😊 DataFrames in Python

A **DataFrame** in Python, primarily from the pandas library, is a highly flexible and efficient data structure designed for working with structured data. It allows you to store and manipulate tabular data where each column can contain data of a different type (e.g., integer, float, string, boolean). This makes it much more powerful than simple arrays or lists when dealing with real-world data.

Core Concepts:

- **Rows and Columns**: A DataFrame is organized in rows and columns, similar to a database table or Excel spreadsheet.
- **Labeled Axes**: Each row and column can be labeled for easy access and manipulation.
- **Heterogeneous Data Types**: Unlike NumPy arrays, which require a single data type, DataFrames support multiple data types within the same table.
- **Size-Mutable**: You can add or remove rows/columns dynamically.
- **Indexing and Selection**: Provides rich options for selecting and filtering data, including label-based (.loc) and position-based (.iloc) indexing.

Use Cases:

1. **Data Ingestion**: Reading from files (CSV, Excel, JSON), databases (SQL), or APIs.
2. **Data Cleaning**: Handling missing values, correcting data types, renaming columns, etc.
3. **Data Transformation**: Aggregation, pivot tables, and reshaping.
4. **Statistical Analysis**: Descriptive statistics, correlation analysis, and regression modeling.
5. **Visualization**: Easily integrates with visualization libraries like Matplotlib and Seaborn.
6. **Machine Learning**: Preprocessing datasets before feeding them into ML algorithms.

Benefits:

- **Efficiency**: Optimized for performance even with large datasets.
- **Ease of Use**: Simple syntax and methods reduce the need for complex code.
- **Integration**: Works well with other Python libraries like NumPy, Scikit-learn, and TensorFlow.
- **Documentation and Community**: Well-documented and widely supported by the data science community.

In essence, DataFrames are the go-to structure in Python for anyone dealing with structured or semi-structured data, making complex data analysis tasks much more accessible and organized.

😄 To begin with the Lab

1. We start by importing NumPy and Pandas in our Jupyter notebook.

[2]:
```
import numpy as np
import pandas as pd
```

2. Now we are going to create a DataFrame from Python objects.
3. The code begins by setting a random seed to ensure reproducible results and generates a 4x3 array of random integers between 0 and 100. This array is then used to create a pandas DataFrame.
4. Initially, the DataFrame is created without labels, but it is later updated to include custom row indices representing U.S. states and column names for the months January to March.
5. The final DataFrame shows a labeled table of random data.
6. The df.info() function provides a concise summary of the DataFrame, including the number of entries, data types, and memory usage.

[6]:
```
# Make sure the seed is in the same cell as the random call
np.random.seed(101)
mydata = np.random.randint(0,101,(4,3))
```

[8]:
```
mydata
```

[8]:
```
array([[95, 11, 81],
       [70, 63, 87],
       [75,  9, 77],
       [40,  4, 63]])
```

[10]:
```
myindex = ['CA','NY','AZ','TX']
```

[12]:
```
mycolumns = ['Jan','Feb','Mar']
```

[14]:
```
df = pd.DataFrame(data=mydata)
df
```

[14]:

	0	1	2
0	95	11	81
1	70	63	87
2	75	9	77
3	40	4	63

[16]:
```
df = pd.DataFrame(data=mydata,index=myindex)
df
```

[16]:

	0	1	2
CA	95	11	81
NY	70	63	87
AZ	75	9	77
TX	40	4	63

[18]:
```
df = pd.DataFrame(data=mydata,index=myindex,columns=mycolumns)
df
```

[18]:

