Pandas DataFrame Function

DataFrame.append()

Pandas append() function is used to add the rows of other dataframe to the end of the given dataframe, returning a new dataframe object.

```
DataFrame.append(other, ignore_index=False)
```

Parameters:

- **other :** DataFrame or Series/dict-like object, or list of these
- **ignore_index :** If True, do not use the index labels.

.

Return Type: appended : DataFrame

_append

```
import pandas as pd
a1=pd.DataFrame({'a':[12,13,45,67,89,90],'b':[21,22,89,76,5,34]})
b1=pd.DataFrame({'a':[410,420],'b':[310,320],'c':[220,330]})

a1.append(b1)
```

maintain continue index use ignore_index parameter

```
import pandas as pd
a1=pd.DataFrame({'a':[12,13,45,67,89,90],'b':[21,22,89,76,5,34]})
b1=pd.DataFrame({'a':[410,420],'b':[310,320],'c':[220,330]})

c1=a1.append(b1,ignore_index=True)
c1
```

Adding Column into data Set

```
c1['dept']=['development' for i in range(0,8)]
c1
```

Pandas DataFrame Function

If you want to add the new column at a specific column position, use the pandas dataframe insert() function.

```
df.insert(loc, column, value, allow_duplicates=False)
```

```
import pandas as pd
a1=pd.DataFrame({'a':[12,13,45,14,89,9],'b':[21,22,89,76,5,34]})
b1=pd.DataFrame({'a':[410,420],'b':[310,320],'c':[220,330]})

c1=a1.append(b1,ignore_index=True,sort=True)
c1
c1.insert(1,'Dept',['devlop' for i in range(0,8)])
c1
```

DataFrame.apply()

- Pandas apply() function allows the user to pass a function and apply it to every single value of the Pandas series
- The objects that are to be passed to function are Series objects whose index is either the DataFrame's index, i.e., axis=0 or the DataFrame's columns, i.e., axis=1.
- By default, the result_type=None and the final return type is inferred from the return type of the applied function. Otherwise, it depends on the result_type argument.

Syntax:

DataFrame.apply(func, axis=0)

Parameters:

- **func**: It is a function that is to be applied to each column or row.
- **axis**: {0 or 'index', 1 or 'columns'}, default value 0

  It is an axis along which the function is applied. It can have two values:

  0 or 'index': It applies the function to each of the columns.

  1 or 'columns': It applies the function to each of the rows.

Pandas DataFrame Function

```
import pandas as pd
import numpy as np

df=pd.DataFrame({'id':[1,2,3],'design':[12,13,14],'dev':[21,22,23]})
print(df)
sm=df.apply(np.sum,axis=1)
print(sm)
```

DataFrame.aggregate()

The main task of DataFrame.aggregate() function is to apply some aggregation to one or more column. Most frequently used aggregations are:

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

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

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

```
import pandas as pd
import numpy as np
df=pd.DataFrame([[12,34,45],[45,6,34],[78,34,76],[np.nan,np.nan,np.nan]],columns=
['a','b','c'])
print(df)
```

Using Aggregate Function

```
import pandas as pd
```

Pandas DataFrame Function

```
import numpy as np
df=pd.DataFrame([[12,34,45],[45,6,34],[78,34,76],[np.nan,np.nan,np.nan]],columns=
['a','b','c'])
print(df)
rs=df.agg(['sum','min','max'])
print(rs)
```

DataFrame.assign()

The assign() method is also responsible for adding a new column into a DataFrame.

If we re-assign an existing column, then its value will be overwritten.

DataFrame.assign(**kwargs)

Parameters

- **kwargs:** keywords are the column names. These keywords are assigned to the new column if the values are callable. If the values are not callable, they are simply assigned.

```
import pandas as pd
import numpy as np

df=pd.DataFrame()
df['subject']=['s1','s2','s3']
df.assign(marks=[45,56,76])
```

DataFrame.count()

CREATIVE INSTITUTE DATA SCIENCE

Pandas DataFrame Function

The Pandas count() is defined as a method that is used to count the number of non-NA cells for each column or row. It is also suitable to work with the non-floating data.

