Session - 9

Python for Data Science

Pandas - Library for Data Analysis

Pandas Data Structures

- Series
- DataFrame

Both of them are built on the top of Numpy (... so are fast)

Series

- A Series is a one-dimensional object similar to an array, list, or column in a table.
- It will assign a labeled index to each item in the Series.
- By default, each item will receive an index label from 0 to N, where N is the length of the Series minus one.

```
>>> import pandas as pd
>>> s = pd.Series(['Usain Bolt', 9.63, 'Yohan Blake', 9.75, 'Justin Gatlin', 9.79])
>>> s
0       Usain Bolt
1             9.63
2      Yohan Blake
3             9.75
4    Justin Gatlin
5             9.79
dtype: object
>>>
```

Series (Cont...)

```
>>> s = pd.Series(['Usain Bolt', 9.63, 'Yohan Blake', 9.75, 'Justin Gatlin', 9.79], index=['A', 'B', 'C', 'D', 'E', 'F'])
>>> s
A       Usain Bolt
B             9.63
C      Yohan Blake
D             9.75
E    Justin Gatlin
F             9.79
dtype: object
>>>
```

You can also use Dictionary to create Series

```
>>> runners  =  {'Usain Bolt': 9.63, 'Yohan Blake': 9.75, 'Justin Gatlin': 9.79}
>>> runners
{'Yohan Blake': 9.75, 'Usain Bolt': 9.63, 'Justin Gatlin': 9.79}
>>> r = pd.Series(runners)
>>> r
Justin Gatlin    9.79
Usain Bolt       9.63
Yohan Blake      9.75
dtype: float64
>>>
```

Series (Cont...)

Accessing Elements are Easy

```
>>> r['Usain Bolt']
9.6300000000000008
>>> r['Yohan Blake']
9.75
>>>
```

```
>>> r[['Yohan Blake', 'Yohan Blake']]
Yohan Blake    9.75
Yohan Blake    9.75
dtype: float64
>>>
>>> r[['Yohan Blake', 'Usain Bolt']]
Yohan Blake    9.75
Usain Bolt     9.63
dtype: float64
>>>
```

```
>>> runners  =  {'Usain Bolt': 9.63, 'Yohan Blake': 9.75, 'Justin Gatlin': 9.79, 'Tyson Gay': 9.8, 'Ryan Bailey': 9.88, 'Churandy Martina': 9.94, 'Richard T
ompson': 9.98, 'Asafa Powell': 11.99}
>>>
>>> r = pd.Series(runners)
>>> r
Asafa Powell        11.99
Churandy Martina     9.94
Justin Gatlin        9.79
Richard Thompson     9.98
Ryan Bailey          9.88
Tyson Gay            9.80
Usain Bolt           9.63
Yohan Blake          9.75
dtype: float64
>>> r[r>10]
Asafa Powell    11.99
dtype: float64
>>> r[r<9.8]
Justin Gatlin    9.79
Usain Bolt       9.63
Yohan Blake      9.75
dtype: float64
>>>
```

Series (Cont...)

```
>>> r <9.8
Asafa Powell          False
Churandy Martina      False
Justin Gatlin          True
Richard Thompson      False
Ryan Bailey           False
Tyson Gay             False
Usain Bolt             True
Yohan Blake            True
dtype: bool
>>> my_requirement = r<9.8
>>> r[my_requirement]
Justin Gatlin    9.79
Usain Bolt       9.63
Yohan Blake      9.75
dtype: float64
>>>
```

Actually r<9.8 or r > 10
are series with True & False values.

Those can be passed to r - returning corresponding “True” value Series.

Series(Cont...)

One can change values on the fly

You can also check the presence of the value

```
>>> print ('Ironman' in r)
False
>>> print ('Superman' in r)
True
>>> print ('Milkha Singh' in r)
False
>>>
```

```
>>> r[['Superman']]
Superman   NaN
dtype: float64
>>> r['Superman']=5.0
>>> r['Batman'] = 6.0
>>> r
Asafa Powell         11.99
Churandy Martina      9.94
Justin Gatlin         9.79
Richard Thompson      9.98
Ryan Bailey           9.88
Tyson Gay             9.80
Usain Bolt            9.63
Yohan Blake           9.75
Superman              5.00
Batman                6.00
dtype: float64
>>> r[r<9] = 1.0
>>> r
Asafa Powell         11.99
Churandy Martina      9.94
Justin Gatlin         9.79
Richard Thompson      9.98
Ryan Bailey           9.88
Tyson Gay             9.80
Usain Bolt            9.63
Yohan Blake           9.75
Superman              1.00
Batman                1.00
dtype: float64
>>>
```

Series (Cont...)