	Jan	Feb	Mar
CA	95	11	81
NY	70	63	87
AZ	75	9	77
TX	40	4	63

[20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4 entries, CA to TX
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Jan     4 non-null      int32
 1   Feb     4 non-null      int32
 2   Mar     4 non-null      int32
dtypes: int32(3)
memory usage: 80.0+ bytes
```

7. We will use a DataFrame to read a CSV file in the Jupyter Notebook. You will get the CSV files with the Notes.
8. The code reads a CSV file named 'tips.csv' into a Pandas DataFrame called 'df'.
9. Once loaded, ‘df’ will display the contents of the CSV as a structured table, where each row represents a record (e.g., a restaurant bill and tip), and each column represents a feature such as total bill, tip, gender of the server, day of the week, time, and size of the party.

[26]:
```
df = pd.read_csv('tips.csv')
```

[28]:
```
df
```

[28]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251
...	...	...	...	...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842	Sat2657
240	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404	Sat1766
241	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196	Sat3880
242	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950	Sat17
243	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139	Thur672

10. These commands provide a comprehensive overview of the `df` DataFrame:

- `df.columns` returns the names of the columns in the DataFrame.
- `df.index` returns the row index range.
- `df.head(3)` shows the first three rows of the DataFrame.
- `df.tail(3)` shows the last three rows.
- `df.info()` gives a summary including column data types and non-null counts.
- `len(df)` returns the number of rows.
- `df.describe()` generates summary statistics for numeric columns (like mean, min, max).
- `df.describe().transpose()` flips the summary table to display columns as rows, making the statistics easier to read for each feature.

[30]:
```
df.columns
```

[30]:
```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size',
       'price_per_person', 'Payer Name', 'CC Number', 'Payment ID'],
      dtype='object')
```

[32]:
```
df.index
```

[32]:
```
RangeIndex(start=0, stop=244, step=1)
```

[34]:
```
df.head(3)
```

[34]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458

[36]:
```
df.tail(3)
```

[36]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
241	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196	Sat3880
242	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950	Sat17
243	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139	Thur672

[38]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   total_bill        244 non-null    float64
 1   tip               244 non-null    float64
 2   sex               244 non-null    object
 3   smoker            244 non-null    object
 4   day               244 non-null    object
 5   time              244 non-null    object
 6   size              244 non-null    int64
 7   price_per_person  244 non-null    float64
 8   Payer Name        244 non-null    object
 9   CC Number         244 non-null    int64
 10  Payment ID        244 non-null    object
dtypes: float64(3), int64(2), object(6)
memory usage: 21.1+ KB
```

[40]:
```
len(df)
```

[40]:
```
244
```

[42]:
```
df.describe()
```

[42]:

	total_bill	tip	size	price_per_person	CC Number
count	244.000000	244.000000	244.000000	244.000000	2.440000e+02
mean	19.785943	2.998279	2.569672	7.888197	2.563496e+15
std	8.902412	1.383638	0.951100	2.914234	2.369340e+15
min	3.070000	1.000000	1.000000	2.880000	6.040679e+10
25%	13.347500	2.000000	2.000000	5.800000	3.040731e+13
50%	17.795000	2.900000	2.000000	7.255000	3.525318e+15
75%	24.127500	3.562500	3.000000	9.390000	4.553675e+15
max	50.810000	10.000000	6.000000	20.270000	6.596454e+15

[44]:
```
df.describe().transpose()
```

[44]:

	count	mean	std	min	25%	50%	75%	max
total_bill	244.0	1.978594e+01	8.902412e+00	3.070000e+00	1.334750e+01	1.779500e+01	2.412750e+01	5.081000e+01
tip	244.0	2.998279e+00	1.383638e+00	1.000000e+00	2.000000e+00	2.900000e+00	3.562500e+00	1.000000e+01
size	244.0	2.569672e+00	9.510998e-01	1.000000e+00	2.000000e+00	2.000000e+00	3.000000e+00	6.000000e+00
price_per_person	244.0	7.888197e+00	2.914234e+00	2.880000e+00	5.800000e+00	7.255000e+00	9.390000e+00	2.027000e+01
CC Number	244.0	2.563496e+15	2.369340e+15	6.040679e+10	3.040731e+13	3.525318e+15	4.553675e+15	6.596454e+15

11. Now we are going to learn about selection and indexing. We will begin by learning how to extract information based on the columns.
12. By running the below cells first, we looked at the first few columns of our file, then we displayed only a single column from our data frame.

