

```
import numpy as np
```

```
import pandas as pd
```

```
# Pandas Series
```

```
# Pandas Series can be created using list
```

```
l = [10,20,30]
s = pd.Series(l)
print (s)
```

```
0    10
1    20
2    30
dtype: int64
```

```
type(s)
```

```
pandas.core.series.Series
def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool |
None=None, fastpath: bool | lib.NoDefault=lib.no_default) -> None

/usr/local/lib/python3.11/dist-packages/pandas/core/series.py
One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object
supports both integer- and label-based indexing and provides a host of
```

```
s.shape
```

```
(3,)
```

```
l = [10,20,30]
s = pd.Series(l,index=['a','b','c'])
print (s)
```

```
a    10
b    20
c    30
dtype: int64
```

```
# creating Series using dictionary
```

```
d = {'a':10,'b':20,'c':30}
s = pd.Series(d)
print (s)
```

```
a    10
b    20
c    30
dtype: int64
```

```
# Pandas dataframe
```

```
d = {'a':[1,2,3], 'b':[4,5,6], 'c':[7,8,9]}
df = pd.DataFrame(d)
print (df)
```

```

↗
  a  b  c
0  1  4  7
1  2  5  8
2  3  6  9

```

```
type(d)
```

```
↗ dict
```

```
type(df)
```

```

↗
pandas.core.frame.DataFrame
def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype |
None=None, copy: bool | None=None) -> None

/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py
Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be

```

```
df.shape
```

```
↗ (3, 3)
```

```
# Creating DataFrame using Nested List
```

```

data = [[1,2,3],[4,5,6],[7,8,9]]
df = pd.DataFrame(data,columns=['a','b','c'])
df

```

```

↗
  a  b  c
0  1  2  3
1  4  5  6

```

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

```
# Create DataFrame using 2D array
```

```

arr_2d = np.array ([[1,2,3],[4,5,6]])
df2 = pd.DataFrame(arr_2d,columns=['a','b','c'])
df2

```

```

↗
  a  b  c
0  1  2  3
1  4  5  6

```


Next steps: [Generate code with df2](#) [View recommended plots](#) [New interactive sheet](#)

```
# Pandas
```




```

df = pd.DataFrame({'Age':[22,33,30,45,24],'Salary':[24000,17000, 44000, 51400,44600],'Gender': ["M", "F", "M", "F", "
df

```



	Age	Salary	Gender
0	22	24000	M
1	33	17000	F
2	30	44000	M
3	45	51400	F



Next steps:


[Generate code with df](#)

[View recommended plots](#)




[New interactive sheet](#)

# Set index

```
df2 = df.set_index('Age')
df2
```



	Salary	Gender
Age		
22	24000	M
33	17000	F
30	44000	M
45	51400	F



Next steps:


[Generate code with df2](#)

[View recommended plots](#)




[New interactive sheet](#)

# Reset index

```
df2.reset_index()
df2
```



	Salary	Gender
Age		
22	24000	M
33	17000	F
30	44000	M
45	51400	F




Next steps:

[Generate code with df2](#)



[View recommended plots](#)

[New interactive sheet](#)

```
df.reset_index()
```




	index	Age	Salary	Gender
0	0	22	24000	M
1	1	33	17000	F
2	2	30	44000	M
3	3	45	51400	F



```
# Drop a single row
```

```
df1 = df.drop(1)
df1
```




	Age	Salary	Gender
0	22	24000	M
2	30	44000	M
3	45	51400	F

Next steps:

[Generate code with df1](#)[View recommended plots](#)[New interactive sheet](#)

```
df3 = df.drop(columns = ["Age"])
df3
```




	Salary	Gender
0	24000	M
1	17000	F
2	44000	M
3	51400	F

Next steps:

[Generate code with df3](#)[View recommended plots](#)[New interactive sheet](#)


```
# sorting values in a dataframe
```

```
df.sort_values(by = 'Salary',ascending = True)
```



	Age	Salary	Gender
1	33	17000	F
0	22	24000	M
2	30	44000	M
4	24	44600	M


```
df.sort_values(by = 'Salary',ascending = False)
```





	Age	Salary	Gender
3	45	51400	F
4	24	44600	M
2	30	44000	M
0	22	24000	M

```
# Combining Dataframes
```

```
df1 = pd.DataFrame({"City":["new york","chicago","orlando"],"temperature": [21,14,35]})
df1
```




	City	temperature
0	new york	21
1	chicago	14






Next steps: [Generate code with df1](#) [View recommended plots](#) [New interactive sheet](#)

```
df3 = pd.DataFrame({"City" : ["new york", "Delhi", "Chennai"], "Humadity" : [68, 65, 77]})
df3
```




	City	Humadity
0	new york	68
1	Delhi	65


Next steps: [Generate code with df3](#) [View recommended plots](#) [New interactive sheet](#)

# Concatenation


```
pd.concat([df1,df3],axis = 0)
```




	city	temperature	City	Humadity
0	new york	21.0	NaN	NaN
1	chicago	14.0	NaN	NaN
2	orlando	35.0	NaN	NaN
0	NaN	NaN	new york	68.0
1	NaN	NaN	Delhi	65.0



```
pd.concat([df1,df3],ignore_index=True)
```



	city	temperature	City	Humadity
0	new york	21.0	NaN	NaN
1	chicago	14.0	NaN	NaN
2	orlando	35.0	NaN	NaN
3	NaN	NaN	new york	68.0
4	NaN	NaN	Delhi	65.0



```
pd.concat([df1,df2], ignore_index= False )
```

	city	temperature	Salary	Gender	
0	new york	21.0	NaN	NaN	
1	chicago	14.0	NaN	NaN	
2	orlando	35.0	NaN	NaN	
22	NaN	NaN	24000.0	M	
33	NaN	NaN	17000.0	F	
30	NaN	NaN	44000.0	M	
45	NaN	NaN	51400.0	F	

# Merging of Data Frame

```
left_df = pd.merge(df1,df3, on='City', how='left')
left_df
```

	City	temperature	Humadity	
0	new york	21	68.0	
1	chicago	14	NaN	

Next steps: [Generate code with left\\_df](#) [View recommended plots](#) [New interactive sheet](#)

```
right_df = pd.merge(df1,df3, on='City', how='right')
right_df
```

	City	temperature	Humadity	
0	new york	21.0	68	
1	Delhi	NaN	65	


Next steps: [Generate code with right\\_df](#) [View recommended plots](#) [New interactive sheet](#)




```
inner_df = pd.merge(df1,df3, on='City', how='inner')
inner_df
```

	City	temperature	Humadity	
0	new york	21	68	

# Outer Join all the records of both DataFrames

```
outer_df = pd.merge(df1,df3, on='City', how='outer')
outer_df
```



	City	temperature	Humadity	
0	Chennai	NaN	77.0	
1	Delhi	NaN	65.0	
2	chicago	14.0	NaN	
3	new york	21.0	68.0	