Mathematics

```
>>> r * r
Asafa Powell          143.7601
Churandy Martina       98.8036
Justin Gatlin          95.8441
Richard Thompson       99.6004
Ryan Bailey            97.6144
Tyson Gay              96.0400
Usain Bolt             92.7369
Yohan Blake            95.0625
Superman                1.0000
Batman                  1.0000
dtype: float64
```

```
>>> r/2
Asafa Powell          5.995
Churandy Martina      4.970
Justin Gatlin         4.895
Richard Thompson      4.990
Ryan Bailey           4.940
Tyson Gay             4.900
Usain Bolt            4.815
Yohan Blake           4.875
Superman              0.500
Batman                0.500
dtype: float64
```

```
>>> r + r
Asafa Powell          23.98
Churandy Martina      19.88
Justin Gatlin         19.58
Richard Thompson      19.96
Ryan Bailey           19.76
Tyson Gay             19.60
Usain Bolt            19.26
Yohan Blake           19.50
Superman               2.00
Batman                 2.00
dtype: float64
```

```
>>> np.square(r)
Asafa Powell          143.7601
Churandy Martina       98.8036
Justin Gatlin          95.8441
Richard Thompson       99.6004
Ryan Bailey            97.6144
Tyson Gay              96.0400
```

Series (Cont...)

Mathematics

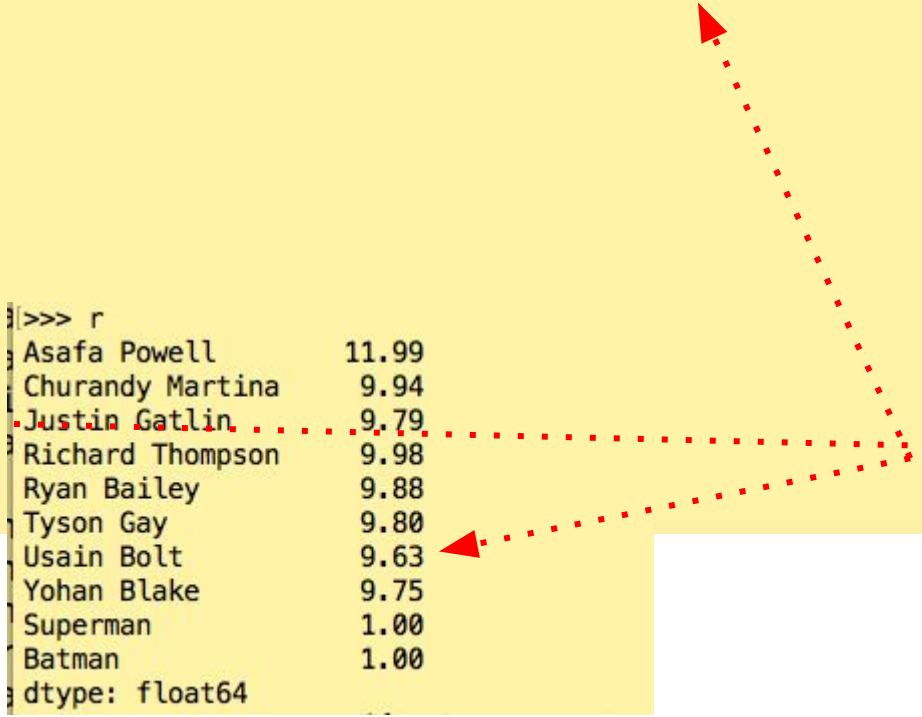
```
>>> new_r = pd.Series({'Lalita Baber': 15.0,  'O P Jaisha': 16.0, 'Kavita Raut': 20.0
>>> new_r
Kavita Raut     20.0
Lalita Baber    15.0
O P Jaisha      16.0
dtype: float64
```

```
>>> r + new_r
Asafa Powell       NaN
Batman             NaN
Churandy Martina   NaN
Justin Gatlin      NaN
Kavita Raut        NaN
Lalita Baber       NaN
O P Jaisha         NaN
Richard Thompson   NaN
Ryan Bailey        NaN
Superman           NaN
Tyson Gay          NaN
Usain Bolt         NaN
Yohan Blake        NaN
dtype: float64
```

Series (Cont...)

```
>>> new_r = pd.Series({'Lalita Baber': 15.0,  'O P Jaisha': 16.0, 'Kavita Raut': 20.0, 'Usain Bolt':9.63})
>>> r + new_r
Asafa Powell          NaN
Batman                NaN
Churandy Martina      NaN
Justin Gatlin         NaN
Kavita Raut           NaN
Lalita Baber          NaN
O P Jaisha            NaN
Richard Thompson      NaN
Ryan Bailey           NaN
Superman              NaN
Tyson Gay             NaN
Usain Bolt          19.26
Yohan Blake           NaN
dtype: float64
>>>
```

```
>>> r
Asafa Powell        11.99
Churandy Martina     9.94
Justin Gatlin        9.79
Richard Thompson     9.98
Ryan Bailey          9.88
Tyson Gay            9.80
Usain Bolt           9.63
Yohan Blake          9.75
Superman             1.00
Batman               1.00
dtype: float64
```



Series (Cont ...)

isnull & notnull instance methods

```
>>> new_r = pd.Series({'Lalita Baber': 15.0,  'O P Jaisha': 16.0, 'Kavita Raut': 20.0, 'Usain Bolt':9.63,  'Milkha Singh':None})
>>> new_r.isnull()
Kavita Raut     False
Lalita Baber    False
Milkha Singh     True
O P Jaisha      False
Usain Bolt      False
dtype: bool
>>> new_r.notnull()
Kavita Raut      True
Lalita Baber     True
Milkha Singh    False
O P Jaisha       True
Usain Bolt       True
dtype: bool
>>>
```

Series - Practice

- Use Fruits Data - create dictionary to work on Protein details. Use Fruits Names as Keys and values as Protien. Name dictionary as - “fruits”
- Create Series from “fruits” - call it “s_fruits”
- Perform
  - >, multiplication, np.square, addition, set values on the fly.
- Add “Onion” as key, with value= None, Recreate “s_fruits” Series
  - Observe “NaN” value
- Create new dictionary - “new_fruits” with keys “Raisins”, “Onion” (use same values of Protien as above)
- Perform “s_fruits” + “new_fruits”

DataFrame

"tabular" data: a data structure representing cases (rows), each of which consists of a number of observations or measurements (columns).

Player Name	Time	Height (Inches)	Weight (Pounds)	Country
Usain Bolt	9.63	76.77	209.439	JAM
Yohan Blake	9.75	70.86	167.551	JAM
Justin Gatlin	9.79	72.83	182.984	USA
Tyson Gay	9.8	70.07	165.347	USA
Ryan Bailey	9.88	75.98	216.053	USA
Churandy Martina	9.94	70.07	163.142	NED
Richard Thompson	9.98	70.01	176.37	TTO
Asafa Powell	11.99	70.8	191.802	JAM

DataFrame

```
>>> data = {'name':['Usain Bolt','Yohan Blake','Justin Gatlin','Tyson Gay','Ryan Bailey','Churandy Martina','Richard Thompson','Asafa Powell'],
...         'speed':[9.63,9.75,9.79,9.8,9.88,9.94,9.98,11.99],
...         'height':[76.77,70.86,72.83,70.07,75.98,70.07,70.01,70.8],
...         'weight':[209.439,167.551,182.984,165.347,216.053,163.142,176.37,191.802]}
>>>
```

```
>>> data
{'speed': [9.63, 9.75, 9.79, 9.8, 9.88, 9.94, 9.98, 11.99], 'name': ['Usain Bolt', 'Yohan Blake', 'Justin Gatlin', 'Tyson Gay', 'Ryan Bailey', 'Churandy Mart
ina', 'Richard Thompson', 'Asafa Powell'], 'height': [76.77, 70.86, 72.83, 70.07, 75.98, 70.07, 70.01, 70.8], 'weight': [209.439, 167.551, 182.984, 165.347,
216.053, 163.142, 176.37, 191.802]}
>>>
```

```
>>> runners = pd.DataFrame(data, columns=["name", "speed", "height", "weight"])
>>> runners
                name  speed  height   weight
0        Usain Bolt   9.63   76.77  209.439
1       Yohan Blake   9.75   70.86  167.551
2     Justin Gatlin   9.79   72.83  182.984
3         Tyson Gay   9.80   70.07  165.347
4       Ryan Bailey   9.88   75.98  216.053
5  Churandy Martina   9.94   70.07  163.142
6  Richard Thompson   9.98   70.01  176.370
7      Asafa Powell  11.99   70.80  191.802
>>>
```

