the dataframe
len(crypto_data), type(crypto_da
```



```
len(crypto_data), type(crypto_data)

#From the above output, it is clear that there is only 1 table with a type list.
```

```
(1, list)
```

A photo by Author

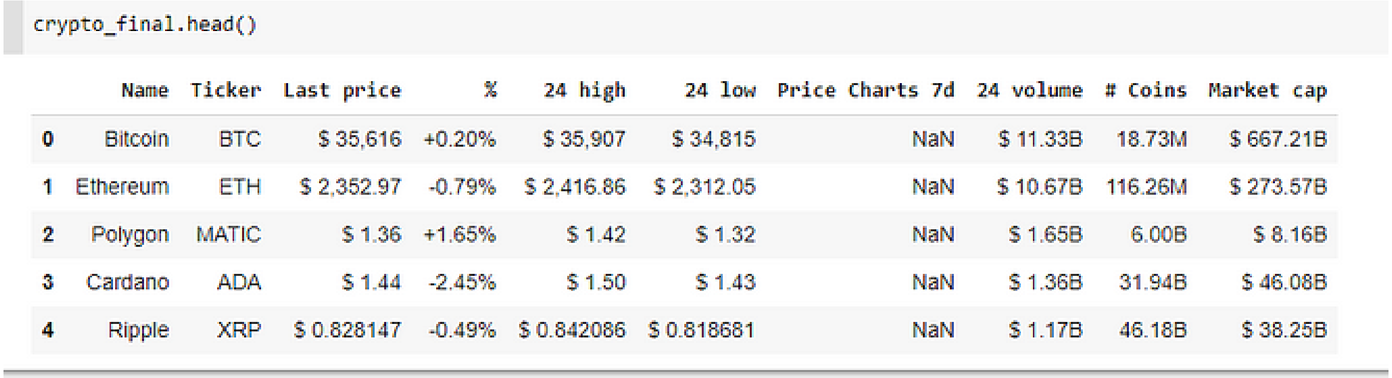
```
crypto_data = crypto_data[0]

#Let's remove the first and second columns since they do not have any
#useful information in them and keep all the rows.

crypto_final = crypto_data.iloc[:,2:]

#Finally, it's time to print the cryptocurrency dataframe!
crypto_final.head()
```

You can observe that Bitcoin has the most Market Capital.



```
crypto_final.head()
```

	Name	Ticker	Last price	%	24 high	24 low	Price Charts 7d	24 volume	# Coins	Market cap
0	Bitcoin	BTC	$ 35,616	+0.20%	$ 35,907	$ 34,815	NaN	$ 11.33B	18.73M	$ 667.21B
1	Ethereum	ETH	$ 2,352.97	-0.79%	$ 2,416.86	$ 2,312.05	NaN	$ 10.67B	116.26M	$ 273.57B
2	Polygon	MATIC	$ 1.36	+1.65%	$ 1.42	$ 1.32	NaN	$ 1.65B	6.00B	$ 8.16B
3	Cardano	ADA	$ 1.44	-2.45%	$ 1.50	$ 1.43	NaN	$ 1.36B	31.94B	$ 46.08B
4	Ripple	XRP	$ 0.828147	-0.49%	$ 0.842086	$ 0.818681	NaN	$ 1.17B	46.18B	$ 38.25B

A photo by Author

I hope you like the article. Reach me on my LinkedIn and twitter.

Recommended Articles

1. NLP — Zero to Hero with Python
2. Python Data Structures Data-types and Objects
3. Exception Handling Concepts in Python
4. Why LSTM more useful than RNN in Deep Learning?
5. Neural Networks: The Rise of Recurrent Neural Networks
6. Fully Explained Linear Regression with Python
7. Fully Explained Logistic Regression with Python
8. Differences Between concat(), merge() and join() with Python
9. Lasso (l1) and Ridge (l2) Regularization Techniques
10. Confusion Matrix in Machine Learning

Python Programming Artificial Intelligence Data Science Machine Learning