[46]:
```
df.head()
```

[46]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251

Grab a Single Column

[48]:
```
df['total_bill']
```

[48]:
```
0      16.99
1      10.34
2      21.01
3      23.68
4      24.59
       ...
239    29.03
240    27.18
241    22.67
242    17.82
243    18.78
Name: total_bill, Length: 244, dtype: float64
```

13. We can also use it to display multiple columns from the data frame.

[52]:
```
# Note how its a python list of column names! Thus the double brackets.
df[['total_bill','tip']]
```

[52]:

	total_bill	tip
0	16.99	1.01
1	10.34	1.66
2	21.01	3.50
3	23.68	3.31
4	24.59	3.61
...	...	...
239	29.03	5.92
240	27.18	2.00
241	22.67	2.00
242	17.82	1.75
243	18.78	3.00

244 rows × 2 columns

14. This code adds two new calculated columns to the existing DataFrame:
    - tip_percentage is created by dividing the tip amount by the total_bill and multiplying by 100 to express it as a percentage. This helps analyze how much tip was given relative to the total bill.
    - price_per_person is calculated by dividing the total_bill by the number of people (size) at the table, providing insight into individual spending.
15. The head() function is used after each addition to preview the updated DataFrame.

[54]:
```
df['tip_percentage'] = 100* df['tip'] / df['total_bill']
```

[56]:
```
df.head()
```

[56]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID	tip_percentage
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959	5.944673
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608	16.054159
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458	16.658734
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260	13.978041
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251	14.680765

[58]:
```
df['price_per_person'] = df['total_bill'] / df['size']
```

[60]:
```
df.head()
```

[60]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID	tip_percentage
0	16.99	1.01	Female	No	Sun	Dinner	2	8.495000	Christy Cunningham	3560325168603410	Sun2959	5.944673
1	10.34	1.66	Male	No	Sun	Dinner	3	3.446667	Douglas Tucker	4478071379779230	Sun4608	16.054159
2	21.01	3.50	Male	No	Sun	Dinner	3	7.003333	Travis Walters	6011812112971322	Sun4458	16.658734
3	23.68	3.31	Male	No	Sun	Dinner	2	11.840000	Nathaniel Harris	4676137647685994	Sun5260	13.978041
4	24.59	3.61	Female	No	Sun	Dinner	4	6.147500	Tonya Carter	4832732618637221	Sun2251	14.680765

16. This code uses NumPy's round() function to round the values in the price_per_person column of the DataFrame to two decimal places.
17. Since pandas is built on top of NumPy, it can directly apply such universal functions to its columns. After rounding, head() displays the first few rows to verify the update.

[64]:
```
# Because pandas is based on numpy, we get awesome capabilities with numpy's universal functions!
df['price_per_person'] = np.round(df['price_per_person'],2)
```

[66]:
```
df.head()
```

[66]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID	tip_percentage
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959	5.944673
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608	16.054159
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458	16.658734
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260	13.978041
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251	14.680765

18. Here, the code removes the column named tip_percentage from the DataFrame df. The axis=1 argument specifies that a column (not a row) is being dropped. After the column is removed, df.head() is called to display the first few rows of the updated DataFrame.

[68]:
```
# df.drop('tip_percentage',axis=1)
```

[70]:
```
df = df.drop("tip_percentage",axis=1)
```

[72]:
```
df.head()
```

[72]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251

19. This sequence of commands demonstrates how to change and then reset the index of a DataFrame in pandas:

    - `df.set_index('Payment ID')` sets the `'Payment ID'` column as the index of the DataFrame, helping uniquely identify each row.
    - Assigning it back using `df = df.set_index('Payment ID')` updates the DataFrame to reflect this change.
    - `df.reset_index()` reverses the change by converting the index back into a regular column, restoring the default integer index.
    - These operations are useful for organizing, accessing, or merging data more effectively based on specific identifiers.

