In [1]:

```
# A Pandas series can also be converted to a Python list using
# the tolist() method, as shown in the script below:
# Converting to List

import pandas as pd
import numpy as np
my_series = pd.Series([5, 8, 2, 11, 9])
print(my_series.tolist())
```

```
[5, 8, 2, 11, 9]
```

In [2]:

```
# A Pandas dataframe is a tabular data structure that stores
# data in the form of rows and columns. As a standard, the rows
# correspond to records while columns refer to attributes. In
# simplest words, a Pandas dataframe is a collection of series.
# As is the case with a series, there are multiple ways to create a
# Pandas dataframe.
# To create an empty dataframe, you can use the DataFrame
# class from the Pandas module, as shown below:
# empty pandas dataframe

import pandas as pd
my_df = pd.DataFrame()
print(my_df)
```

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

In [3]:

```
# You can create a Pandas dataframe using a list of lists. Each
# sublist in the outer list corresponds to a row in a dataframe.
# Each item within a sublist becomes an attribute value.
# To specify column headers, you need to pass a list of values to
# the columns attribute of DataFrame class.
# Here is an example of how you can create a Pandas dataframe
# using a list.
# dataframe using list of lists

import pandas as pd
scores = [['Mathematics', 85], ['English', 91], ['History', 95]]
my_df = pd.DataFrame(scores, columns = ['Subject', 'Score'])
my_df
```

Out[3]:

	Subject	Score
0	Mathematics	85
1	English	91
2	History	95

In [4]:

```
# Similarly, you can create a Pandas dataframe using a
# dictionary. One of the ways is to create a dictionary where
# keys correspond to column headers. In contrast,
# corresponding dictionary values are a list, which corresponds
# to the column values in the Pandas dataframe.
# Here is an example for your reference:
# dataframe using dictionaries

import pandas as pd
scores = {'Subject':["Mathematics", "History", "English", "Science", "Arts"], 'Score':[9
my_df = pd.DataFrame(scores)
my_df
```

Out[4]:

	Subject	Score
0	Mathematics	98
1	History	75
2	English	68
3	Science	82
4	Arts	99

In [5]:

```
# Another way to create a Pandas dataframe is using a list of
# dictionaries. Each dictionary corresponds to one row. Here is
#an example of how to do that.

# dataframe using list of dictionaries
import pandas as pd
scores = [{'Subject':'Mathematics', 'Score':85},{'Subject':'History', 'Score':98},
          {'Subject':'English', 'Score':76}, {'Subject':'Science', 'Score':72},
          {'Subject':'Arts', 'Score':95},]
my_df = pd.DataFrame(scores)
my_df
```

Out[5]:

	Subject	Score
0	Mathematics	85
1	History	98
2	English	76
3	Science	72
4	Arts	95

In [6]:

```
# The dictionaries within the list used to create a Pandas
# dataframe need not be of the same size.
# For example, in the script below, the fourth dictionary in the
# list contains only one item, unlike the rest of the dictionaries in
# this list. The corresponding dataframe will contain a null value
# in place of the second item, as shown in the output of the
# script below:

# dataframe using list of dictionaries
# with null items

import pandas as pd
scores = [{'Subject':'Mathematics', 'Score':85}, {'Subject':'History', 'Score':98},
          {'Subject':'English', 'Score':76}, {'Score':72}, {'Subject':'Arts', 'Score':95
my_df = pd.DataFrame(scores)
my_df
```

Out[6]:

	Subject	Score
0	Mathematics	85
1	History	98
2	English	76
3	NaN	72
4	Arts	95

In [7]:

```
# Let's now see some of the basic operations that you can
# perform on Pandas dataframes.
# To view the top(N) rows of a dataframe, you can call the
# head() method, as shown in the script below:
# viewing header

import pandas as pd
scores = [
{'Subject':'Mathematics', 'Score':85},
{'Subject':'History', 'Score':98},
{'Subject':'English', 'Score':76},
{'Subject':'Science', 'Score':72},
{'Subject':'Arts', 'Score':95},
]
my_df = pd.DataFrame(scores)
my_df.head(2)
```

Out[7]:

	Subject	Score
0	Mathematics	85
1	History	98

In [8]:

```
# To view the last N rows, you can use the tail() method. Here is
# an example:
# viewing tail

import pandas as pd
scores = [
{'Subject':'Mathematics', 'Score':85},
{'Subject':'History', 'Score':98},
{'Subject':'English', 'Score':76},
{'Subject':'Science', 'Score':72},
{'Subject':'Arts', 'Score':95},
]
my_df = pd.DataFrame(scores)
my_df.tail(2)
```

Out[8]:

	Subject	Score
3	Science	72
4	Arts	95

In [9]:

```
# You can also get a summary of your Pandas dataframe using
# the info() method.
# gettingdataframe info

import pandas as pd
scores = [
{'Subject':'Mathematics', 'Score':85},
{'Subject':'History', 'Score':98},
{'Subject':'English', 'Score':76},
{'Subject':'Science', 'Score':72},
{'Subject':'Arts', 'Score':95},
]
my_df = pd.DataFrame(scores)
my_df.info()

# In the output below, you can see the number of entries in your
# Pandas dataframe, the number of columns along with their
# column type, and so on.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   Subject  5 non-null      object
 1   Score    5 non-null      int64 
dtypes: int64(1), object(1)
memory usage: 208.0+ bytes
```

In [10]:

```
# Finally, to get information such as mean, minimum, maximum,
# standard deviation, etc., for numeric columns in your Pandas
# dataframe, you can use the describe() method, as shown in
# the script below:
# getting info about numeric columns

import pandas as pd
scores = [
{'Subject':'Mathematics', 'Score':85},
{'Subject':'History', 'Score':98},
{'Subject':'English', 'Score':76},
{'Subject':'Science', 'Score':72},
{'Subject':'Arts', 'Score':95},
]
my_df = pd.DataFrame(scores)
my_df.describe()
```

Out[10]:

	Score
count	5.000000
mean	85.200000
std	11.388591
min	72.000000
25%	76.000000
50%	85.000000
75%	95.000000
max	98.000000