Untitled

March 1, 2023

Q1. Create a Pandas Series that contains the following data: 4, 8, 15, 16, 23, and 42. Then, print the series.

[1]:
```
import pandas as pd

# Create a Pandas Series
series = pd.Series([4, 8, 15, 16, 23, 42])

# Print the series
print(series)
```

```
0     4
1     8
2    15
3    16
4    23
5    42
dtype: int64
```

Q2. Create a variable of list type containing 10 elements in it, and apply pandas.Series function on the varaiable print it

[2]:
```
import pandas as pd

my_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

my_series = pd.Series(my_list)

print(my_series)
```

```
0     10
1     20
2     30
3     40
4     50
5     60
6     70
7     80
8     90
```

```
9    100
dtype: int64
```

q.3 create a pandas dataframes that caontains the following data Name Alice Bob Claire

Age 25 30 27

Gender Female Male Female

[3]:
```
import pandas as pd

# Create the data as a dictionary
data = {
    'Name': ['Alice', 'Bob', 'Claire'],
    'Age': [25, 30, 27],
    'Gender': ['Female', 'Male', 'Female']
}

# Create a DataFrame from the dictionary
df = pd.DataFrame(data)

# Print the DataFrame
print(df)
```

```
     Name  Age  Gender
0   Alice   25  Female
1     Bob   30    Male
2  Claire   27  Female
```

Q4. What is 'DataFrame' in pandas and how is it different from pandas.series? Explain with an example.

in Pandas, a DataFrame is a two-dimensional labeled data structure with columns of potentially different types. It is essentially a table with rows and columns, where each column can be thought of as a Pandas Series. The difference between a Pandas Series and a Pandas DataFrame is that a Series represents a single column of data, while a DataFrame is a table of data with multiple columns.

[4]:
```
import pandas as pd

# Create a Series
s = pd.Series([1, 2, 3, 4, 5])

# Create a DataFrame
df = pd.DataFrame({'A': [1, 2, 3, 4, 5], 'B': [10, 20, 30, 40, 50]})

# Print the Series and DataFrame
print('Series:')
print(s)

print('\nDataFrame:')
```

```
print(df)
```

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

DataFrame:
   A   B
0  1  10
1  2  20
2  3  30
3  4  40
4  5  50
```

Q.4 What are some common functions you can use to manipulate data in a Pandas DataFrame? Can you give an example of when you might use one of these functions?

Pandas DataFrame is a powerful tool for data manipulation and analysis. Here are some common functions that you can use to manipulate data in a Pandas DataFrame:

head() and tail(): These functions allow you to display the first or last few rows of your DataFrame. For example, if you have a large dataset and want to get a quick view of what your data looks like, you can use head() function to display the first few rows.

describe(): This function provides statistical information about the data in your DataFrame. It returns the count, mean, standard deviation, minimum, maximum, and quartiles for each numeric column.

info(): This function provides information about the data types, null values, and memory usage of each column in the DataFrame.

drop(): This function allows you to drop one or more columns or rows from your DataFrame. For example, if you have a column that is not relevant to your analysis, you can drop it using the drop() function.

groupby(): This function groups the data in your DataFrame based on one or more columns, and then allows you to perform operations on the groups. For example, if you have a sales dataset and want to group the data by product type, you can use groupby() function to group the data by product type and then calculate the total sales for each product.

pivot_table(): This function allows you to summarize your data by creating a pivot table. For example, if you have a sales dataset and want to summarize the sales by product type and region, you can use pivot_table() function to create a pivot table that shows the total sales for each product type and region.

apply(): This function applies a function to each element in a column or row of your DataFrame. For example, if you have a column that contains strings and want to convert them to uppercase, you can use apply() function to apply the str.upper() function to each element in the column.

merge(): This function allows you to merge two or more DataFrames based on a common column or index. For example, if you have a sales dataset and a customer dataset, you can use merge() function to merge the two datasets based on the common customer ID column.

sort_values(): This function sorts the data in your DataFrame based on one or more columns. For example, if you have a sales dataset and want to sort the data by product type and then by sales in descending order, you can use sort_values() function to sort the data based on the product type and then the sales column in descending order.

fillna(): This function allows you to fill null or missing values in your DataFrame with a specified value or method. For example, if you have a column that contains null values and want to replace them with the average value of the column, you can use fillna() function to fill the null values with the mean value of the column.

These are just some of the common functions that you can use to manipulate data in a Pandas DataFrame. The specific function that you would use depends on the specific problem you are trying to solve or the analysis you are trying to perform.

Which of the following is mutable in nature Series, DataFrame, Panel?

in Pandas, Series and DataFrame are mutable in nature, whereas Panel is immutable.

A mutable object is an object whose value or state can be modified after it is created. For example, you can change the values of a Series or DataFrame after they are created using methods like loc or iloc. On the other hand, an immutable object is an object whose value or state cannot be changed after it is created. Panel was an important data structure in earlier versions of Pandas, but it is now deprecated and replaced by MultiIndex dataframes, which are also mutable.

Create a DataFrame using multiple Series. Explain with an example.

to create a DataFrame using multiple Series, we can use the pd.DataFrame() function in Pandas, where we pass a dictionary containing the Series as key-value pairs, with each key representing a column name and each value representing the Series.

Here's an example:

[5]:
```
import pandas as pd

# Create three Series with sample data
names = pd.Series(['John', 'Mike', 'Sara', 'Jessie'])
ages = pd.Series([25, 30, 28, 32])
genders = pd.Series(['M', 'M', 'F', 'F'])

# Create a DataFrame using the above Series
df = pd.DataFrame({'Name': names, 'Age': ages, 'Gender': genders})

# Print the resulting DataFrame
print(df)
```

```
   Name  Age Gender
0  John   25      M
1  Mike   30      M
```

```
2    Sara   28      F
3  Jessie   32      F
```

[ ]: