pandas-series-and-dataframe

April 18, 2023

1 Getting Started with Pandas

1.1 Pandas Series

A Pandas Series is like a column in a table.It is a one-dimensional array holding data of any type

[85]:
```
#importing pandas liabrary
import pandas as pd
```

1.1.1 Pandas Series can be created by two methods

- using list
- using dictionary

[86]:
```
# using list
l = [1,2,3,4,5,6]
pd.Series(l) # -> here we can observe that this one dimensional array also has␣
 ↪index section which is assigned to each element value in Series
```

[86]:
```
0    1
1    2
2    3
3    4
4    5
5    6
dtype: int64
```

[87]:
```
# using dictionary
d = {'a':1,'b':2,'c':3,'d':4}
pd.Series(d) # so here the keys become index and values of dictinary become␣
 ↪values of Series
```

[87]:
```
a    1
b    2
c    3
d    4
dtype: int64
```

[88]:
```
## one important note - all the values in the pandas series will be of similar datatype only
```

[89]:
```
# Also one can give custom index to series
order_id = ['1a23','1a24','1a25','1a27','1a30']
order_amount = [1000,2000,1500,10000,15000]
a = pd.Series(order_amount,index=order_id) # so by using index parameter we can specify custom indexes
a
```

[89]:
```
1a23     1000
1a24     2000
1a25     1500
1a27    10000
1a30    15000
dtype: int64
```

[90]:
```
# One can also give a name to the series
pd.Series([10,20,30,40,50,60,70,80],name="marks") # -> name represents attribute of that series
```

[90]:
```
0    10
1    20
2    30
3    40
4    50
5    60
6    70
7    80
Name: marks, dtype: int64
```

1.1.2 Series Attributes

1. Size

[91]:
```
s = pd.Series([10,20,30,40,50,60],name='sample')
s.size # tells about total no of elements in a series
```

[91]:
```
6
```

2. dtype

[92]:
```
s.dtype # it tells the data type of the elements of a particular series
```

[92]:
```
dtype('int64')
```

3. name

[93]:
```
s.name # tells the name given to that series
```

[93]:
```
'sample'
```

4. is_unique

[94]:
```
s.is_unique # tells whether all the elements in the series are unique or not␣
↪and it returns a boolean value
```

[94]:
```
True
```

5. index

[95]:
```
s.index # it gives all the indexes of a particular series . if indexes are␣
↪numerical then it will give RangeIndex object but if the indexes are
# categorical then it will give Index object
```

[95]:
```
RangeIndex(start=0, stop=6, step=1)
```

[96]:
```
a.index
```

[96]:
```
Index(['1a23', '1a24', '1a25', '1a27', '1a30'], dtype='object')
```

6. values

[97]:
```
s.values # -> make sure it returns a numpy array of the all the elements of a␣
↪particular series
```

[97]:
```
array([10, 20, 30, 40, 50, 60], dtype=int64)
```

1.2 Series through CSV

[98]:
```
# csv stands for comma separated values
```

[99]:
```
# if the csv file is with only one column then
pd.read_csv('scores.csv') # so by default it will read even single column csv␣
↪file as a DataFrame so to avoid this use squeeze parameter
```

[99]:
```
     scores
0        48
1        57
2        40
3        43
4        44
..      ...
360     231
361     226
362     155
```

```
363     144
364     172

[365 rows x 1 columns]
```

[100]:
```
scores = pd.read_csv('scores.csv',squeeze=True)
scores
```

```
C:\Users\malho\AppData\Local\Temp\ipykernel_1508\594718307.py:1: FutureWarning:
The squeeze argument has been deprecated and will be removed in a future
version. Append .squeeze("columns") to the call to squeeze.


  scores = pd.read_csv('scores.csv',squeeze=True)
```

[100]:
```
0       48
1       57
2       40
3       43
4       44
      ...
360    231
361    226
362    155
363    144
364    172
Name: scores, Length: 365, dtype: int64
```

[101]:
```
# with 2 columns
runs = pd.read_csv('batsman_runs_series.csv',index_col='batter',squeeze=True) #␣
 ↪we use index_col to make one column as index and then using
# squeeze=True to make it a series
runs
```

```
C:\Users\malho\AppData\Local\Temp\ipykernel_1508\1656446687.py:2: FutureWarning:
The squeeze argument has been deprecated and will be removed in a future
version. Append .squeeze("columns") to the call to squeeze.


  runs = pd.read_csv('batsman_runs_series.csv',index_col='batter',squeeze=True)
# we use index_col to make one column as index and then using
```

