pandas Series

Setup

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

Creation

Create Series from ndarray

```
s1 = pd.Series(np.arange(0,5))
s1
```

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

```
type(np.arange(0, 5))
```

```
numpy.ndarray
```

Create Series with index

```
s2 = pd.Series(np.arange(0,5), index=['a','b','c','d','e'])
s2
```

```
a    0
b    1
c    2
d    3
e    4
dtype: int64
```

Assign index to Existing Series

```
s2.index = ['A','B','C','D','E']
s2
```

```
A    0
B    1
C    2
D    3
E    4
dtype: int64
```

```
s2.index
```

```
Index(['A', 'B', 'C', 'D', 'E'], dtype='object')
```

Create One-Item Series from Scalar

```
s3 = pd.Series(5)
s3
```

```
0    5
dtype: int64
```

Create Series from list

```
s4 = pd.Series([1,2,3,4,5])
s4
```

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

```
s4.index
```

```
RangeIndex(start=0, stop=5, step=1)
```

Create Series from dict

```
from datetime import date
bdays = {
    'Aaron': date(2001, 10, 10),
    'Brian': date(2002, 6, 6),
    'Christine': date(2003, 2, 2),
    'Di': date(2004, 9, 9),
}
s5 = pd.Series(bdays)
s5
```

```
Aaron        2001-10-10
Brian        2002-06-06
Christine    2003-02-02
Di           2004-09-09
dtype: object
```

```
ar = np.array([1,2,3,np.nan,5,6,7,np.nan,9,10])
ar
```

```
array([ 1.,  2.,  3., nan,  5.,  6.,  7., nan,  9., 10.])
```

```
ar.mean()
```

```
nan
```

```
s6 = pd.Series(ar)
s6
```

```
0     1.0
1     2.0
2     3.0
3     NaN
4     5.0
5     6.0
6     7.0
7     NaN
8     9.0
9    10.0
dtype: float64
```

```
s6.mean(), sum([1,2,3,5,6,7,9,10])/8
```

```
(5.375, 5.375)
```

```
s6.mean(skipna=False)
```

```
nan
```

The index and values Properties

```
s5.index
```

```
Index(['Aaron', 'Brian', 'Christine', 'Di'], dtype='object')
```

```
s5.values
```

```
array([datetime.date(2001, 10, 10), datetime.date(2002, 6, 6),
       datetime.date(2003, 2, 2), datetime.date(2004, 9, 9)], dtype=object)
```

```
type(s5.values)
```

```
numpy.ndarray
```

Access

loc[] and iloc[]

```
s7 = pd.Series(np.random.sample(5), index=['a','b','c','d','e'])
s7
```

```
a    0.052907
b    0.550551
c    0.704588
d    0.734350
e    0.383290
dtype: float64
```

```
s7.loc['a'], s7.iloc[0]
```

```
(0.05290694875658386, 0.05290694875658386)
```

```
s7.loc['b':'d']
```

```
b    0.550551
c    0.704588
d    0.734350
dtype: float64
```

```
s7.iloc[1:4]
```

```
b    0.550551
c    0.704588
d    0.734350
dtype: float64
```

```
s7.loc[['a','c','d']]
```

```
a    0.052907
c    0.704588
d    0.734350
dtype: float64
```

```
s7.iloc[[0,2,4]]
```

```
a    0.052907
c    0.704588
e    0.383290
dtype: float64
```

Manipulation

Alignment

```
grades1 = pd.Series([17, 44, 28, 8, 3], index=['A','B','C','D','F'])
grades2 = pd.Series([76, 122, 151, 21, 0], index=['D','C','B','A','F'])
```

```
grades1
```

```
A    17
B    44
C    28
D     8
F     3
dtype: int64
```

```
grades2
```

```
D     76
C    122
B    151
A     21
F      0
dtype: int64
```

```
grades_all = grades1 + grades2
grades_all
```

```
A     38
B    195
C    150
D     84
F      3
dtype: int64
```

```
grades1 = pd.Series([17, 44, 28, 8, 3], index=['A','B','C','D','F'])
grades2 = pd.Series([76, 122, 151, 21], index=['D','C','B','A'])
grades_all = grades1 + grades2
grades1, grades2, grades_all
```

```
(A    17
 B    44
 C    28
 D     8
 F     3
 dtype: int64,
 D     76
 C    122
 B    151
 A     21
 dtype: int64,
 A     38.0
 B    195.0
 C    150.0
 D     84.0
 F      NaN
 dtype: float64)
```

```
grades_all = grades1.add(grades2, fill_value=10)
grades_all
```

```
A     38.0
B    195.0
C    150.0
D     84.0
F     13.0
dtype: float64
```

Comparing Series

```
dies1 = pd.Series(np.random.randint(1, 7, (100,)))
dies2 = pd.Series(np.random.randint(1, 7, (100,)))
dies1 == dies2
```

```
0     False
1     False
2     False
3     False
4     False
      ...
95    False
96    False
97    False
98    False
99    False
Length: 100, dtype: bool
```

```
type(dies1 == dies2)
```

```
pandas.core.series.Series
```

```
dies1[dies1 == dies2]
```

```
6     2
15    1
17    1
18    5
27    3
38    2
40    4
50    4
53    6
57    3
64    5
68    4
70    3
75    3
76    3
85    1
91    5
dtype: int64
```

Element-wise Operations

```
np.random.seed(1)
exam_grades = pd.Series(np.random.randint(60,101,100))
exam_grades
```

```
0     97
1     72
2     68
3     69
4     71
      ..
95    87
96    81
97    71
98    67
99    73
Length: 100, dtype: int64
```

```
curved_grades = exam_grades.multiply(1.05)
curved_grades
```

```
0     101.85
1      75.60
2      71.40
3      72.45
4      74.55
       ...
95     91.35
96     85.05
97     74.55
98     70.35
99     76.65
Length: 100, dtype: float64
```

```
def convert_to_letter(grade):
    if grade >= 90:
        return 'A'
    elif grade >= 80:
        return 'B'
    elif grade >= 70:
        return 'C'
    elif grade >= 65:
        return 'D'
    else:
        return 'F'
```

```
letter_grades = curved_grades.apply(convert_to_letter)
letter_grades
```

```
0     A
1     C
2     C
3     C
4     C
     ..
95    A
96    B
97    C
98    C
99    C
Length: 100, dtype: object
```

```
part_letter_grade = curved_grades.iloc[:50].apply(convert_to_letter)
part_letter_grade
```

```
0     A
1     C
2     C
3     C
4     C
5     D
6     C
7     F
8     C
9     F
10    C
11    C
12    D
13    B
14    B
15    A
16    B
17    B
18    C
19    A
20    A
21    C
22    D
23    B
24    B
25    A
26    A
27    B
28    C
29    C
30    C
31    B
32    F
33    F
34    B
35    C
36    B
37    C
38    C
39    A
40    C
41    D
42    D
43    B
44    D
45    D
46    B
47    C
48    A
49    C
dtype: object
```