In [1]:
```
# keyless dict - set

s1 = {1,2,1,2,3,4,5,1,2,1}
```

In [2]:
```
s1
```

Out[2]:
```
{1, 2, 3, 4, 5}
```

In [7]:
```
l1 = [12,34,56,32,12,3,4,57,54,46,56,78,99,100,23]
l1
```

Out[7]:
```
[12, 34, 56, 32, 12, 3, 4, 57, 54, 46, 56, 78, 99, 100, 23]
```

In [4]:
```
import pandas as pd
```

In [5]:
```
# pd.Series(l1/t1/d1/s1)
ser1 = pd.Series(l1)
```

In [8]:
```
ser1
```

Out[8]:
```
0      12
1      34
2      56
3      32
4      12
5       3
6       4
7      57
8      54
9      46
10     56
11     78
12     99
13    100
14     23
dtype: int64
```

In [11]:
```
# indexes
# list/tuple/dict/set/pd.Series
# always from 0 to n-1

# for pd.Series, indexes can also be user defined

ser2 = pd.Series(l1, index= range(1,len(l1)+1))
# index takes a list as input -
#        i. elements should be unique
#        ii. the size of the list should be same as the size of the series
```

In [12]:
```
ser2
```

Out[12]:
```
1      12
2      34
3      56
4      32
5      12
6       3
7       4
8      57
9      54
10     46
11     56
12     78
13     99
14    100
15     23
dtype: int64
```

In [13]:
```
ser3 = pd.Series(l1, index = ['a','b','c','d','e','f','g','h','i','j','k','l',
'm','n','o'])
```

In [14]:
```
ser3
```

Out[14]:
```
a     12
b     34
c     56
d     32
e     12
f      3
g      4
h     57
i     54
j     46
k     56
l     78
m     99
n    100
o     23
dtype: int64
```

index to series can be # 1. Default 0 to n-1 - .iloc[] if user doesn't ginve any index, int and ext indexes are from 0 - n-1 # 2. User defined - .loc[] int - default index ext - user def index

In [15]:
```
l1[3]
```

Out[15]:
```
32
```

In [20]:
```
# ser2

# access fourth element - element with def index as 3

ser2.iloc[3]
```

Out[20]:
```
32
```

In [21]:
```
ser2.loc[4]
```

Out[21]: 32

In [22]:
```
ser3.loc["d"]
```

Out[22]: 32

In [23]:
```
ser3.iloc[3]
```

Out[23]: 32

In [26]:
```
ser1.iloc[0:8]
ser2.iloc[0:8]
```

Out[26]:
```
1    12
2    34
3    56
4    32
5    12
6     3
7     4
8    57
dtype: int64
```

In [28]:
```
ser2.loc[1:9] # Contrary to the regular 1,2,3,4,5,6,7,8 a colon op in .loc wil
l get 1,2,3,4,5,6,7,8,9
```

Out[28]:
```
1    12
2    34
3    56
4    32
5    12
6     3
7     4
8    57
9    54
dtype: int64
```

In [34]:
```
# Apply conditions to series
l1
# fetch elements > 30n and < 80

[x for x in l1 if x > 30 and x < 80] # list comprehension

res = []
for i in l1:
    if i > 30 and i < 80:
        res.append(i)
print res
```

```
[34, 56, 32, 57, 54, 46, 56, 78]
```

In [36]:
```
ser1 > 30
```

Out[36]:
```
0     False
1      True
2      True
3      True
4     False
5     False
6     False
7      True
8      True
9      True
10     True
11     True
12     True
13     True
14    False
dtype: bool
```

In [37]:
```
ser1[ser1 > 30]
```

Out[37]:
```
1      34
2      56
3      32
7      57
8      54
9      46
10     56
11     78
12     99
13    100
dtype: int64
```

In [39]:
```
ser1[(ser1 > 30) & (ser1 < 80)]

#and &
#or  |
#not -
```

Out[39]:
```
1     34
2     56
3     32
7     57
8     54
9     46
10    56
11    78
dtype: int64
```

