Pandas

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data easy.

It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python.

Additionally, it has the broader goal of becoming **the most powerful and flexible open source data analysis / manipulation tool available in any language**.

It is already well on its way toward this goal.

pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels.
- Any other form of observational / statistical data sets. The data actually need not be labelled at all to be placed into a pandas data structure.

The two primary data structures of pandas, **Series** (1-dimensional) and **DataFrame** (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

Data structures

Dimensions	Name	Description
1	Series	1D labelled homogeneously-typed array
2	DataFrame	General 2D labelled, size-mutable tabular structure with potentially heterogeneously-typed column

Series

Series is a one-dimensional labelled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.).

s = pd.Series(data, index=index)

NaN (not a number) is the standard missing data marker used in pandas.

We can use dictionary keys to access the values from Series.

Using the get method, a missing label will return None or specified default.

Example 1:

```
import numpy as np
import pandas as pd

s = pd.Series([1, 3, 5, np.nan, 6, 8])
print(s)
```

Note : Default indexing will begin with zero.

Example 2 :

```
import numpy as np
import pandas as pd

s = pd.Series([1, 3, 5, np.nan, 6],['a','b','c','d','e'])
print(s)
```

Note : Series can be instantiated from dictionary's also.

Example 3:

```
import pandas as pd

emp_data = {"idno":101,"name":"Ravi","salary":185000.00,
"status":False}
s = pd.Series(emp_data)
print(s)
```

Note: We can use indexing on Series

```
print(s[0]) # 101
print(s[1]) # Ravi
print(s[5])  # IndexError: index out of bounds
```

Note: We can use dictionary keys to access the values from Series

```
print(s['idno'])  # 101
print(s['name'])  # Ravi

print(s['designation'])  # KeyError: 'designation'
```

Note : Using the get method, a missing label will return None or specified default.

```
data = s.get(["designation"],np.NaN)
print(data) # nan


data = s.get(['designation'],"Sorry Wrong KEY")
print(data) # Sorry Wrong KEY
```

DataFrame

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types.

You can think of it like a spreadsheet or SQL table, or a dict of Series objects.

It is generally the most commonly used pandas object. Like Series, DataFrame accepts many different kinds of input:

- Dict of 1D ndarrays, lists, dicts, or Series
- 2-D numpy.ndarray
- Structured or record ndarray
- A Series
- Another DataFrame

Example 1 : dict of Series or dicts

```
import pandas as pd

employee_details = {
        "idno":pd.Series([101,102,103,104,105]),
"name":pd.Series(["Ravi","Kumar","Mohan","Krishna","Prasad"]),
"salary":pd.Series([185000.00,285000.00,125000.00,225000.00,100000.00])
                }
df = pd.DataFrame(employee_details)
print(df)


# 1st Employee Details
print("IDNO = ",df["idno"][0])  # IDNO =  101
print("NAME = ",df["name"][0])  # NAME =  Ravi
print("SALARY = ",df["salary"][0]) # SALARY =  185000.0
```

Example 2 : dict of ndarrays / lists

The ndarrays must all be the same length.

If an index is passed, it must clearly also be the same length as the arrays. If no index is passed, the result will be range(n), where n is the array length.

```
import pandas as pd

employee_details = {
        "idno":[101,102,103,104,105],
```

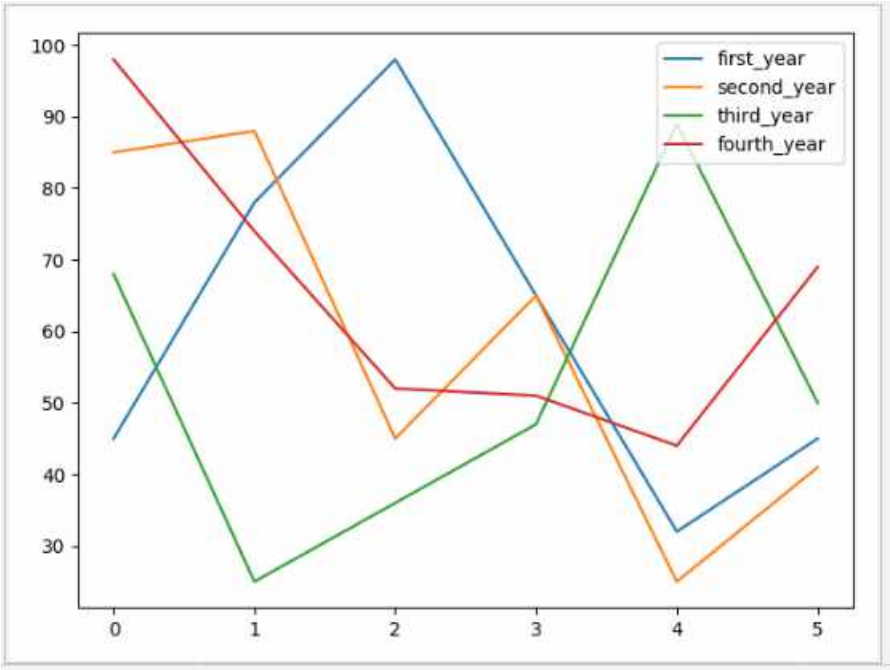
```
        "name":["Ravi","Kumar","Mohan","Krishna","Prasad"],
"salary":[185000.00,285000.00,125000.00,225000.00,100000.00]
                }
df = pd.DataFrame(employee_details)
print(df)
```

