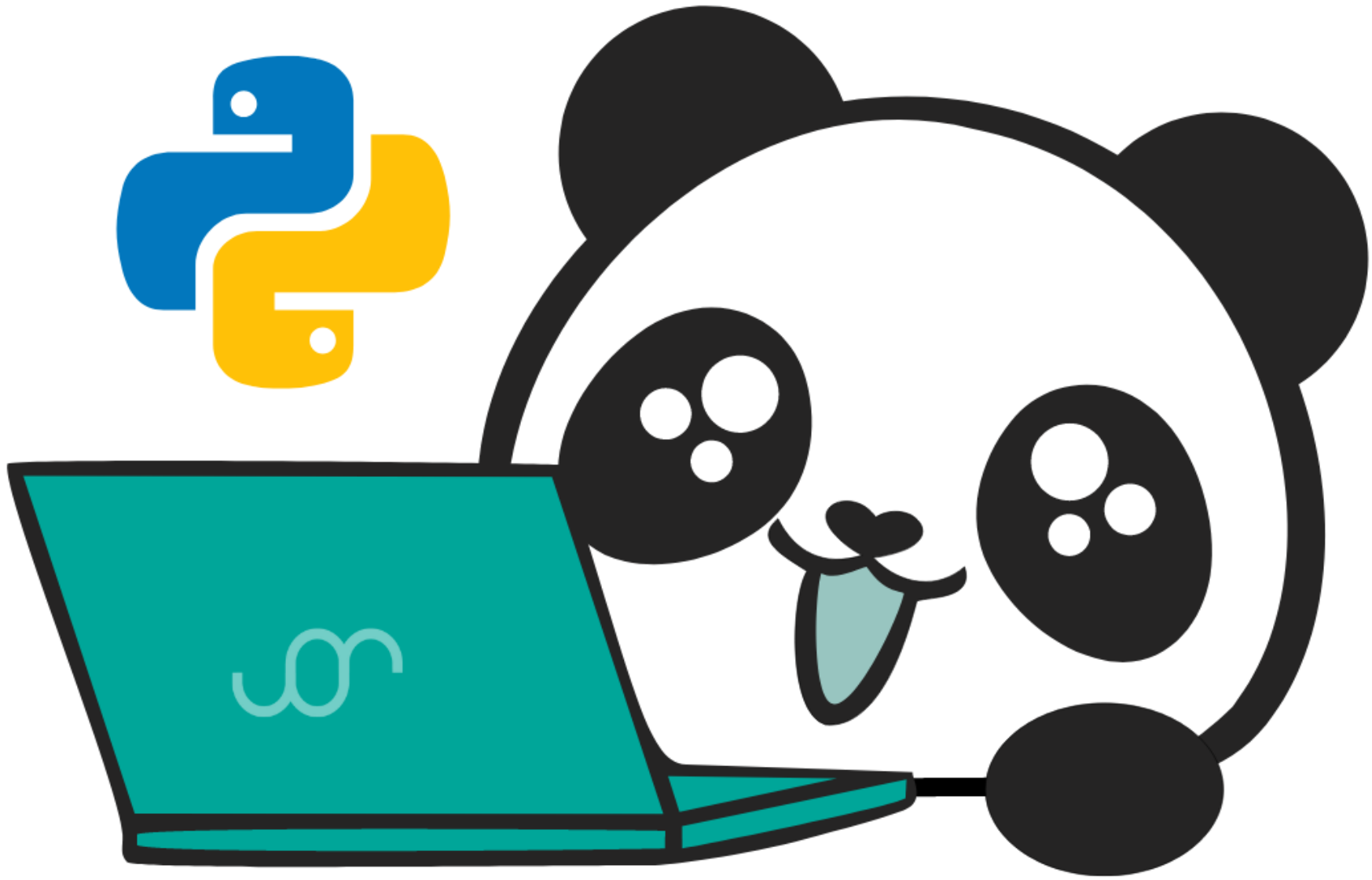


pandas

BEGINNER'S CODE GUIDE

ABHISHEK MISHRA
@abhishekmishra3

Data Structure	Dimensionality	Format	View
Series	1D	Column	
DataFrame	2D	Single Sheet	
Panel	3D	Multiple Sheets	

