Python Pandas interview questions

A list of top frequently asked **Python Pandas Interview Questions and answers** are given below.

1) Define the Pandas/Python pandas?

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It can be used for data analysis in Python and developed by Wes McKinney in 2008. It can perform five significant steps that are required for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyze.

---

2) Mention the different types of Data Structures in Pandas?

Pandas provide two data structures, which are supported by the pandas library, **Series,** and **DataFrames.** Both of these data structures are built on top of the NumPy.

A **Series** is a one-dimensional data structure in pandas, whereas the **DataFrame** is the two-dimensional data structure in pandas.

---

3) Define Series in Pandas?

A Series is defined as a one-dimensional array that is capable of storing various data types. The row labels of series are called the **index**. By using a '**series**' method, we can easily convert the list, tuple, and dictionary into series. A Series cannot contain multiple columns.

---

4) How can we calculate the standard deviation from the Series?

The Pandas **std()** is defined as a function for calculating the standard deviation of the given set of numbers, DataFrame, column, and rows.

Series.std(axis=None, skipna=None, level=None, ddof=1, numeric_only=None, **kwargs)

---

5) Define DataFrame in Pandas?

A DataFrame is a widely used data structure of pandas and works with a two-dimensional array with labeled axes (rows and columns) DataFrame is defined as a standard way to store data and has two different indexes, i.e., row index and column index. It consists of the following properties:

The columns can be heterogeneous types like int and bool.

It can be seen as a dictionary of Series structure where both the rows and columns are indexed. It is denoted as "columns" in the case of columns and "index" in case of rows.

---

6) What are the significant features of the pandas Library?

The key features of the panda's library are as follows:

Memory Efficient

Data Alignment

Reshaping

Merge and join

Time Series

---

7) Explain Reindexing in pandas?

Reindexing is used to conform DataFrame to a new index with optional filling logic. It places NA/NaN in that location where the values are not present in the previous index. It returns a new object unless the new index is produced as equivalent to the current one, and the value of copy becomes False. It is used to change the index of the rows and columns of the DataFrame.

---

8) What is the name of Pandas library tools used to create a scatter plot matrix?

Scatter_matrix

---

9) Define the different ways a DataFrame can be created in pandas?

We can create a DataFrame using following ways:

Lists

Dict of ndarrays

Example-1: Create a DataFrame using List:

```
import pandas as pd

# a list of strings

a = ['Python', 'Pandas']

# Calling DataFrame constructor on list

info = pd.DataFrame(a)

print(info)
```

Output:

```
        0
0  Python
1  Pandas
```

Example-2: Create a DataFrame from dict of ndarrays:

```
import pandas as pd

info = {'ID' :[101, 102, 103],'Department' :['B.Sc','B.Tech','M.Tech',]}

info = pd.DataFrame(info)

print (info)
```

Output:

```
   ID  Department
0  101       B.Sc
1  102      B.Tech
2  103      M.Tech
```

---

10) Explain Categorical data in Pandas?

A Categorical data is defined as a Pandas data type that corresponds to a categorical variable in statistics. A categorical variable is generally used to take a limited and usually fixed number of possible values. Examples: gender, country affiliation, blood type, social class, observation time, or rating via Likert scales. All values of categorical data are either in categories or np.nan.

This data type is useful in the following cases:

It is useful for a string variable that consists of only a few different values. If we want to save some memory, we can convert a string variable to a categorical variable.

It is useful for the lexical order of a variable that is not the same as the logical order (?one?, ?two?, ?three?) By converting into a categorical and specify an order on the categories, sorting and min/max is responsible for using the logical order instead of the lexical order.

It is useful as a signal to other Python libraries because this column should be treated as a categorical variable.

---

11) How will you create a series from dict in Pandas?

A Series is defined as a one-dimensional array that is capable of storing various data types.

We can create a Pandas Series from Dictionary:

Create a Series from dict:

We can also create a Series from dict. If the dictionary object is being passed as an input and the index is not specified, then the dictionary keys are taken in a sorted order to construct the index.

If index is passed, then values correspond to a particular label in the index will be extracted from the dictionary.

```
import pandas as pd
```

```
import numpy as np

info = {'x' : 0., 'y' : 1., 'z' : 2.}

a = pd.Series(info)

print (a)
```

Output:

```
x    0.0
y    1.0
z    2.0
dtype: float64
```

---

12) How can we create a copy of the series in Pandas?