In [41]:
```
ser1.loc[(ser1 > 30) & (ser1 < 80)]
```

Out[41]:
```
1     34
2     56
3     32
7     57
8     54
9     46
10    56
11    78
dtype: int64
```

In [44]:
```
# functions associated with series
print(dir(l1))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__del
slice__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '_
_getitem__', '__getslice__', '__gt__', '__hash__', '__iadd__', '__imul__', '_
_init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__
new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul_
_', '__setattr__', '__setitem__', '__setslice__', '__sizeof__', '__str__', '_
_subclasshook__', 'append', 'count', 'extend', 'index', 'insert', 'pop', 'rem
ove', 'reverse', 'sort']
```

In [45]:
```
print(dir(ser1))
```

```
['T', '_AXIS_ALIASES', '_AXIS_IALIASES', '_AXIS_LEN', '_AXIS_NAMES', '_AXIS_N
UMBERS', '_AXIS_ORDERS', '_AXIS_REVERSED', '_AXIS_SLICEMAP', '__abs__', '__ad
d__', '__and__', '__array__', '__array_prepare__', '__array_priority__', '__a
rray_wrap__', '__bool__', '__bytes__', '__class__', '__contains__', '__copy_
_', '__deepcopy__', '__delattr__', '__delitem__', '__dict__', '__dir__', '__d
iv__', '__divmod__', '__doc__', '__eq__', '__finalize__', '__float__', '__flo
ordiv__', '__format__', '__ge__', '__getattr__', '__getattribute__', '__getit
em__', '__getstate__', '__gt__', '__hash__', '__iadd__', '__iand__', '__idiv_
_', '__ifloordiv__', '__imod__', '__imul__', '__init__', '__int__', '__invert
__', '__ior__', '__ipow__', '__isub__', '__iter__', '__itruediv__', '__ixor_
_', '__le__', '__len__', '__long__', '__lt__', '__mod__', '__module__', '__mu
l__', '__ne__', '__neg__', '__new__', '__nonzero__', '__or__', '__pow__', '__
radd__', '__rand__', '__rdiv__', '__reduce__', '__reduce_ex__', '__repr__',
'__rfloordiv__', '__rmod__', '__rmul__', '__ror__', '__round__', '__rpow__',
'__rsub__', '__rtruediv__', '__rxor__', '__setattr__', '__setitem__', '__sets
tate__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__truediv_
_', '__unicode__', '__weakref__', '__xor__', '_accessors', '_add_numeric_oper
ations', '_add_series_only_operations', '_add_series_or_dataframe_operation
s', '_agg_by_level', '_agg_doc', '_aggregate', '_aggregate_multiple_funcs',
'_align_frame', '_align_series', '_allow_index_ops', '_at', '_binop', '_box_i
tem_values', '_builtin_table', '_can_hold_na', '_check_inplace_setting', '_ch
eck_is_chained_assignment_possible', '_check_percentile', '_check_setitem_cop
y', '_clear_item_cache', '_clip_with_one_bound', '_clip_with_scalar', '_conso
lidate', '_consolidate_inplace', '_construct_axes_dict', '_construct_axes_dic
t_for_slice', '_construct_axes_dict_from', '_construct_axes_from_arguments',
'_constructor', '_constructor_expanddim', '_constructor_sliced', '_convert',
'_create_indexer', '_cython_table', '_deprecations', '_dir_additions', '_dir_
deletions', '_drop_axis', '_expand_axes', '_formatting_values', '_from_axes',
'_get_axis', '_get_axis_name', '_get_axis_number', '_get_axis_resolvers', '_g
et_block_manager_axis', '_get_bool_data', '_get_cacher', '_get_index_resolver
s', '_get_item_cache', '_get_numeric_data', '_get_value', '_get_values', '_ge
t_values_tuple', '_get_with', '_gotitem', '_iat', '_iget_item_cache', '_ilo
c', '_index', '_indexed_same', '_info_axis', '_info_axis_name', '_info_axis_n
umber', '_init_mgr', '_internal_names', '_internal_names_set', '_is_builtin_f
unc', '_is_cached', '_is_cython_func', '_is_datelike_mixed_type', '_is_mixed_
type', '_is_numeric_mixed_type', '_is_view', '_ix', '_ixs', '_loc', '_maybe_c
ache_changed', '_maybe_update_cacher', '_metadata', '_needs_reindex_multi',
'_obj_with_exclusions', '_protect_consolidate', '_reduce', '_reindex_axes',
'_reindex_axis', '_reindex_indexer', '_reindex_multi', '_reindex_with_indexer
s', '_repr_data_resource_', '_repr_latex_', '_reset_cache', '_reset_cacher',
'_selected_obj', '_selection', '_selection_list', '_selection_name', '_set_as
_cached', '_set_axis', '_set_axis_name', '_set_is_copy', '_set_item', '_set_l
abels', '_set_name', '_set_subtyp', '_set_value', '_set_values', '_set_with',
'_set_with_engine', '_setup_axes', '_shallow_copy', '_slice', '_stat_axis',
'_stat_axis_name', '_stat_axis_number', '_take', '_to_dict_of_blocks', '_try_
aggregate_string_function', '_typ', '_unpickle_series_compat', '_update_inpla
ce', '_validate_dtype', '_values', '_where', '_xs', 'abs', 'add', 'add_prefi
x', 'add_suffix', 'agg', 'aggregate', 'align', 'all', 'any', 'append', 'appl
y', 'argmax', 'argmin', 'argsort', 'as_matrix', 'asfreq', 'asobject', 'asof',
'astype', 'at', 'at_time', 'autocorr', 'axes', 'base', 'between', 'between_ti
me', 'bfill', 'bool', 'clip', 'clip_lower', 'clip_upper', 'combine', 'combine
_first', 'compound', 'compress', 'copy', 'corr', 'count', 'cov', 'cummax', 'c
ummin', 'cumprod', 'cumsum', 'data', 'describe', 'diff', 'div', 'divide', 'do
t', 'drop', 'drop_duplicates', 'dropna', 'dtype', 'dtypes', 'duplicated', 'em
pty', 'eq', 'equals', 'ewm', 'expanding', 'factorize', 'ffill', 'fillna', 'fi
lter', 'first', 'first_valid_index', 'flags', 'floordiv', 'from_array', 'ftyp
e', 'ftypes', 'ge', 'get', 'get_dtype_counts', 'get_ftype_counts', 'get_value
```

```
s', 'groupby', 'gt', 'hasnans', 'head', 'hist', 'iat', 'idxmax', 'idxmin', 'i
loc', 'imag', 'index', 'infer_objects', 'interpolate', 'is_copy', 'is_monoton
ic', 'is_monotonic_decreasing', 'is_monotonic_increasing', 'is_unique', 'isi
n', 'isna', 'isnull', 'item', 'items', 'itemsize', 'iteritems', 'ix', 'keys',
'kurt', 'kurtosis', 'last', 'last_valid_index', 'le', 'loc', 'lt', 'mad', 'ma
p', 'mask', 'max', 'mean', 'median', 'memory_usage', 'min', 'mod', 'mode', 'm
ul', 'multiply', 'name', 'nbytes', 'ndim', 'ne', 'nlargest', 'nonzero', 'notn
a', 'notnull', 'nsmallest', 'nunique', 'pct_change', 'pipe', 'plot', 'pop',
'pow', 'prod', 'product', 'ptp', 'put', 'quantile', 'radd', 'rank', 'ravel',
'rdiv', 'real', 'reindex', 'reindex_axis', 'reindex_like', 'rename', 'rename_
axis', 'reorder_levels', 'repeat', 'replace', 'resample', 'reset_index', 'rfl
oordiv', 'rmod', 'rmul', 'rolling', 'round', 'rpow', 'rsub', 'rtruediv', 'sam
ple', 'searchsorted', 'select', 'sem', 'set_axis', 'shape', 'shift', 'size',
'skew', 'slice_shift', 'sort_index', 'sort_values', 'squeeze', 'std', 'stride
s', 'sub', 'subtract', 'sum', 'swapaxes', 'swaplevel', 'tail', 'take', 'to_cl
ipboard', 'to_csv', 'to_dense', 'to_dict', 'to_excel', 'to_frame', 'to_hdf',
'to_json', 'to_latex', 'to_msgpack', 'to_period', 'to_pickle', 'to_sparse',
'to_sql', 'to_string', 'to_timestamp', 'to_xarray', 'tolist', 'transform', 't
ranspose', 'truediv', 'truncate', 'tshift', 'tz_convert', 'tz_localize', 'uni
que', 'unstack', 'update', 'valid', 'value_counts', 'values', 'var', 'view',
'where', 'xs']
```

In [47]:
```
# sum mean quantiles/deciles median sd var skewness kurtosis max min
ser1.sum()
```

Out[47]: 666

In [48]:
```
ser1.mean()
```

Out[48]: 44.4

In [49]:
```
ser1.median()
```

Out[49]: 46.0

In [50]:
```
ser1.max()
```

Out[50]: 100

In [51]:
```
ser1.min()
```

Out[51]: 3

In [53]:
```
ser1.quantile([0,0.25,0.5,0.75,1])
```

Out[53]:
```
0.00      3.0
0.25     17.5
0.50     46.0
0.75     56.5
1.00    100.0
dtype: float64
```

In [56]:
```
ser1.astype(bool)
```

Out[56]:
```
0     True
1     True
2     True
3     True
4     True
5     True
6     True
7     True
8     True
9     True
10    True
11    True
12    True
13    True
14    True
dtype: bool
```

In [ ]:
```
1,2,4,6,8,10,1000,15,20
< 4

> 15
4,4,4,6,8,10,15,15,15
```

In [71]:
```
res = ser1.describe()
```

In [74]:
```
type(res)
```

Out[74]:
```
pandas.core.series.Series
```

In [ ]:
```
4,4,4,6,8,10,15,15,15

Y,N,N,Y,Y,Y,Y,N,N
```

In [58]:
```
ser1.head(3)
```

Out[58]:
```
0    12
1    34
2    56
dtype: int64
```

In [59]:
```
ser1.tail(3)
```

Out[59]:
```
12     99
13    100
14     23
dtype: int64
```

In [61]:
```
ser1.index
```

Out[61]:
```
RangeIndex(start=0, stop=15, step=1)
```

In [63]:
```
ser3.index.tolist()
```

Out[63]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o']
```

In [66]:
```
ser3.loc["a"]
#ser3.iloc["a"] # incorrect

ser3.ix["d"] # can accept both loc and iloc
```

Out[66]:
```
32
```

In [68]:
```
['to_hdf', 'to_json', 'to_latex', 'to_msgpack', 'to_period', 'to_pickle', 'to_
sparse', 'to_sql', 'to_string', 'to_timestamp', 'to_xarray', 'tolist']

ser1.tolist()
```

Out[68]:
```
[12L, 34L, 56L, 32L, 12L, 3L, 4L, 57L, 54L, 46L, 56L, 78L, 99L, 100L, 23L]
```

In [70]:
```
ser1.to_csv("text.csv")
```

In [75]:
```
# Working directory
import os
os.getcwd()
# either use \\ or a single /
```

Out[75]:
```
'C:\\Users\\admin\\pandas'
```

In [76]:
```
# setting a wd
os.chdir("C:/Users/admin/pandas/DataSets")
```

In [78]:
```
os.getcwd()
```

Out[78]:
```
'C:\\Users\\admin\\pandas\\DataSets'
```

In [79]:
```
# read a csv (stores.csv) from the DataSets folder
stores = pd.read_csv("stores.csv")
```

In [83]:
```
# read an excel file into pandas

Auto1 = pd.read_excel("AutoInsurance.xlsx",sheet = 1)
```

In [84]:
```
# read an excel file into pandas

Auto2 = pd.read_excel("AutoInsurance.xlsx",sheet = "AutoClaims")
```

In [85]:
```
# read SAS
# stores.csv -> stores.sas7bdat
stores_sas = pd.read_sas("stores.sas7bdat")
```

In [87]:
```
type(stores)
```

Out[87]:
```
pandas.core.frame.DataFrame
```

In [88]:
```
stores
```

Out[88]:

	StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales
0	STR101	Electronics Zone	Electronincs	Delhi	21.0	60	160.0
1	STR102	Apparel Zone	Apparel	Delhi	21.0	60	160.0
2	STR103	Super Bazar	Super Market	Delhi	22.8	40	108.0
3	STR104	Super Market	Super Market	Delhi	21.4	60	258.0
4	STR105	Central Store	Super Market	Delhi	18.7	80	360.0
5	STR106	Apparel Zone	Apparel	Delhi	18.1	60	225.0
6	STR107	Fashion Bazar	Apparel	Delhi	14.3	80	360.0
7	STR108	Digital Bazar	Electronincs	Delhi	24.4	40	146.7
8	STR109	Electronics Zone	Electronincs	Chennai	22.8	40	140.8
9	STR110	Apparel Zone	Apparel	Chennai	19.2	60	167.6
10	STR111	Super Bazar	Super Market	Chennai	17.8	60	167.6
11	STR112	Super Market	Super Market	Chennai	16.4	80	275.8
12	STR113	Central Store	Super Market	Chennai	17.3	80	275.8
13	STR114	Apparel Zone	Apparel	Chennai	15.2	80	275.8
14	STR115	Fashion Bazar	Apparel	Chennai	10.4	80	472.0
15	STR116	Digital Bazar	Electronincs	Chennai	10.4	80	460.0
16	STR117	Electronics Zone	Electronincs	Mumbai	14.7	80	440.0
17	STR118	Apparel Zone	Apparel	Mumbai	32.4	40	78.7
18	STR119	Super Bazar	Super Market	Mumbai	30.4	40	75.7

	StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales
19	STR120	Super Market	Super Market	Mumbai	33.9	40	71.1
20	STR121	Central Store	Super Market	Mumbai	21.5	40	120.1
21	STR122	Apparel Zone	Apparel	Mumbai	15.5	80	318.0
22	STR123	Fashion Bazar	Apparel	Mumbai	15.2	80	304.0
23	STR124	Digital Bazar	Electronincs	Mumbai	13.3	80	350.0
24	STR125	Electronics Zone	Electronincs	Kolkata	19.2	80	400.0
25	STR126	Apparel Zone	Apparel	Kolkata	27.3	40	79.0
26	STR127	Super Bazar	Super Market	Kolkata	26.0	40	120.3
27	STR128	Super Market	Super Market	Kolkata	30.4	40	95.1
28	STR129	Central Store	Super Market	Kolkata	15.8	80	351.0
29	STR130	Apparel Zone	Apparel	Kolkata	19.7	60	145.0
30	STR131	Fashion Bazar	Apparel	Kolkata	15.0	80	301.0
31	STR132	Digital Bazar	Electronincs	Kolkata	21.4	40	121.0

In [91]:
```
# get descriptive stats
stores.head(5) # first n rows
```

Out[91]:

	StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales	T
0	STR101	Electronics Zone	Electronincs	Delhi	21.0	60	160.0	1
1	STR102	Apparel Zone	Apparel	Delhi	21.0	60	160.0	1
2	STR103	Super Bazar	Super Market	Delhi	22.8	40	108.0	9
3	STR104	Super Market	Super Market	Delhi	21.4	60	258.0	1
4	STR105	Central Store	Super Market	Delhi	18.7	80	360.0	1

In [92]:
```
stores.tail(3)
```

Out[92]:

	StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales
29	STR130	Apparel Zone	Apparel	Kolkata	19.7	60	145.0
30	STR131	Fashion Bazar	Apparel	Kolkata	15.0	80	301.0
31	STR132	Digital Bazar	Electronincs	Kolkata	21.4	40	121.0

In [94]:
```
stores.columns.tolist()
```

Out[94]:
```
['StoreCode',
 'StoreName',
 'StoreType',
 'Location',
 'OperatingCost',
 'Staff_Cnt',
 'TotalSales',
 'Total_Customers',
 'AcqCostPercust',
 'BasketSize',
 'ProfitPercust',
 'OwnStore',
 'OnlinePresence',
 'Tenure',
 'StoreSegment']
```

In [95]:
```
stores.info()
# 1. Row - 32 entries with 0 - 31 and Data columns (total 15 columns)
# 2. All columns - followed by it's data types
        # object - strings
        # number of non missing values in that column
# 3. Freq table of the data types
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 15 columns):
StoreCode          32 non-null object
StoreName          32 non-null object
StoreType          32 non-null object
Location           32 non-null object
OperatingCost      32 non-null float64
Staff_Cnt          32 non-null int64
TotalSales         32 non-null float64
Total_Customers    32 non-null int64
AcqCostPercust     29 non-null float64
BasketSize         32 non-null float64
ProfitPercust      32 non-null float64
OwnStore           32 non-null int64
OnlinePresence     32 non-null int64
Tenure             32 non-null int64
StoreSegment       32 non-null int64
dtypes: float64(5), int64(6), object(4)
memory usage: 3.8+ KB
```

In [96]:
```
stores.describe()
```

Out[96]:

	OperatingCost	Staff_Cnt	TotalSales	Total_Customers	AcqCostPercust	Basket
count	32.000000	32.000000	32.000000	32.000000	29.000000	32.000
mean	20.090625	61.875000	230.721875	146.687500	3.651034	3.2172
std	6.026948	17.859216	123.938694	68.562868	0.532664	0.9784
min	10.400000	40.000000	71.100000	52.000000	2.760000	1.5130
25%	15.425000	40.000000	120.825000	96.500000	3.150000	2.5812
50%	19.200000	60.000000	196.300000	123.000000	3.730000	3.3250
75%	22.800000	80.000000	326.000000	180.000000	3.920000	3.6100
max	33.900000	80.000000	472.000000	335.000000	4.930000	5.4240

In [97]:
```
# I - Structural
```

In [99]:
```
# Try to extract columns

# 1. Use of the . operator
TS = stores.TotalSales
type(TS)
```

Out[99]:
```
pandas.core.series.Series
```

In [102]:
```
# 2. use a simple []
Case1 = stores["TotalSales"]

Case2 = stores[["Location","TotalSales","OperatingCost"]]
```

In [107]:
```
# 3. Use .loc[]  and .iloc[]
# For using loc and iloc, it's nesessary to pass both column and row


# [r,c] -> left of comma - row indexes/conditions and right is for column name
s

stores.loc[:,["Location","TotalSales","OperatingCost"]]
```

Out[107]:

	Location	TotalSales	OperatingCost
0	Delhi	160.0	21.0
1	Delhi	160.0	21.0
2	Delhi	108.0	22.8
3	Delhi	258.0	21.4
4	Delhi	360.0	18.7
5	Delhi	225.0	18.1
6	Delhi	360.0	14.3
7	Delhi	146.7	24.4
8	Chennai	140.8	22.8
9	Chennai	167.6	19.2
10	Chennai	167.6	17.8
11	Chennai	275.8	16.4
12	Chennai	275.8	17.3
13	Chennai	275.8	15.2
14	Chennai	472.0	10.4
15	Chennai	460.0	10.4
16	Mumbai	440.0	14.7
17	Mumbai	78.7	32.4
18	Mumbai	75.7	30.4
19	Mumbai	71.1	33.9
20	Mumbai	120.1	21.5
21	Mumbai	318.0	15.5
22	Mumbai	304.0	15.2
23	Mumbai	350.0	13.3
24	Kolkata	400.0	19.2
25	Kolkata	79.0	27.3
26	Kolkata	120.3	26.0
27	Kolkata	95.1	30.4
28	Kolkata	351.0	15.8
29	Kolkata	145.0	19.7
30	Kolkata	301.0	15.0
31	Kolkata	121.0	21.4

In [113]:
```
# Extract ["Location","TotalSales","OperatingCost"] first 10 rows

stores.loc[0:9,["Location","TotalSales","OperatingCost"]]
```

Out[113]:

	Location	TotalSales	OperatingCost
0	Delhi	160.0	21.0
1	Delhi	160.0	21.0
2	Delhi	108.0	22.8
3	Delhi	258.0	21.4
4	Delhi	360.0	18.7
5	Delhi	225.0	18.1
6	Delhi	360.0	14.3
7	Delhi	146.7	24.4
8	Chennai	140.8	22.8
9	Chennai	167.6	19.2

Data types str() object float() float64 long() int64 int() bool() bool dates datetime64 and datetime64[ns]

In [ ]:
```
#  pd.to_numeric(ser1)        any data type to numbers
#  pd.to_datetime()       suitable characters to date type
------------------------------------------------------------
# dataframe.astype(str)   anything to string (object type)
# dataframe.astype(bool)  anything to bool
```

In [118]:
```
TS = stores.TotalSales
TS.dtype
```

Out[118]: dtype('float64')

In [120]:
```
TS = TS.astype(str)
```

In [121]:
```
stores["TotalSales"] = stores.TotalSales.astype(str)
```

In [122]:
```
stores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 15 columns):
StoreCode          32 non-null object
StoreName          32 non-null object
StoreType          32 non-null object
Location           32 non-null object
OperatingCost      32 non-null float64
Staff_Cnt          32 non-null int64
TotalSales         32 non-null object
Total_Customers    32 non-null int64
AcqCostPercust     29 non-null float64
BasketSize         32 non-null float64
ProfitPercust      32 non-null float64
OwnStore           32 non-null int64
OnlinePresence     32 non-null int64
Tenure             32 non-null int64
StoreSegment       32 non-null int64
dtypes: float64(4), int64(6), object(5)
memory usage: 3.8+ KB
```

In [124]:
```
stores["TotalSales"] = pd.to_numeric(stores.TotalSales)
```

In [126]:
```
stores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 15 columns):
StoreCode          32 non-null object
StoreName          32 non-null object
StoreType          32 non-null object
Location           32 non-null object
OperatingCost      32 non-null float64
Staff_Cnt          32 non-null int64
TotalSales         32 non-null float64
Total_Customers    32 non-null int64
AcqCostPercust     29 non-null float64
BasketSize         32 non-null float64
ProfitPercust      32 non-null float64
OwnStore           32 non-null int64
OnlinePresence     32 non-null int64
Tenure             32 non-null int64
StoreSegment       32 non-null int64
dtypes: float64(5), int64(6), object(4)
memory usage: 3.8+ KB
```

In [127]:
```
# date

date1 = "25Feb2018"
date2 = "1/1/18"
date3 = "01/23/1986"
date4 = "Sunday 25 Feb 2018"
```

In [129]:
```
pd.to_datetime(date1,format="%d%b%Y")
```

Out[129]: Timestamp('2018-02-25 00:00:00')

In [131]:
```
d = pd.to_datetime(date3,format="%m/%d/%Y")
```

In [132]:
```
type(d)
```

Out[132]: pandas._libs.tslib.Timestamp