Syntax:

1. DataFrame.count(axis=0, level=None, numeric_only=False)

```
import pandas as pd
import numpy as np

df=pd.DataFrame({"sub":['s1','s2','s3'],'marks':[12,13,np.nan]})

print(df)
df.count()
```

DataFrame.describe()

The describe() method is used for calculating some statistical data like percentile, mean and std of the numerical values of the Series or DataFrame.

It analyzes both numeric and object series and also the DataFrame column sets of mixed data types.

1. DataFrame.describe(percentiles=None, include=None, exclude=None)

```
import pandas as pd
import numpy as np
a1 = pd.Series([10, 20, 30])
a1.describe()
```

DataFrame.drop_duplicates()

1. DataFrame.drop_duplicates(subset=None, keep='first', inplace=False)

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

Pandas DataFrame Function

```
emp={"name":['a','b','c','a'],"salary":[12000,13000,14000,12000]}
df=pd.DataFrame(emp)

print(df)

df=df.drop_duplicates()
print(df)
```

DataFrame.head()

The head() returns the first n rows for the object based on position.

If your object has the right type of data in it, it is useful for quick testing. This method is used for returning top n (by default value 5) rows of a data frame or series.

1. DataFrame.head(n=5)

```
df = pd.DataFrame({'language':['C', 'C++', 'Python', 'Java','PHP']})
df.head()
df.head(3)
```

DataFrame.groupby()

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)

This operation consists of the following steps for aggregating/grouping the data:

- **Splitting datasets**

Pandas DataFrame Function

- **Analyzing data**
- **Aggregating or combining data**

> ***Note: The result of Groupby operation is not a DataFrame, but dict of DataFrame objects.***

Split data into groups

There are multiple ways to split any object into the group which are as follows:

- obj.groupby('key')
- obj.groupby(['key1','key2'])
- obj.groupby(key,axis=1)

- **Aggregation:** Computes summary statistic.
- **Filtration:** It filters the data by discarding it with some condition.

```
import pandas as pd
import numpy as np

data = {'Name': ['sachin', 'raj', 'virat', 'dhoni'],
   'Percentage': [82, 98, 91, 87],
   'Course': ['B.Sc','B.Ed','M.Phill','BA']}
df = pd.DataFrame(data)

grouped = df.groupby('Course')
print(grouped['Percentage'].agg(np.mean))
```

Filtration

The **filter()** function filters the data by defining some criteria and returns the subset of data

```
import pandas as pd
import numpy as np

data = {'Name': ['sachin', 'raj', 'virat', 'dhoni'],
   'Percentage': [82, 98, 91, 87],
   'Course': ['B.Sc','B.Ed','M.Phill','BA']}
df = pd.DataFrame(data)

grouped = df.groupby('Course')
```

Pandas DataFrame Function

```
print(grouped['Percentage'].agg(np.mean))


grouped = df.groupby('Course')
rs=df.groupby('Course').filter(lambda x:len(x)>=1)
print(rs)
```

DataFrame.sort()

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

- By label
- By Actual value

By label

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.

```
import pandas as pd
import numpy as np

df=pd.DataFrame(np.random.randn(10,2),index=[1,3,7,2,4,5,9,8,0,6],columns=['col2'
,'col1'])
dff=info.sort_index()
print(dff)
```

- **Order of Sorting**

The order of sorting can be controlled by passing the Boolean value to the ascending parameter.

CREATIVE INSTITUTE DATA SCIENCE

Pandas DataFrame Function

```
import pandas as pd
import numpy as np

df=pd.DataFrame(np.random.randn(10,2),index=[1,3,7,2,4,5,9,8,0,6],columns=['col2'
,'col1'])
dff=info.sort_index(ascending=False)
print(dff)
```

By Actual Value

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'col1':[7,1,8,3],'col2':[8,12,4,9]})
print(df)
dff = df.sort_values(by='col2')
print(dff)
```