	name
0	Rukshan
1	Prasadi
2	Gihan
3	Hansana

	age
0	25
1	25
2	26
3	24

	marks
0	85
1	90
2	70
3	80

	name	age	marks
0	Rukshan	25	85
1	Prasadi	25	90
2	Gihan	26	70
3	Hansana	24	80

	name	age	marks
0	Rukshan	25	85
1	Prasadi	25	90
2	Gihan	26	70
3	Hansana	24	80

Series + Series = DataFrame

	apples
0	3
1	2
2	0
3	1

+

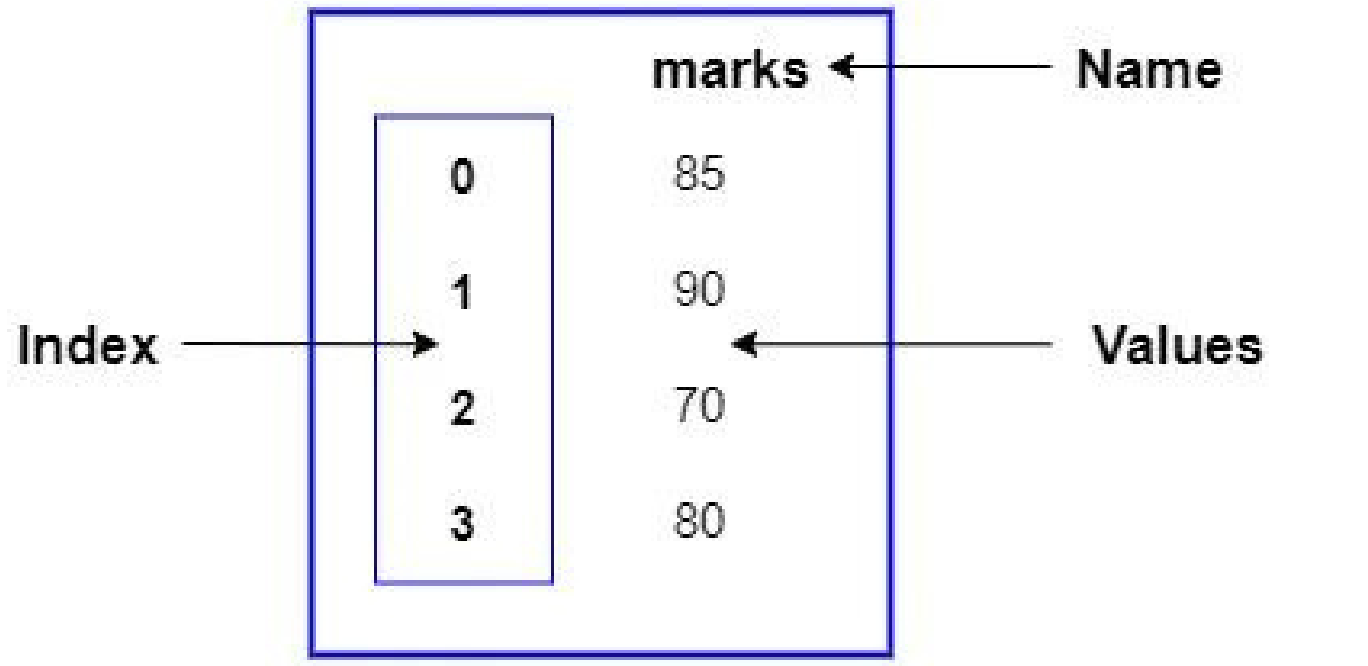
	oranges
0	0
1	3
2	7
3	2

=

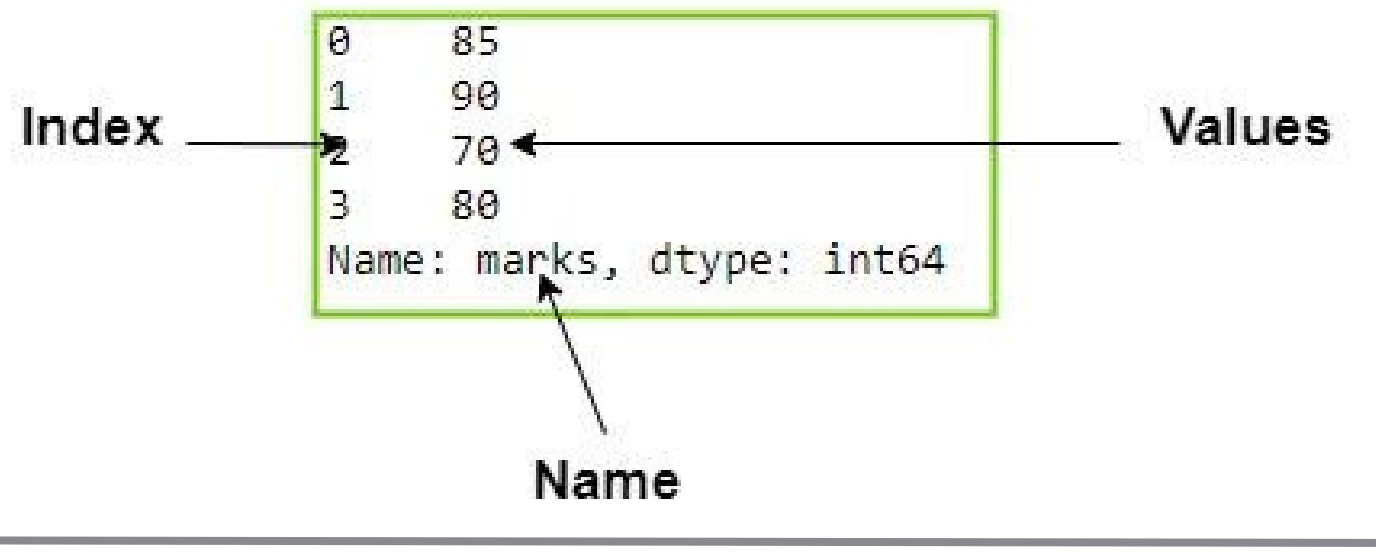
	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