Visualization with matplotlib

pip install matplotlib

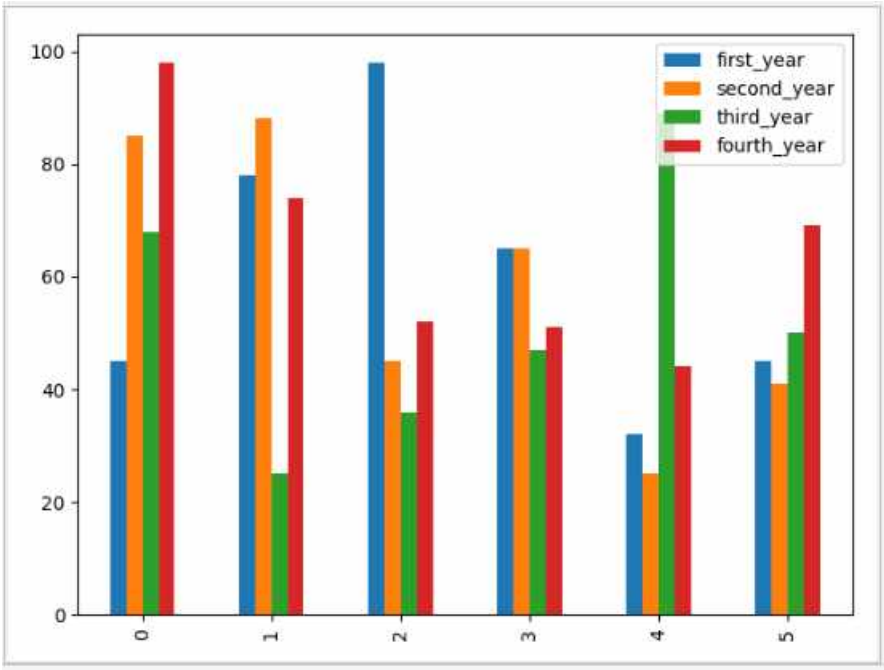
Example 1:

```
import pandas as pd
import matplotlib.pyplot as plt

employee_details = {
        "first_year":[45,78,98,65,32,45],
        "second_year":[85,88,45,65,25,41],
        "third_year":[68,25,36,47,89,50],
        "fourth_year":[98,74,52,51,44,69]
            }
df = pd.DataFrame(employee_details)
df.plot()
plt.show()
```



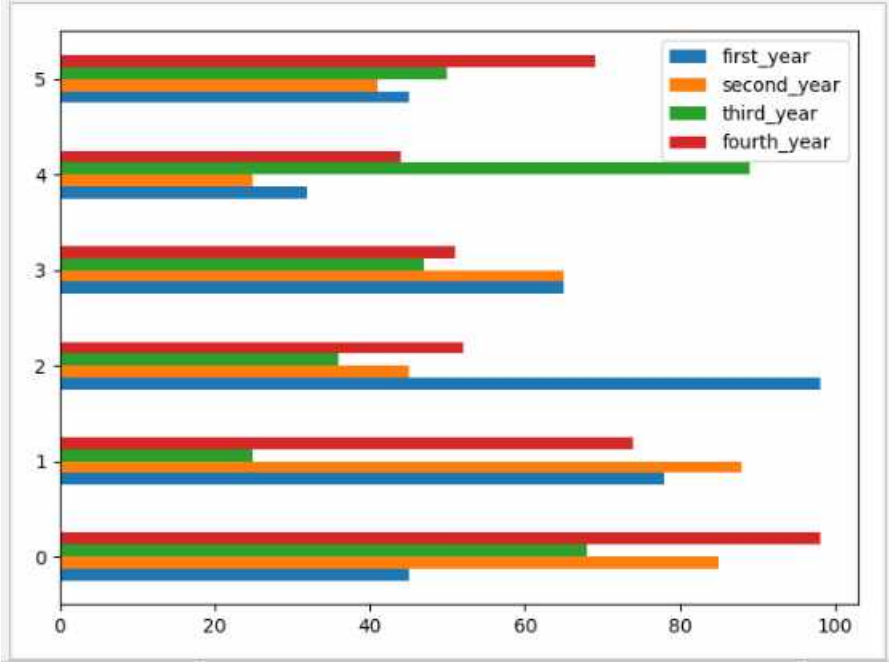
Example 2:

```
import pandas as pd
import matplotlib.pyplot as plt

employee_details = {
        "first_year":[45,78,98,65,32,45],
        "second_year":[85,88,45,65,25,41],
        "third_year":[68,25,36,47,89,50],
        "fourth_year":[98,74,52,51,44,69]
            }

df = pd.DataFrame(employee_details)
df.plot.bar()
plt.show()
```