DataFrame

```
>>> food = pd.read_csv("food.csv")
>>> food
                    Food  Index  Calories  Cholesterol  Total_Fat  Sodium  \
0          Frozen Broccoli      1      73.8          0.0        0.8    68.2
1              Carrots,Raw      2      23.7          0.0        0.1    19.2
2              Celery, Raw      3       6.4          0.0        0.1    34.8
3              Frozen Corn      4      72.2          0.0        0.6     2.5
4      Lettuce,Iceberg,Raw      5       2.6          0.0        0.0     1.8
5      Peppers, Sweet, Raw      6      20.0          0.0        0.1     1.5
6          Potatoes, Baked      7     171.5          0.0        0.2    15.2
7                     Tofu      8      88.2          0.0        5.5     8.1
8          Roasted Chicken      9     277.4        129.9       10.8   125.6
9        Spaghetti W/ Sauce     10     358.2          0.0       12.3  1237.1
10     Tomato,Red,Ripe,Raw     11      25.8          0.0        0.4    11.1
11        Apple,Raw,W/Skin     12      81.4          0.0        0.5     0.0
12                  Banana     13     104.9          0.0        0.5     1.1
13                  Grapes     14      15.1          0.0        0.1     0.5
14     Kiwifruit,Raw,Fresh     15      46.4          0.0        0.3     3.8
15                 Oranges     16      61.6          0.0        0.2     0.0
16                  Bagels     17      78.0          0.0        0.5   151.4
17            Wheat Bread     18      65.0          0.0        1.0   134.5
18            White Bread     19      65.0          0.0        1.0   132.5
19          Oatmeal Cookies     20      81.0          0.0        3.3    68.9
20                Apple Pie     21      67.2          0.0        2.1    75.4
```

DataFrame

```
>>> food.head()
                 Food  Index  Calories  Cholesterol  Total_Fat  Sodium  \
0      Frozen Broccoli      1      73.8          0.0        0.8    68.2
1          Carrots,Raw      2      23.7          0.0        0.1    19.2
2          Celery, Raw      3       6.4          0.0        0.1    34.8
3          Frozen Corn      4      72.2          0.0        0.6     2.5
4  Lettuce,Iceberg,Raw      5       2.6          0.0        0.0     1.8

   Carbohydrates  Dietary_Fiber  Protein    Vit_A  Vit_C  Calcium  Iron  \
0           13.6            8.5      8.0   5867.4  160.2    159.0   2.3
1            5.6            1.6      0.6  15471.0    5.1     14.9   0.3
2            1.5            0.7      0.3     53.6    2.8     16.0   0.2
3           17.1            2.0      2.5    106.6    5.2      3.3   0.3
4            0.4            0.3      0.2     66.0    0.8      3.8   0.1

   Price/Serving ($)
0               0.16
1               0.07
2               0.04
3               0.18
4               0.02
>>> food.tail(1)
                  Food  Index  Calories  Cholesterol  Total_Fat  Sodium  \
63  Beanbacn Soup,W/Watr     64     172.0          2.5        5.9   951.3

    Carbohydrates  Dietary_Fiber  Protein  Vit_A  Vit_C  Calcium  Iron  \
63           22.8            8.6      7.9  888.0    1.5     81.0   2.0

    Price/Serving ($)
63               0.67
>>> food.tail(3)
                  Food  Index  Calories  Cholesterol  Total_Fat  Sodium  \
61  New E Clamchwd,W/Mlk     62     163.7         22.3        6.6   992.0
62  Crm Mshrm Soup,W/Mlk     63     203.4         19.8       13.6  1076.3
```

- head()
- tail()
- info()
- describe()
- dtype
- columns
-

DataFrame

```
>>> food.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64 entries, 0 to 63
Data columns (total 14 columns):
Food                64 non-null object
Index               64 non-null int64
Calories            64 non-null float64
Cholesterol         64 non-null float64
Total_Fat           64 non-null float64
Sodium              64 non-null float64
Carbohydrates       64 non-null float64
Dietary_Fiber       64 non-null float64
Protein             64 non-null float64
Vit_A               64 non-null float64
Vit_C               64 non-null float64
Calcium             64 non-null float64
Iron                64 non-null float64
Price/Serving ($)   64 non-null float64
dtypes: float64(12), int64(1), object(1)
memory usage: 7.1+ KB
>>>
```

```
>>> runners.describe()
            speed     height      weight
count    8.000000   8.000000    8.000000
mean    10.095000  72.173750  184.086000
std      0.773656   2.756555   20.171263
min      9.630000  70.010000  163.142000
25%      9.780000  70.070000  167.000000
50%      9.840000  70.830000  179.677000
75%      9.950000  73.617500  196.211250
max     11.990000  76.770000  216.053000
>>>
```

Thank you!