Series

- **One-dimensional data (Values)**
- **Index**

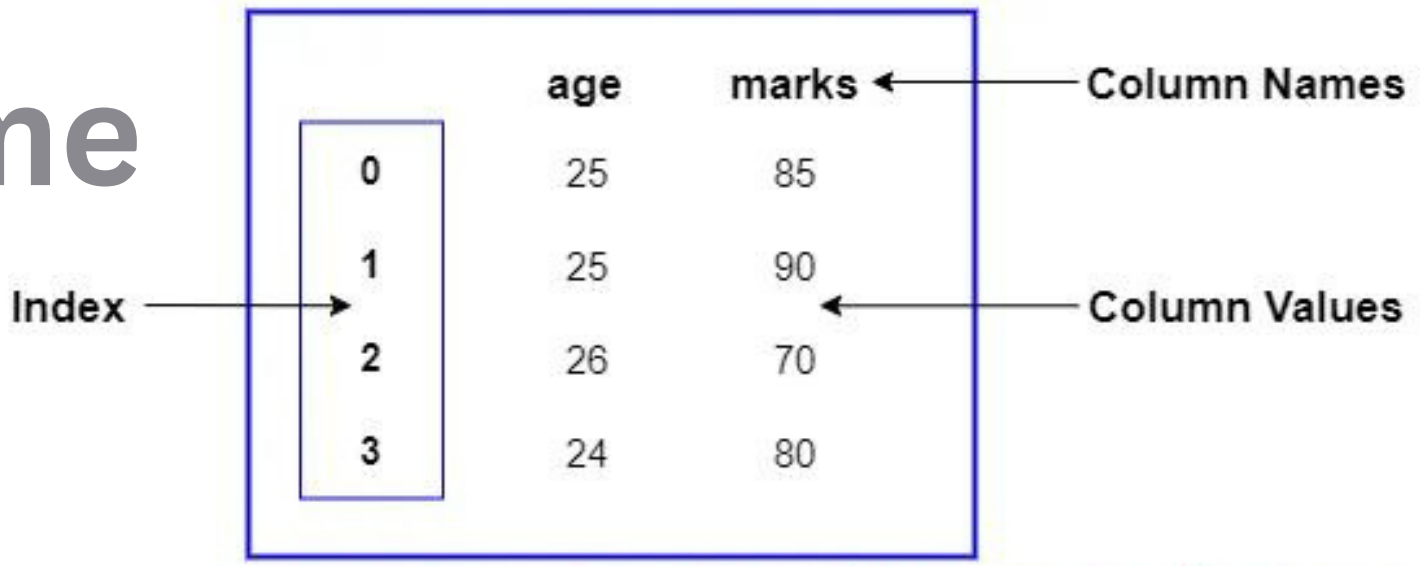



```
pd.Series([85, 90, 70, 80], name='marks')
```

