Pandas Objects

"pandas" is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. 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 labeled 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. pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Usually, pandas and numpy are imported as follows:

In [ ]:

```
import pandas as pd
import numpy as np
```

In [ ]:

```
print("Pandas version", pd.__version__)
```

In [ ]:

```
print("Numpy versoion ", np.__version__)
```

Series - 1D labeled homogeneously-typed array

DataFrame - General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed column

The best way to think about the pandas data structures is as flexible containers for lower dimensional data. For example, DataFrame is a container for Series, and Series is a container for scalars. We would like to be able to insert and remove objects from these containers in a dictionary-like fashion.

Pandas Series

Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the index. The basic method to create a Series is to call:

In [ ]:

```
#series = pd.Series(data, index=index)
```

Here, data can be many different things:

- a Python dict
- an ndarray
- a scalar value (like 5)
  The passed index is a list of axis labels. Thus, this separates into a few cases depending on what data is:

From Python dict

Series can be instantiated from dicts:

In [ ]:

```
dict = {"key1" : 1, "key2" : 1, "key3": 2}
dict
```

In [ ]:

```
series = pd.Series(dict)

print("Series elements : ")
print(series)

print("\n data type : " , type(series))

print("\n data type of series elements: ", series.dtype)

print("\n number of series elements : ", len(series))
```

From ndarray

If data is an ndarray, index must be the same length as data. If no index is passed, one will be created having values [0, ..., len(data) - 1].

In [ ]:

```
array = np.array([10, 23, 34, 45, 45])
series = pd.Series(array)
series
```

First value in each row is index starting from 0 to number of elements minus one. Second value is actual data value.

In [ ]:

```
array = np.array([12, 23, 45, 56, 56])
index = ["a", "b", "c", "d", "e"]
series = pd.Series(array, index)
series
```

First value in each row is user defined index. Second value is actual data value.

Series index can be obtained using "index" propety of series object.

In [ ]:

```
series.index
```

From scalar value

If data is a scalar value, an index must be provided. The value will be repeated to match the length of index.

In [ ]:

```
series = pd.Series(1., index=['a', 'b', 'c'])
series
```

Here data value '1.0' is repeated thrice (based on the number of index values provided).

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

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

Along with the data, you can optionally pass index (row labels) and columns (column labels) arguments.

From dictionary of Series

The data frame can be created from dictionary object which holds series as its values. Here dictionry contains four series in it like name, age, salary and working with corresponding data values in it.

In [ ]:

```
dict = {
           "name" : pd.Series(["A", "B", "c"]),
           "age": pd.Series([21, 22, 23]),
           "salary": pd.Series([11.5, 23.4, 56]),
           "working" : pd.Series([True, False, True])
       }
dict
```

In [ ]:

```
data_frame = pd.DataFrame(dict)
data_frame
```

Various properties of data frame can be explored as follows :

In [ ]:

```
print("Type of data frame object : ", type(data_frame))
print("Data types of data frame columns : ", data_frame.dtypes)
print("Columns of data frame : ", data_frame.columns)
print("Row labels of data frame : ", data_frame.index)
print("Number of entries in data frame : ", len(data_frame))
print("Number of cells is data frame : " , data_frame.size)
print()
```

A dictionary can be created from series which have different number of data values present in it. For example, first series can have 4 values whereas second one can have only two values in it. Then lets see how a data frame can be created out of such dictionary object.

In [ ]:

```
dict = {
            "one" : pd.Series(['A', 'B', 'C', 'D'], index=["a", "b", "c", "d"]),
            "two" : pd.Series([1, 2, 3], index=["a", "b", "c"]),
            "three " : pd.Series([11, 22], index=["a", "b"])
       }
dict
```

In [ ]:

```
data_frame = pd.DataFrame(dict)
data_frame
```

As you can see, for the series elements where values are less are filled with NaN i.e. Not a number , kind of missing value representation.

From dict of ndarrays / lists

Data frame can be created out of dictionary having ndarrays or lists as its object values.

In [ ]:

```
array = np.random.randint(5, size = 5)
array
```

In [ ]:

```
list1 = [ i*i for i in range(5)]
list1
```

In [ ]:

```
dict = {
            "column1" : array,
            "column2" : list1
       }
dict
```

In [ ]:

```
data_frame = pd.DataFrame(dict)
data_frame
```

From series

If the series containt the data values for a column then following steps needs to be followed to create a data frame for it.

- Create a serieses using the data values for a column with index specified
- Create a list of serieses
- Use pd.concat along horizontal axis to add the series to the data frames
- Rename the data frames columns

In [ ]:

```
#Create an index that will be used along with each series
list_of_indices = ["index1", "index2", "index3", "index4", "index5"]
```

In [ ]:

```
#Create a serieses using the data values for a column with index specified
series1 = pd.Series([1, 2, 3, 4, 5], index = list_of_indices)
series2 = pd.Series([11, 22, 33, 44, 55], index = list_of_indices)
series3 = pd.Series(["A", "B", "C", "D", "E"], index = list_of_indices)
```

In [ ]:

```
#Create a list of serieses
list_of_series = [ series1, series2, series3]
```

In [ ]:

```
#Use pd.concat along horizontal axis to add the series to the data frames
data_frame = pd.concat(list_of_series, axis=1)
data_frame
```

In [ ]:

```
#Rename the data frames columns
list_of_columns = ["column1", "column2", "column3"]
data_frame.columns = list_of_columns
data_frame
```

If series is row then following steps needs to be followed to create a data frame for it.

- Create a serieses using the data values for a column with index specified
- Create a list of serieses
- Use pd.DataFrame the series list to the data frames
- Rename the data frames columns

In [ ]:

```
#Create an index that will be used along with each series
list_of_indices = ["column1", "column2", "column3", "column4", "column5"]
```

In [ ]:

```
#Create a serieses using the data values for a column with index specified
series1 = pd.Series([1, 2, "A", True, 5.0], index = list_of_indices)
series2 = pd.Series([11, 22, "B", False, 55.0], index = list_of_indices)
series3 = pd.Series([21, 34, "C", True, 12], index = list_of_indices)
```

In [ ]:

```
#Create a list of serieses
list_of_series = [ series1, series2, series3]
```

In [ ]:

```
#Use pd.DataFrame along horizontal axis to add the series to the data frames
data_frame = pd.DataFrame(list_of_series)
data_frame
```

Exercise

Q1. Create a DataFrame by passing a NumPy array, with a datetime index and labeled columns.

In [ ]:

```
#Try it here
```