July-19

describe() → description of Numeric Column

Python-tutorial 12

- Pandas Data Manipulation:

```
import pandas as pd
titanic_train.describe()
```

(analysis of only columns with data type = int/float)

	passangerid	Age	Fare
count	150.000000	126.000000	156.000000
mean			
std			
min			
25%			
50%			
75%			
max			

(no. of records) (means some rows have null values)

suppose we have some data:

(1, 3, 4, 8, 10, 11, 2, 1, 3, 4, 8, 12, 14, 16)

first arrange it in ascending order:

1, 1, 2, 2, 4, 4, 8, 10, 11, 12, 14, 16

min · 25% · 50% · 75% · 100% (max.)

(25th percentile record)

- how to find description of other data-types:

way-1 (not recommended)

```
titanic_train[['Name','Sex',Ticket']].describe()
```

	Name	Sex	Ticket
count	156	156	156
unique	156	2	145
top	MADHU	Female	2651
freq	1	100	2

(freq of top data-set)

way-2 (for any particular data-type)

~~a = titanic_train.dtypes[titanic_train.dtypes == "object"].index~~

a

```
titanic_train.dtypes == 'object'
```

Passangerid	False
Name	True
Sex	True
Age	False
Ticket	True
Fare	False

```
titanic_train.dtypes[titanic_train.dtypes == 'object']
```

Name	object
Sex	object
Ticket	object

↑ data-type = Series

So we can find index of series:

```
titanic_train.dtypes[titanic_train.dtypes == "object"].index
```

```
Index(['Name', 'Sex', 'Ticket'],
      dtype='object')
```

Dataframe from this index:

```
a = titanic_train.dtypes[titanic_train.dtypes == "object"].index
titanic_train[a].describe()
```

	Name	Sex	Ticket
count	156	156	156
unique	156	2	145
top	MADHU	Female	2651
freq	1	100	2

```
titanic_train.columns
```

```
Index(['passengerid', 'Name', 'Sex', 'Age',
       'Ticket', 'Fare'])
```

```
titanic_train['survived'][10]
```

10th - index

```
10  1
```

'sibSp',

```
titanic_train['survived'][10:15:2]
```

in form of table, index 10 to 15 with gap of 2

```
10  1  0
12  0  1
14  0  0
```

- Sorting:

```
titanic_train = pd.read_csv('address')
titanic_train.describe()
result = titanic_train.describe()
h = titanic_train['Age'][::2]
b = pd.DataFrame(h)
k = b.sort_values(by='Age')
k
```

- mapping some data of column with other data:

```
new_Pclass = pd.Categorical
             (titanic_train['PClass'])
```

```
new_Pclass
[3,1,3,1,---,1,3,3,1]  ← Pclass column
Length : 156
Categories (3, int64) : [1,2,3]
```

- insert data in new column: (exact 0)

```
import numpy as np
char_cabin = tit_tra["Cabin"].astype(str)   # convert data in str type
new_cabin = [cabin[0] for cabin in char_cabin]   # : takes first letter of strings
new_cabin = pd.Categories(new_cabin)
```

```
new_cabin
[n,C,n,C,n, ---C,n,n,n,n]
Length : 156
Categories (8, object) : [A,B,C,D,E,F,G,n]
```

Function for selecting rows inside numpy { loc, iloc - integer location, ix }

```
tit_tra['cabin_i'] = new_cabin
```
← added new column in tit_tra

- index having null-values:

```
np.where(titanic_train['Age'].isnull()==True)
```
→ extract (where the given condition is true).

```
np.where(titanic_train['Fare'] =
        max(titanic_train['Fare']))
```
→ (array[27, 88], dtype=int64)

information of person given max. fare:

```
titanic_train.iloc[np.where( )=max( )]
```
row index say

	PassengerId	Name	Fare	Age
27	28	MADHU	630.00	22
88	69	NIDHI	630.00	23

we are interested only in Name & Age of passenger given max. fare:

```
titanic_train.iloc[row_index][['Name','Age']]
```

- add two columns & make third:

```
tit['Family'] = tit['SibSp']+tit['Pearch']
most_family = np.where(tit['Family']
        ==max(tit['Family']))
```

```
most_family
tit.iloc[most_family]
```
→ information of family having max. members of members

- list → dict → Series → Dataframe

```
my_data = [10, 20, 30]  lable = ['a', 'b', 'c']
pd.Series(my_data)   (∵ convert list in series)
```

```
0   10
1   20
2   30
```

(diff. b/w list & series is that in list you won't be able to see indexes)

- if you don't want these default indexes:

```
se = pd.Series(my_data, index = lable)
```

```
a   10
b   20
c   30
```

even if you override new index but system will remember default index too

se[1] → 20 or se['a'] → 10

- dict → dataframe

```
d = {'a': 1, 'b': 2, 'c': 3}
pd.DataFrame(d)
```

↳ error: you must pass index

```
pd.DataFrame(d, index = lable)
```

- addition of series: (also multiplication ...)

```
ser1 = pd.Series([1,2,3,4], ['A', 'B', 'A', 'D'])
ser2 = pd.Series([5,6,7,8], ['A', 'D', 'E', 'B'])
ser1 + ser2
```

↳

A	6	(1+5)
A	8	(3+5)
B	NaN	
C		
D	12	(4+8)
E	NaN	