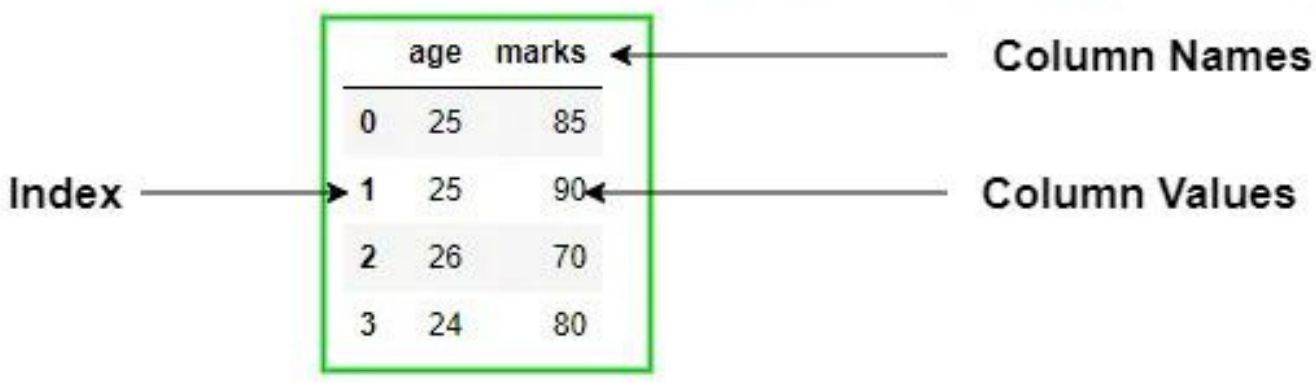


DataFrame

- **Two-dimensional data (Values)**
- **Row Index**
- **Column Index**




```
an_array = np.array([[25, 85], [25, 90], [26, 70], [24, 80]])
pd.DataFrame(an_array, columns=['age','marks'])
```



Pandas library in Python

used for working with data sets.
It analyzes, cleans, explores, manipulates data
Pandas: 'Panel Data' and 'Python Data Analysis'

import it in your applications

```
import pandas as pd
```

Pandas provide two data structures that shape data into a readable form: Series & DataFrame

Series

one-dimensional data structure like a table column
comprises of key-value pair. keys/labels are the indices & values are values stored on that index.

Initializing a series

pandas.Series()

```
# Creating empty series

a = pd.Series()

print(a)
```

Create Series from a List

```
# Create list

a = [9, np.nan, 7]
```

```
a    # list output
```

```
[9, nan, 7]
```

```
# convert to Pandas series

b = pd.Series(a) # capital S
```

```
b # series output
```

```
0    9.0
1    NaN
2    7.0
dtype: float64
```

Check Data Structure type

```
print(type(b))
```

```
<class 'pandas.core.series.Series'>
```

Series from ndarray Numpy

```
import numpy as np
```

```
# create Numpy array

a = np.array([6,5,4])
```

```
a    # Numpy array
```

```
array([6, 5, 4])
```

```
# convert to Pandas Series

b = pd.Series(a)
```

```
b    # series output
```

```
0    6
1    5
2    4
dtype: int32
```

Series from Dictionary

```
# create Dictionary

d = {"b": 1, "a": 0, "c": 2}
```

```
d  # dictionary output
```

```
{'b': 1, 'a': 0, 'c': 2}
```

```
# convert to pandas series

c = pd.Series(d)
```

```
c
```

```
b    1
a    0
c    2
dtype: int64
```

Labels / Index / Keys in series

Every element in the series has a index/label.
Default indices are 0 to N-1, like array indexes
We can specify index using 'index' parameter; values allow repetition

```
a = pd.Series([9,8,7],
              index=[3,4,5])

a
```

```
3    9
4    8
5    7
dtype: int64
```

```
b = pd.Series([5.4,3,2.5],
    index=['one','one','two'])


b
```

```
one    5.4
one    3.0
two    2.5
dtype: float64
```

Accessing Series Element

.iloc[] or [] to access default/built-in labels
.loc[] method for user-defined labels/indices

```
# default index 0


b.iloc[0]
```

```
5.4
```

```
# default index 1


b[1]
```

```
3.0
```

```
# user define index 'two'


b.loc['two']
```

```
2.5
```

DataFrame

> it's a two-dimensional data structure like a spreadsheet.
> collection of two or more series with common indices.

Initializing a DataFrame

pandas.DataFrame()

```
# Calling DataFrame constructor

df = pd.DataFrame()

print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

Dictionary to DataFrame

