# series

July 22, 2024

## 1 Pandas - Series

• One Dimensional ndarray with index labels. • It is Homogenous , Labeled , Size Immutable. • Labels need not to be unique but hashable.

[1]:
```
#Importing
import pandas as pd
```

### 1.1 Creating Series

[2]:
```
pd.Series() # Create a empty series
```

[2]:
```
Series([], dtype: object)
```

[3]:
```
a=pd.Series([i for i in range(6)]) # Creating a iterable as series
print(a)
```

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

[4]:
```
a={'1st Index':['a','b','c'],'2nd Index':['d','e','f'],'3rd Index':
  ↪['g','h','i']}
a=pd.Series(a)
print(a) # To show Dict keys are assigned in Index and values in data.
```

```
1st Index    [a, b, c]
2nd Index    [d, e, f]
3rd Index    [g, h, i]
dtype: object
```

## 1.2 Inspecting

[5]:
```
a=dict(zip([chr(i) for i in range(97,107)],[[i**2,i**3] for i in range(10)]))
```

[6]:
```
s=pd.Series(a)
print(s)
```

```
a       [0, 0]
b       [1, 1]
c       [4, 8]
d      [9, 27]
e     [16, 64]
f    [25, 125]
g    [36, 216]
h    [49, 343]
i    [64, 512]
j    [81, 729]
dtype: object
```

[7]:
```
# Head
print("Head(5)\n",s.head(5),"\n")                # top 5

# Tail
print("Tail(5)\n",s.tail(5),'\n\n')                # Bottom 5

# Sample
print("Sample(5):\n",s.sample(5),'\n')              # Random 5
```

```
Head(5)
 a      [0, 0]
b      [1, 1]
c      [4, 8]
d     [9, 27]
e    [16, 64]
dtype: object

Tail(5)
 f    [25, 125]
g    [36, 216]
h    [49, 343]
i    [64, 512]
j    [81, 729]
dtype: object


Sample(5):
 d      [9, 27]
i    [64, 512]
```

```
b       [1, 1]
a       [0, 0]
e     [16, 64]
dtype: object
```

[8]:
```
a=dict(zip([chr(i) for i in range(97,107)],[i for i in range(10)]))
s=pd.Series(a)
s
```

[8]:
```
a    0
b    1
c    2
d    3
e    4
f    5
g    6
h    7
i    8
j    9
dtype: int64
```

[9]:
```
print("Describe\n",s.describe(include='all'))   # Descriptive statistics on the␣
↪Series
```

```
Describe
 count    10.00000
mean      4.50000
std       3.02765
min       0.00000
25%       2.25000
50%       4.50000
75%       6.75000
max       9.00000
dtype: float64
```

### 1.3 Manipulations and Transformations

[10]:
```
# Change type
print("Astype \n",s.astype('str'))  # Change int to object
```

```
Astype
 a    0
b    1
c    2
d    3
e    4
f    5
```

```
g    6
h    7
i    8
j    9
dtype: object
```

[11]:
```
# Copy
s1=s.copy()
print(s1)
```

```
a    0
b    1
c    2
d    3
e    4
f    5
g    6
h    7
i    8
j    9
dtype: int64
```

[12]:
```
def sqr(i):
    return i**2
s.apply(sqr)   # Apply Function
```

[12]:
```
a     0
b     1
c     4
d     9
e    16
f    25
g    36
h    49
i    64
j    81
dtype: int64
```

[13]:
```
# Map Function
def hl(i):
    if i<5:
        return 'low'
    if i>=5:
        return "h!gh"
s1=s.map(hl)
s1
```

[13]:
```
a     low
b     low
c     low
d     low
e     low
f    h!gh
g    h!gh
h    h!gh
i    h!gh
j    h!gh
dtype: object
```

[14]:
```
s1.str.replace('!','i')
```

[14]:
```
a     low
b     low
c     low
d     low
e     low
f    high
g    high
h    high
i    high
j    high
dtype: object
```