We can create the copy of series by using the following syntax:

pandas.Series.copy

Series.copy(deep=True)

The above statements make a deep copy that includes a copy of the data and the indices. If we set the value of deep to **False**, it will neither copy the indices nor the data.

> Note: If we set **deep=True**, the data will be copied, and the actual python objects will not be copied recursively, only the reference to the object will be copied.

---

13) How will you create an empty DataFrame in Pandas?

A DataFrame is a widely used data structure of pandas and works with a two-dimensional array with labeled axes (rows and columns) It is defined as a standard way to store data and has two different indexes, i.e., row index and column index.

Create an empty DataFrame:

The below code shows how to create an empty DataFrame in Pandas:

```
# importing the pandas library

import pandas as pd
```

```
info = pd.DataFrame()

print (info)
```

Output:

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

---

14) How will you add a column to a pandas DataFrame?

We can add any new column to an existing DataFrame. The below code demonstrates how to add any new column to an existing DataFrame:

```
# importing the pandas library

import pandas as pd

info = {'one' : pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e']),

        'two' : pd.Series([1, 2, 3, 4, 5, 6], index=['a', 'b', 'c', 'd', 'e', 'f'])}


info = pd.DataFrame(info)


# Add a new column to an existing DataFrame object


print ("Add new column by passing series")

info['three']=pd.Series([20,40,60],index=['a','b','c'])

print (info)

print ("Add new column using existing DataFrame columns")

info['four']=info['one']+info['three']

print (info)
```

Output:

```
Add new column by passing series
   one  two  three
a  1.0    1   20.0
b  2.0    2   40.0
c  3.0    3   60.0
d  4.0    4    NaN
e  5.0    5    NaN
f  NaN    6    NaN

Add new column using existing DataFrame columns
   one  two  three  four
a  1.0    1   20.0  21.0
b  2.0    2   40.0  42.0
c  3.0    3   60.0  63.0
d  4.0    4    NaN   NaN
e  5.0    5    NaN   NaN
f  NaN    6    NaN   NaN
```

---

15) How to add an Index, row, or column to a Pandas DataFrame?

Adding an Index to a DataFrame

Pandas allow adding the inputs to the **index** argument if you create a DataFrame. It will make sure that you have the desired index. If you don?t specify inputs, the DataFrame contains, by default, a numerically valued index that starts with 0 and ends on the last row of the DataFrame.

Adding Rows to a DataFrame

We can use **.loc, iloc,** and **ix** to insert the rows in the DataFrame.

The **loc** basically works for the labels of our index. It can be understood as if we insert in loc[4], which means we are looking for that values of DataFrame that have an index labeled 4.

The **iloc** basically works for the positions in the index. It can be understood as if we insert in iloc[4], which means we are looking for the values of DataFrame that are present at index '4`.

The **ix** is a complex case because if the index is integer-based, we pass a label to **ix**. The **ix[4]** means that we are looking in the DataFrame for those values that have an index labeled 4. However, if the index is not only integer-based, ix will deal with the positions as **iloc**.

Adding Columns to a DataFrame

If we want to add the column to the DataFrame, we can easily follow the same procedure as adding an index to the DataFrame by using loc or iloc.

---

16) How to Delete Indices, Rows or Columns From a Pandas Data Frame?

Deleting an Index from Your DataFrame

If you want to remove the index from the DataFrame, you should have to do the following:

Reset the index of DataFrame.

Executing **del df.index.name** to remove the index name.

Remove duplicate index values by resetting the index and drop the duplicate values from the index column.

Remove an index with a row.

Deleting a Column from Your DataFrame

You can use the **drop()** method for deleting a column from the DataFrame.

The axis argument that is passed to the **drop()** method is either 0 if it indicates the rows and 1 if it drops the columns.

You can pass the argument **inplace** and set it to True to delete the column without reassign the DataFrame.

You can also delete the duplicate values from the column by using the drop_duplicates() method.

Removing a Row from Your DataFrame

By using **df.drop_duplicates(),** we can remove duplicate rows from the DataFrame.

You can use the **drop()** method to specify the index of the rows that we want to remove from the DataFrame.