```
# create dictionary

data = {
   "k1": [9,8,7],
   "k2": ['a','b','c']
}
```

```
data   # print dictionary
```

```
{'k1': [9, 8, 7], 'k2': ['a', 'b', 'c']}
```

```
# convert to DataFrame

df = pd.DataFrame(data)
```

```
df  # print dataframe
```

	k1	k2
0	9	a
1	8	b
2	7	c

List to DataFrame

```
a = [3,4,5]

a
```

```
[3, 4, 5]
```

```
df = pd.DataFrame(a)

df
```

	0
0	3
1	4
2	5

```
# check data structure type

type(df)
```

```
pandas.core.frame.DataFrame
```

Named Index

```
data = {
   "k1": [9,8,7],
   "k2": ['a','b','c']
}

df = pd.DataFrame(data,
          index=['x','y','z'])

df
```

	k1	k2
x	9	a
y	8	b
z	7	c

Access DataFrame element

```
# default/in-built index

df.iloc[0]
```

```
k1    9
k2    a
Name: x, dtype: object
```

```
# named index

df.loc['y']
```

```
k1    8
k2    b
Name: y, dtype: object
```

Load Files Into a DataFrame

If your data sets are stored in a file, Pandas can load them into a DataFrame.

large DataFrame with many rows, Pandas will only return the first 5 rows, and the last 5 rows

```
df = pd.read_csv('data.csv')
```

df

	Product Name	Sale Price	Mrp	Number Of Ratings	Ram
0	APPLE iPhone 8 Plus (Gold, 64 GB)	49900	49900	3431	2 GB
1	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900	84900	3431	2 GB
2	APPLE iPhone 8 Plus (Silver, 256 GB)	84900	84900	3431	2 GB
3	APPLE iPhone 8 (Silver, 256 GB)	77000	77000	11202	2 GB
4	APPLE iPhone 8 (Gold, 256 GB)	77000	77000	11202	2 GB
...	...	...	...	...	...
57	APPLE iPhone SE (Black, 64 GB)	29999	39900	95909	4 GB
58	APPLE iPhone 11 (Purple, 64 GB)	46999	54900	43470	4 GB
59	APPLE iPhone 11 (White, 64 GB)	46999	54900	43470	4 GB
60	APPLE iPhone 11 (Black, 64 GB)	46999	54900	43470	4 GB
61	APPLE iPhone 11 (Red, 64 GB)	46999	54900	43470	4 GB

62 rows × 5 columns

to_string()

to print the entire DataFrame

```
df = pd.read_csv('data.csv')

print(df.to_string())
```

Increase the maximum number of rows to display the entire DataFrame

```
# change the maximum rows number

pd.options.display.max_rows = 9999

pd.read_csv('data.csv')
```

Read JSON

```
pd.read_json('data.json')
```

	Duration	Maxpulse	Calories
0	60	130	409.1
1	60	145	479.0
2	60	135	340.0
3	45	175	282.4
4	45	148	406.0
5	60	127	300.5

Load a Python Dictionary into a DataFrame

```
data = {
  "Duration":{
```

```
    "0":60,
    "1":60,
    "2":60,
  },
  "Pulse":{
    "0":110,
    "1":117,
    "2":103,
  },
  "Maxpulse":{
    "0":130,
    "1":145,
    "2":135,
    }
}

df = pd.DataFrame(data)

print(df)
```

```
   Duration  Pulse  Maxpulse
0        60    110       130
1        60    117       145
2        60    103       135
```

head()

```
returns the headers and a specified number of rows,
starting from the top.

by default returns 5 rows from top
```

```
df = pd.read_csv('data.csv')

df.head()
```

	Product Name	Sale Price	Mrp	Number Of Ratings	Ram
0	APPLE iPhone 8 Plus (Gold, 64 GB)	49900	49900	3431	2 GB
1	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900	84900	3431	2 GB
2	APPLE iPhone 8 Plus (Silver, 256 GB)	84900	84900	3431	2 GB
3	APPLE iPhone 8 (Silver, 256 GB)	77000	77000	11202	2 GB
4	APPLE iPhone 8 (Gold, 256 GB)	77000	77000	11202	2 GB

tail()

```
returns the headers and a specified number of rows,
starting from the bottom.

by default returns 5 rows from bottom
```

```
df.tail()
```

	Product Name	Sale Price	Mrp	Number Of Ratings	Ram
57	APPLE iPhone SE (Black, 64 GB)	29999	39900	95909	4 GB
58	APPLE iPhone 11 (Purple, 64 GB)	46999	54900	43470	4 GB
59	APPLE iPhone 11 (White, 64 GB)	46999	54900	43470	4 GB
60	APPLE iPhone 11 (Black, 64 GB)	46999	54900	43470	4 GB
61	APPLE iPhone 11 (Red, 64 GB)	46999	54900	43470	4 GB

info()

gives information about data type of each column, Non-Null value coumn

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Product Name       62 non-null     object
 1   Sale Price         62 non-null     int64 
 2   Mrp                62 non-null     int64 
 3   Number Of Ratings  62 non-null     int64 
 4   Ram                62 non-null     object
dtypes: int64(3), object(2)
memory usage: 2.6+ KB
```

```

```