Pandas

A pandas Series is a one-dimensional labeled array that can hold data of any type (integer, float, string, etc.).To create a pandas Series, you can pass in a list or array of data to the pd.Series() function. For example:

```
import pandas as pd
data = [10, 20, 30, 40, 50]
s = pd.Series(data)
print(s)
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

print(s[2]) # output: 30

In this example, the Series is created from a list of integers. The resulting Series has an index of integers from 0 to 4 and contains the values from the original list.

You can also create a Series with a **custom index** by passing in a list of labels to the index parameter or by using **dictionary** :

```
import pandas as pd
data = {'a': 10, 'b': 20, 'c': 30, 'd': 40, 'e': 50}
s = pd.Series(data)
print(s)
```

```
a    10
b    20
c    30
d    40
e    50
dtype: int64
```

```
import pandas as pd
data = [10, 20, 30, 40, 50]
labels = ['a', 'b', 'c', 'd', 'e']
s = pd.Series(data, index=labels)
print(s)
```

print(s['b']) # output: 20

You can access elements of a Series using **the index label or the integer position in examples of printing .**You can also perform various operations on a Series

s_filtered = s[s > 30]

print(s_filtered)

s_sliced = s['b':'d']

```
d    40
e    50
dtype: int64
b    20
c    30
d    40
dtype: int64
```

```
data = [10, 20, 30, 40, 50]
labels = ['a', 'b', 'c', 'd', 'e']
s = pd.Series(data, index=labels)
print(20 in s)  # output: True
print('f' in s)  # output: False
```

```
s = pd.Series([1, 2, 3, 4, 5])
s_squared = s.apply(lambda x: x**2)
print(s_squared)
```

```
0     1
1     4
2     9
3    16
4    25
dtype: int64
```

```
data = [10, 20, 30, 40, 50]
s = pd.Series(data)
print(s.shape)
print(s.ndim)
print(s.size)
print(s.values)
print(s.index)
```

```
(5,)
1
5
[10 20 30 40 50]
RangeIndex(start=0, stop=5, step=1)
```

You can create a DataFrame : 1.From a list of dictionaries 2. From a dictionary of lists

```python
import pandas as pd
data = [{'name': 'Alice', 'age': 25},
        {'name': 'Bob', 'age': 30},
        {'name': 'Charlie', 'age': 35}]
df = pd.DataFrame(data)
print(df)
```

```
      name  age
0    Alice   25
1      Bob   30
2  Charlie   35
```

```python
import pandas as pd
data = {'name': ['Alice', 'Bob', 'Charlie'],
        'age': [25, 30, 35]}
df = pd.DataFrame(data)
print(df)
```

```
      name  age
0    Alice   25
1      Bob   30
2  Charlie   35
```

```python
import pandas as pd
data = {'name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily'],
        'age': [25, 30, 35, 40, 45],
        'gender': ['F', 'M', 'M', 'M', 'F']}
df = pd.DataFrame(data)
print(df)
print(df.loc[0])
print(df.loc[[0, 2, 4]])
print(df.loc[1:3])
print("--------------------")
print(df.loc[:, 'name'])
print(df.loc[:, ['name', 'age']])
print(df.loc[1:3, ['name', 'age']])
print("--------------------")
print(df.iloc[0])
print(df.iloc[[0, 2, 4]])
print(df.iloc[1:3])
print(df.iloc[:, 0])
print(df.iloc[:, [0, 1]])
print(df.iloc[1:3, [0, 1]])
```

```
      name  age gender
0    Alice   25      F
1      Bob   30      M
2  Charlie   35      M
3    David   40      M
4    Emily   45      F
name      Alice
age          25
gender        F
Name: 0, dtype: object
      name  age gender
0    Alice   25      F
2  Charlie   35      M
4    Emily   45      F
      name  age gender
1      Bob   30      M
2  Charlie   35      M
3    David   40      M
0      Alice
1        Bob
2    Charlie
3      David
4      Emily
Name: name, dtype: object
      name  age
0    Alice   25
1      Bob   30
2  Charlie   35
3    David   40
4    Emily   45
      name  age
1      Bob   30
2  Charlie   35
3    David   40
```

```python
import pandas as pd
data = {'name': ['Alice', 'Bob', 'Charlie', 'David', 'Bob'],
        'age': [25, 30, 35, 40, 30],
        'gender': ['F', 'M', 'M', 'M', 'M']}
df = pd.DataFrame(data)
# count the number of missing values in each column
print(df.isnull().sum())
# filter rows that contain at least one missing value
print(df[df.isnull().any(axis=1)])
# create a mask of boolean values for missing values
mask = df.isnull()
# select elements where the mask is True
print(df[mask])
# fill any null values with a specified value
df.fillna(0, inplace=True)
# fill null values with the value from the previous row
df.fillna(method='ffill', inplace=True)
# fill null values with the value from the next row
df.fillna(method='bfill', inplace=True)
# interpolate null values using linear interpolation
df.interpolate(method='linear', inplace=True)
# interpolate null values using polynomial interpolation
df.interpolate(method='polynomial', order=2, inplace=True)
# drop duplicate rows based on all columns
df.drop_duplicates(inplace=True)
# drop duplicate rows based on a subset of columns
df.drop_duplicates(subset=['name', 'age'], inplace=True)
# drop rows with missing values
df.dropna(inplace=True)
# drop columns with missing values
df.dropna(axis=1, inplace=True)
# replace inconsistent values with a constant value
df.replace({'gender': {'m': 'M'}}, inplace=True)
# convert string values to lowercase
df['name'] = df['name'].str.lower()
# clip extreme values to a lower and upper bound
df['age'] = df['age'].clip(lower=0, upper=100)
# remove extreme values based on a percentile threshold
q = df['age'].quantile(0.95)
df = df[df['age'] < q]
```

statistical methods available in pandas:

describe(): Generates descriptive statistics of a DataFrame or Series, including count, mean, standard deviation, minimum, maximum, and quartile values.

min(): Returns the minimum value of a DataFrame or Series.

max(): Returns the maximum value of a DataFrame or Series.

mean(): Calculates the mean of a DataFrame or Series.

median(): Calculates the median of a DataFrame or Series.

mode(): Calculates the mode(s) of a DataFrame or Series.

std(): Calculates the standard deviation of a DataFrame or Series.

var(): Calculates the variance of a DataFrame or Series.

quantile(): Calculates the specified quantile value(s) of a DataFrame or Series.

skew(): Calculates the skewness of a DataFrame or Series.

kurtosis(): Calculates the kurtosis of a DataFrame or Series.

corr(): Calculates the correlation between columns of a DataFrame.

cov(): Calculates the covariance between columns of a DataFrame.

sum(): Calculates the sum of values in a DataFrame or Series.

count(): Calculates the number of non-null values in a DataFrame or Series.

prod(): Calculates the product of values in a DataFrame or Series.

cumsum(): Calculates the cumulative sum of values in a DataFrame or Series.

cumprod(): Calculates the cumulative product of values in a DataFrame or Series.

value_counts(): Calculates the frequency of unique values in a DataFrame or Series.