### 1.4 Aggregation and Statistical Operations

[15]:
```
# Sum
print("Sum \t:",s.sum())
print("Mean \t:",s.mean())
print("Median\t:",s.median())
print("Std\t:",s.std())
print("Min\t:",s.min())
print("Max \t:",s.max())
print(f"ValueCounts\t:\n{s.value_counts()}")
```

```
Sum     : 45
Mean    : 4.5
Median  : 4.5
Std     : 3.0276503540974917
Min     : 0
Max     : 9
ValueCounts     :
0    1
1    1
2    1
```

```
3    1
4    1
5    1
6    1
7    1
8    1
9    1
Name: count, dtype: int64
```

### 1.5 Sorting and Indexing

[16]:
```
s.sort_values(ascending=False)  # Sort based on Values
```

[16]:
```
j    9
i    8
h    7
g    6
f    5
e    4
d    3
c    2
b    1
a    0
dtype: int64
```

[17]:
```
s.sort_index(ascending=False)   # Sort based on index
```

[17]:
```
j    9
i    8
h    7
g    6
f    5
e    4
d    3
c    2
b    1
a    0
dtype: int64
```

### 1.6 Selection , Filtering and Handling Missing Data

[18]:
```
# Selecting
print(s[0:5])       # Slicing
print(s[[2,3,5]])   # Multiple Selection
```

```
a    0
b    1
```

```
c    2
d    3
e    4
dtype: int64
c    2
d    3
f    5
dtype: int64
```

```
C:\Users\rajen\AppData\Local\Temp\ipykernel_16052\135015164.py:3: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  print(s[[2,3,5]])   # Multiple Selection
```

[19]:
```
s=pd.Series([1,2,None,4,None,5,3,2])
```

[20]:
```
print(s.loc[0:5])       #Label Indexing
print(s.iloc[[1,2,3]])  #Integer Indexing
```

```
0    1.0
1    2.0
2    NaN
3    4.0
4    NaN
5    5.0
dtype: float64
1    2.0
2    NaN
3    4.0
dtype: float64
```

[21]:
```
s
```

[21]:
```
0    1.0
1    2.0
2    NaN
3    4.0
4    NaN
5    5.0
6    3.0
7    2.0
dtype: float64
```

[22]:
```
# Isnull - Returns a bool
print(s.isnull())
print("Sum of Total Null :",s.isnull().sum())
```

```
0    False
1    False
2     True
3    False
4     True
5    False
6    False
7    False
dtype: bool
Sum of Total Null : 2
```

[23]:
```
# Isna
print(s.isna())
```

```
0    False
1    False
2     True
3    False
4     True
5    False
6    False
7    False
dtype: bool
```

[24]:
```
s.fillna(5)          # Fills missing values
```

[24]:
```
0    1.0
1    2.0
2    5.0
3    4.0
4    5.0
5    5.0
6    3.0
7    2.0
dtype: float64
```

[25]:
```
s.dropna(inplace=True)  # Removes the NaN values
s
```

[25]:
```
0    1.0
1    2.0
3    4.0
5    5.0
6    3.0
7    2.0
dtype: float64
```

[26]:
```
t=pd.Series(range(10))
```

[27]:
```
t.isin(s)       # Checks t is in s
```

[27]:
```
0    False
1     True
2     True
3     True
4     True
5     True
6    False
7    False
8    False
9    False
dtype: bool
```

[28]:
```
# Filetering
t>4 # Returns all Bool
```

[28]:
```
0    False
1    False
2    False
3    False
4    False
5     True
6     True
7     True
8     True
9     True
dtype: bool
```

[29]:
```
t[t>4] # Returns all values that are true
```

[29]:
```
5    5
6    6
7    7
8    8
9    9
dtype: int64
```

I hope you found this information helpful! Feel free to save this post for future reference. Let's continue to learn and grow together!

Rajendra Prasad