```
1.  Define the Pandas/Python pandas?

-Pandas is a free, open-source software library, built on top of NumPy, for data manipulation and analysis in Python. It offers data structures and operations for working with numerical tables and time series.

-Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
```

```
2.  What are the different types of Data Structures in Pandas?

-There are three main data structures in pandas:
1.Series: A one-dimensional data structure that similar to a single column of data.
          -A series consists of two arrays: an array of values and an
               array of labels.

2.DataFrame: A two-dimensional data structure that similar to a spreadsheet or SQL table. A DataFrame has rows and columns, with each column being a pandas Series.

3.Panel: A three-dimensional array that generally used for 3D time-series data
```

```
3.  Explain Series and DataFrame In Pandas

-In Pandas, Series and DataFrame are two fundamental data structures used for handling and manipulating data efficiently.

-They are part of the Pandas library, which is widely used for data analysis and manipulation in Python.

1.Series:
   - A Series is essentially a one-dimensional labeled array capable of holding any data type (integers, strings, floats, Python objects, etc.).
   - It's like a column in a spreadsheet or a single column in a database table.
   - Each element in a Series has a corresponding label, referred to as its index.

- Syntax to create a Series:

    import pandas as pd
    s = pd.Series(data, index=index)

- Example:

    import pandas as pd
    data = [1, 2, 3, 4, 5]
    s = pd.Series(data)

    Output:

    0    1
```

```
    1    2
    2    3
    3    4
    4    5
    dtype: int64


2.DataFrame:-
   - A DataFrame is a two-dimensional labeled data structure with columns of potentially different types.
   - It's like a spreadsheet or a SQL table where each column can be of a different data type.
   - A DataFrame can be thought of as a collection of Series that share the same index.

- Syntax to create a DataFrame:

    import pandas as pd
    df = pd.DataFrame(data, index=index, columns=columns)

- Example:

    import pandas as pd
    data = {'Name': ['Alice', 'Bob', 'Charlie'],
            'Age': [25, 30, 35],
            'City': ['New York', 'Los Angeles', 'Chicago']}
    df = pd.DataFrame(data)

    Output:
       Name  Age         City
    0  Alice   25     New York
    1    Bob   30  Los Angeles
    2 Charlie   35      Chicago
    ```
   - DataFrame provides powerful methods and functions for data manipulation, exploration, cleaning, and analysis, such as merging, joining, grouping, and aggregation.
```

```
4. How Can You Create An Empty DataFrame and series in Pandas?
-In Pandas, you can create an empty DataFrame or Series using specific constructors provided by the library.

1.Empty DataFrame:
   - To create an empty DataFrame, you can use the pd.DataFrame() constructor without passing any data.
   - You can then add data to this DataFrame later.

- Example:

    import pandas as pd
    df = pd.DataFrame()

2.Empty Series:-
   - To create an empty Series, you can use the `pd.Series()` constructor without passing any data.
   - You can then append elements to this Series as needed.

- Example:

    import pandas as pd
```
```

```
    s = pd.Series()
```

5. How to check an empty DataFrame?

In [1]:

```
import pandas as pd
# Create an empty DataFrame
df = pd.DataFrame()
# Check if the DataFrame is empty
if df.empty:
    print("DataFrame is empty")
else:
    print("DataFrame is not empty")

```

```
C:\Users\kimay\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:60: UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (

DataFrame is empty
```

6. What Are The Most Important Features Of The Pandas Library?

```
Important Features Of The Pandas Library:-

1.DataFrame: The DataFrame is the primary data structure in Pandas. It allows for handling two-dimensional labeled data, SQL table, with rows and columns.
-This enables easy data manipulation, cleaning, and analysis.

2.Series: Series is another fundamental data structure in Pandas. It represents a one-dimensional labeled array capable of holding any data type. It's like a single column of data in a DataFrame.

3.Data Input/Output: Pandas provides functionality to read from and write to various file formats, including CSV, Excel, SQL databases, JSON, HTML, and more. This makes it easy to import data from external sources and export processed data.

4.Data Cleaning: Pandas offers robust tools for data cleaning, including handling missing values (NaN), data transformation, filtering, and handling duplicates. Methods like dropna(), fillna(), and drop_duplicates() are commonly used for data cleaning tasks.

5.Indexing and Selection: Pandas provides powerful indexing and selection capabilities, allowing for easy extraction, filtering, and manipulation of data. This includes label-based indexing using column names and positional indexing.

6.Grouping and Aggregation: Pandas supports grouping data based on one or more keys, enabling split-apply-combine operations. This allows for aggregating, summarizing, and analyzing data at different levels of granularity.

```

```
7.Merge and Join: Pandas offers functionality to merge and join DataFrame objects based on common columns or indices. This is useful for combining data from different sources or performing relational database-like operations.

8.Visualization: Pandas integrates with other libraries like Matplotlib and Seaborn for data visualization. It provides convenient methods for creating plots directly from DataFrame and Series objects.