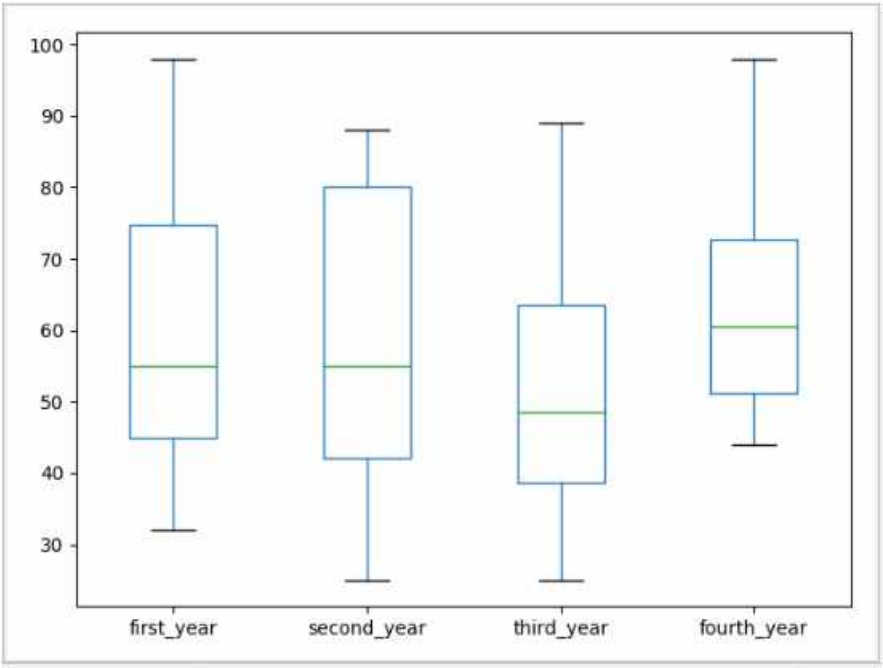
Example 3:

df.plot.barh() # horizontal bar



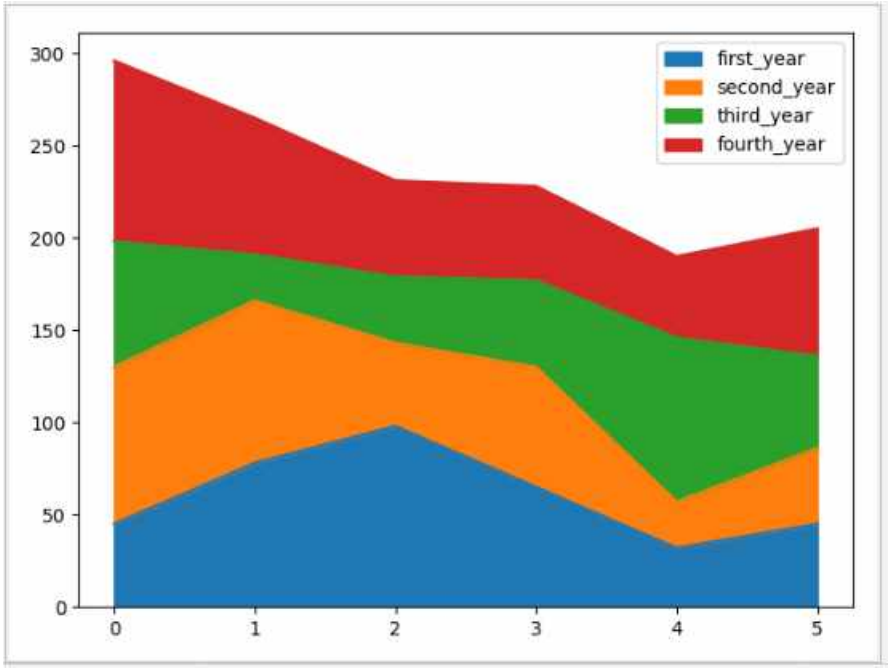
Example 4:

df.plot.box() # Box Plots



Example 5:

df.plot.area() # Area Plot



Example 6 :

df.plot.pie(subplots=**True**) # Pie Chart