---

17) How to Rename the Index or Columns of a Pandas DataFrame?

You can use the **.rename** method to give different values to the columns or the index values of DataFrame.

---

18) How to iterate over a Pandas DataFrame?

You can iterate over the rows of the DataFrame by using for loop in combination with an iterrows() call on the DataFrame.

---

19) How to get the items of series A not present in series B?

We can remove items present in **p2** from **p1** using isin() method.

```
import pandas as pd

p1 = pd.Series([2, 4, 6, 8, 10])

p2 = pd.Series([8, 10, 12, 14, 16])

p1[~p1.isin(p2)]
```

Solution

```
0    2
1    4
2    6
dtype: int64
```

---

20) How to get the items not common to both series A and series B?

We get all the items of p1 and p2 not common to both using below example:

```
import pandas as pd

import numpy as np

p1 = pd.Series([2, 4, 6, 8, 10])

p2 = pd.Series([8, 10, 12, 14, 16])

p1[~p1.isin(p2)]

p_u = pd.Series(np.union1d(p1, p2))  # union

p_i = pd.Series(np.intersect1d(p1, p2))  # intersect

p_u[~p_u.isin(p_i)]
```

Output:

```
0    2
1    4
2    6
5   12
6   14
7   16
dtype: int64
```

---

21) How to get the minimum, 25th percentile, median, 75th, and max of a numeric series?

We can compute the minimum, 25th percentile, median, 75th, and maximum of **p** as below example:

```
import pandas as pd

import numpy as np

p = pd.Series(np.random.normal(14, 6, 22))

state = np.random.RandomState(120)

p = pd.Series(state.normal(14, 6, 22))

np.percentile(p, q=[0, 25, 50, 75, 100])
```

Output:

```
array([ 4.61498692, 12.15572753, 14.67780756, 17.58054104, 33.24975515])
```

---

22) How to get frequency counts of unique items of a series?

We can calculate the frequency counts of each unique value p as below example:

```
import pandas as pd

import numpy as np

p= pd.Series(np.take(list('pqrstu'), np.random.randint(6, size=17)))

p = pd.Series(np.take(list('pqrstu'), np.random.randint(6, size=17)))

p.value_counts()
```

Output:

```
s  4
r  4
q  3
p  3
u  3
```

---

23) How to convert a numpy array to a dataframe of given shape?

We can reshape the series **p** into a dataframe with 6 rows and 2 columns as below example:

```
import pandas as pd

import numpy as np

p = pd.Series(np.random.randint(1, 7, 35))

# Input

p = pd.Series(np.random.randint(1, 7, 35))

info = pd.DataFrame(p.values.reshape(7,5))

print(info)
```

Output:

```
0 1 2 3 4
0 3 2 5 5 1
1 3 2 5 5 5
2 1 3 1 2 6
3 1 1 1 2 2
4 3 5 3 3 3
5 2 5 3 6 4
6 3 6 6 6 5
```

---

24) How can we convert a Series to DataFrame?

The Pandas **Series.to_frame()** function is used to convert the series object to the DataFrame.

```
Series.to_frame(name=None)
```

name: Refers to the object. Its Default value is None. If it has one value, the passed name will be substituted for the series name.

```
s = pd.Series(["a", "b", "c"],

name="vals")

s.to_frame()
```

Output:

```
   vals
0     a
1     b
2     c
```

---

25) What is Pandas NumPy array?

Numerical Python (Numpy) is defined as a Python package used for performing the various numerical computations and processing of the multidimensional and single-dimensional array elements. The calculations using Numpy arrays are faster than the normal Python array.

---

26) How can we convert DataFrame into a NumPy array?

For performing some high-level mathematical functions, we can convert Pandas DataFrame to numpy arrays. It uses the **DataFrame.to_numpy()** function.

The **DataFrame.to_numpy()** function is applied to the DataFrame that returns the numpy ndarray.

```
DataFrame.to_numpy(dtype=None, copy=False)
```

---

27) How can we convert DataFrame into an excel file?

We can export the DataFrame to the excel file by using the **to_excel()** function.

To write a single object to the excel file, we have to specify the target file name. If we want to write to multiple sheets, we need to create an **ExcelWriter** object with target filename and also need to specify the sheet in the file in which we have to write.