[74]:
```
df.head()
```

[74]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251

[76]:
```
df.index
```

[76]:
```
RangeIndex(start=0, stop=244, step=1)
```

[78]:
```
df.set_index('Payment ID')
```

[78]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221
...	...	...	...	...	...	...	...	...	...	...
Sat2657	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842
Sat1766	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404
Sat3880	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196
Sat17	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950
Thur672	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139

244 rows × 10 columns

```
[80]: df.head()
```

[80]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251

```
[82]: df = df.set_index('Payment ID')
```

```
[84]: df.head()
```

[84]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

```
[86]: df = df.reset_index()
```

```
[88]: df.head()
```

[88]:

	Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
0	Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
1	Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
2	Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
3	Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
4	Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

20. What we have done above, we are going to do the same, but this time with rows. So, I'll just be attaching the snapshots if you want to explore. Just open the Jupyter notebook for a better understanding.
21. This sequence of commands shows various ways to access, modify, and extend a pandas DataFrame using both integer-based and label-based indexing:

    - `df.iloc[0]` retrieves the first row by its integer position, while `df.loc['Sun2959']` accesses a row by its index label.
    - `df.iloc[0:4]` returns the first four rows, and `df.loc[['Sun2959','Sun5260']]` fetches specific rows by index names.
    - `df.drop('Sun2959', axis=0)` removes a row from the DataFrame without altering the original unless reassigned.
    - `one_row = df.iloc[0]` stores the first row as a Series, and `type(one_row)` confirms this.
    - Finally, `pd.concat()` is used to re-add that row back to the DataFrame, ensuring it's in the correct format (`to_frame().T`) to match the DataFrame's shape.

- This showcases how to manipulate rows flexibly and handle row-level operations like removal and reinsertion.

[90]:
```
df.head()
```

[90]:

	Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
0	Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
1	Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
2	Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
3	Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
4	Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

[92]:
```
df = df.set_index('Payment ID')
```

[94]:
```
df.head()
```

[94]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

[96]:
```
# Integer Based
df.iloc[0]
```

[96]:
```
total_bill                      16.99
tip                              1.01
sex                            Female
smoker                             No
day                               Sun
time                           Dinner
size                                2
price_per_person                 8.49
Payer Name          Christy Cunningham
CC Number             3560325168603410
Name: Sun2959, dtype: object
```

[98]:
```
# Name Based
df.loc['Sun2959']
```

[98]:
```
total_bill                      16.99
tip                              1.01
sex                            Female
smoker                             No
day                               Sun
time                           Dinner
size                                2
price_per_person                 8.49
Payer Name          Christy Cunningham
CC Number             3560325168603410
Name: Sun2959, dtype: object
```

[100]:
```
df.iloc[0:4]
```

[100]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994

[102]:
```
df.loc[['Sun2959','Sun5260']]
```

[102]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994

[104]:
```
df.head()
```

[104]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

[106]:
```
df.drop('Sun2959',axis=0).head()
```

[106]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221
Sun9679	25.29	4.71	Male	No	Sun	Dinner	4	6.32	Erik Smith	213140353657882

[110]:
```
one_row = df.iloc[0]
```

[112]:
```
one_row
```

[112]:
```
total_bill                      16.99
tip                              1.01
sex                            Female
smoker                             No
day                               Sun
time                           Dinner
size                                2
price_per_person                 8.49
Payer Name          Christy Cunningham
CC Number             3560325168603410
Name: Sun2959, dtype: object
```

[114]:
```
type(one_row)
```

[114]:
```
pandas.core.series.Series
```

[116]:
```
df.tail()
```

[116]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sat2657	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842
Sat1766	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404
Sat3880	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196
Sat17	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950
Thur672	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139

[120]:
```
df = pd.concat([df, one_row.to_frame().T if isinstance(one_row, pd.Series) else one_row])

df.tail()
```

[120]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sat1766	27.18	2.0	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404
Sat3880	22.67	2.0	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196
Sat17	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950
Thur672	18.78	3.0	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410