9.High Performance: Pandas is optimized for performance, particularly for handling large datasets. It leverages efficient data structures and algorithms to ensure fast data processing.
```

7. How Will You Explain Reindexing In Pandas?

```
-Reindexing in Pandas refers to the process of changing the index of a DataFrame or a Series.
-It involves creating a new object with the data conformed to a new index.
-Reindexing is useful for aligning data to new indices, handling missing values, and reshaping data.

-example of reindexing in Pandas:

import pandas as pd

data = {'A': [1, 2, 3], 'B': [4, 5, 6]}
df = pd.DataFrame(data, index=['a', 'b', 'c'])

new_index = ['a', 'b', 'c', 'd']
df_reindexed = df.reindex(new_index)

print(df_reindexed)

-Output:
     A    B
a  1.0  4.0
b  2.0  5.0
c  3.0  6.0
d  NaN  NaN
```

8. What are the different ways of creating DataFrame in pandas? Explain with examples.

In [2]:
```
#From a Dictionary of Lists or Arrays:
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'City': ['New York', 'Los Angeles', 'Chicago']}

df = pd.DataFrame(data)
print(df)

```

```
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
```

In [3]:
```
#From a NumPy Array:
import pandas as pd
import numpy as np

data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
df = pd.DataFrame(data, columns=['A', 'B', 'C'])
print(df)
```

```
   A  B  C
0  1  2  3
1  4  5  6
2  7  8  9
```

In [4]:
```
#From a Dictionary of Series:
import pandas as pd

data = {'Name': pd.Series(['Alice', 'Bob', 'Charlie']),
        'Age': pd.Series([25, 30, 35]),
        'City': pd.Series(['New York', 'Los Angeles', 'Chicago'])}

df = pd.DataFrame(data)
print(df)
```

```
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
```

9. Create a DataFrame using List.

In [5]:
```
import pandas as pd
data = [
    ['Alice', 25, 'New York'],
    ['Bob', 30, 'Los Angeles'],
    ['Charlie', 35, 'Chicago']
]
columns = ['Name', 'Age', 'City']
df = pd.DataFrame(data, columns=columns)
print(df)

```

```
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
```

10. Create a DataFrame using Numpy Functions.

In [6]:
```
import pandas as pd
import numpy as np
data_array = np.random.randn(5, 3)
df = pd.DataFrame(data_array, columns=['A', 'B', 'C'])
print(df)

```

```
          A         B         C
0 -0.993902 -1.256191  1.127076
1 -0.022590  0.202611 -2.730451
2  1.072568 -1.738293 -0.449500
3 -0.993815  1.277863 -0.189149
4  0.151810 -1.170128 -0.402904
```

11. How to convert a NumPy array to a DataFrame of a given shape?

In [7]:
```
import pandas as pd
import numpy as np

desired_shape = (3, 4)
array = np.arange(np.prod(desired_shape)).reshape(desired_shape)
df = pd.DataFrame(array)
print(df)

```

```
   0  1   2   3
0  0  1   2   3
1  4  5   6   7
2  8  9  10  11
```

12. Create a DataFrame using Dictionary with a list and arrays

In [8]:
```
import pandas as pd
import numpy as np
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': np.array([25, 30, 35]),
    'City': ['New York', 'Los Angeles', 'Chicago']}
df = pd.DataFrame(data)
print(df)

```

```
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
```

13. How To Create A Copy Of The Series and DataFrame in Pandas?

In [9]:
```
#series
import pandas as pd
s = pd.Series([1, 2, 3, 4, 5])
s_copy = s.copy()
s_copy[0] = 10
print(s)
print(s_copy)

```

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
0    10
1     2
2     3
3     4
4     5
dtype: int64
```

14 .How Will You Add An Index, Row, Or Column To A DataFrame In Pandas?

In [10]:
```
#dataframe
import pandas as pd
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'City': ['New York', 'Los Angeles', 'Chicago']}
df = pd.DataFrame(data)
df_copy = df.copy()
df_copy['Age'] = [28, 33, 38]
print(df)
print(df_copy)

```

```
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
      Name  Age         City
0    Alice   28     New York
1      Bob   33  Los Angeles
2  Charlie   38      Chicago
```

14. How Will You Add An Index, Row, Or Column To A DataFrame In Pandas?

15. What Method Will You Use To Rename The Index Or Columns Of Pandas DataFrame?

In [11]:
```
import pandas as pd

df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]}, index=['X', 'Y', 'Z

# Rename index labels
df.rename(index={'X': 'New_X', 'Y': 'New_Y', 'Z': 'New_Z'}, inplace=Tru

```

In [13]:
```
`import pandas as pd

# Create a DataFrame
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})

# Rename column names
df.rename(columns={'A': 'New_A', 'B': 'New_B'}, inplace=True)

```

```
  Cell In[13], line 1
    `import pandas as pd
    ^
SyntaxError: invalid syntax
```

15. What Method Will You Use To Rename The Index Or Columns Of Pandas DataFrame?

In [ ]:
```
1
```