---

28) How can we sort the DataFrame?

We can efficiently perform sorting in the DataFrame through different kinds:

By label

By Actual value

By label

The DataFrame can be sorted by using the sort_index() method. It can be done by passing the axis arguments and the order of sorting. The sorting is done on row labels in ascending order by default.

By Actual Value

It is another kind through which sorting can be performed in the DataFrame. Like index sorting, **sort_values()** is a method for sorting the values.

It also provides a feature in which we can specify the column name of the DataFrame with which values are to be sorted. It is done by passing the '**by**' argument.

---

29) What is Time Series in Pandas?

The Time series data is defined as an essential source for information that provides a strategy that is used in various businesses. From a conventional finance industry to the education industry, it consists of a lot of details about the time.

Time series forecasting is the machine learning modeling that deals with the Time Series data for predicting future values through Time Series modeling.

---

30) What is Time Offset?

The offset specifies a set of dates that conform to the DateOffset. We can create the DateOffsets to move the dates forward to valid dates.

---

31) Define Time Periods?

The Time Periods represent the time span, e.g., days, years, quarter or month, etc. It is defined as a class that allows us to convert the frequency to the periods.

---

32) How to convert String to date?

The below code demonstrates how to convert the string to date:

```
fromdatetime import datetime


# Define dates as the strings

dmy_str1 = 'Wednesday, July 14, 2018'

dmy_str2 = '14/7/17'

dmy_str3 = '14-07-2017'


# Define dates as the datetime objects

dmy_dt1 = datetime.strptime(date_str1, '%A, %B %d, %Y')

dmy_dt2 = datetime.strptime(date_str2, '%m/%d/%y')

dmy_dt3 = datetime.strptime(date_str3, '%m-%d-%Y')


#Print the converted dates

print(dmy_dt1)

print(dmy_dt2)

print(dmy_dt3)
```

Output:

```
2017-07-14 00:00:00
2017-07-14 00:00:00
2018-07-14 00:00:00
```

---

33) What is Data Aggregation?

The main task of Data Aggregation is to apply some aggregation to one or more columns. It uses the following:

sum: It is used to return the sum of the values for the requested axis.

min: It is used to return a minimum of the values for the requested axis.

max: It is used to return a maximum values for the requested axis.

---

34) What is Pandas Index?

Pandas Index is defined as a vital tool that selects particular rows and columns of data from a DataFrame. Its task is to organize the data and to provide fast accessing of data. It can also be called a **Subset Selection**.

---

35) Define Multiple Indexing?

Multiple indexing is defined as essential indexing because it deals with data analysis and manipulation, especially for working with higher dimensional data. It also enables us to store and manipulate data with the arbitrary number of dimensions in lower-dimensional data structures like Series and DataFrame.

---

36) Define ReIndexing?

Reindexing is used to change the index of the rows and columns of the DataFrame. We can reindex the single or multiple rows by using the reindex() method. Default values in the new index are assigned NaN if it is not present in the DataFrame.

DataFrame.reindex(labels=None, index=None, columns=None, axis=None, method=None, copy=True, level=None, fill_value=nan, limit=None, tolerance=None)

---

37) How to Set the index?

We can set the index column while making a data frame. But sometimes, a data frame is made from two or more data frames, and then the index can be changed using this method.

---

38) How to Reset the index?

The Reset index of the DataFrame is used to reset the index by using the '**reset_index**' command. If the DataFrame has a MultiIndex, this method can remove one or more levels.

---

39) Describe Data Operations in Pandas?

In Pandas, there are different useful data operations for DataFrame, which are as follows:

Row and column selection

We can select any row and column of the DataFrame by passing the name of the rows and columns. When you select it from the DataFrame, it becomes one-dimensional and considered as Series.

Filter Data

We can filter the data by providing some of the boolean expressions in DataFrame.

Null values

A Null value occurs when no data is provided to the items. The various columns may contain no values, which are usually represented as NaN.

---

40) Define GroupBy in Pandas?

In Pandas, **groupby()** function allows us to rearrange the data by utilizing them on real-world data sets. Its primary task is to split the data into various groups. These groups are categorized based on some criteria. The objects can be divided from any of their axes.

DataFrame.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, **kwargs)