[101]:
```
batter
A Ashish Reddy     280
A Badoni           161
A Chandila           4
A Chopra            53
```

```
A Choudhary         25
                    …
Yash Dayal           0
Yashpal Singh       47
Younis Khan          3
Yuvraj Singh      2754
Z Khan             117
Name: batsman_run, Length: 605, dtype: int64
```

1.3 Series Methods

1. head()

[102]:
```
runs.head() #-> it displays top 5 values from top but to see specific no of␣
 ↪values from top enter the no as arguement in method
```

[102]:
```
batter
A Ashish Reddy    280
A Badoni          161
A Chandila          4
A Chopra           53
A Choudhary        25
Name: batsman_run, dtype: int64
```

[103]:
```
runs.head(10)
```

[103]:
```
batter
A Ashish Reddy    280
A Badoni          161
A Chandila          4
A Chopra           53
A Choudhary        25
A Dananjaya         4
A Flintoff         62
A Kumble           35
A Manohar         108
A Mishra          362
Name: batsman_run, dtype: int64
```

2. tail

[104]:
```
runs.tail() #-> it displays top 5 values from bottom but to see specific no of␣
 ↪values from bottom enter the no as arguement in method
```

[104]:
```
batter
Yash Dayal          0
Yashpal Singh      47
Younis Khan         3
```

```
Yuvraj Singh     2754
Z Khan            117
Name: batsman_run, dtype: int64
```

3. sample

[105]:
```
runs.sample() # it returns a random value from series by default but to get any␣
↪amout of random values enter the no as arguements in method
```

[105]:
```
batter
TM Srivastava    8
Name: batsman_run, dtype: int64
```

4. value_counts

[106]:
```
runs.value_counts() #it returns the frequency count of unique values and gives␣
↪frequencies in descending order
```

[106]:
```
0       23
2       16
1       13
4       13
3       12
        ..
247      1
2029     1
417      1
65       1
2754     1
Name: batsman_run, Length: 317, dtype: int64
```

5. sort_values

[107]:
```
runs.sort_values() # it sort the series based on values in ascending order
```

[107]:
```
batter
V Pratap Singh       0
Y Prithvi Raj        0
KR Sen               0
YA Abdulla           0
K Yadav              0
                   ...
SK Raina          5536
RG Sharma         5881
DA Warner         5883
S Dhawan          6244
V Kohli           6634
Name: batsman_run, Length: 605, dtype: int64
```

6. sort_index

[108]:
```
runs.sort_index() # it sorts the series based on index in ascending order
```

[108]:
```
batter
A Ashish Reddy     280
A Badoni           161
A Chandila           4
A Chopra            53
A Choudhary         25
                   ...
Yash Dayal           0
Yashpal Singh       47
Younis Khan          3
Yuvraj Singh      2754
Z Khan             117
Name: batsman_run, Length: 605, dtype: int64
```

1.4 Series Mathematical Methods

1.count()

[109]:
```
runs.count() # it tells the total no of elements in  a series but it doesn't␣
↪consier null values
```

[109]:
```
605
```

2.sum()

[110]:
```
runs.sum() #it adds up all the elements in a series
```

[110]:
```
280979
```

3.prod()

[111]:
```
runs.prod() # it calculates product of all the values in a series but the␣
↪values should be numerical
```

[111]:
```
0
```

4.mean()

[112]:
```
runs.mean() # it calculates the mean value for all the elements in a series but␣
↪the values should be numerical
```

[112]:
```
464.42809917355373
```

5.median()

[113]:
```
runs.median() # it calculates the median value for all the elements in a series␣
↪but the values should be numerical
```

[113]:
```
73.0
```

6.mode()

[114]:
```
runs.mode() ## it calculates the mode value for all the elements in a series
```

[114]:
```
0    0
Name: batsman_run, dtype: int64
```

7.std()

[115]:
```
runs.std()  # it gives standard deviation value for all the values in series␣
↪but the values should be numerical
```

[115]:
```
985.2728553757356
```

8.var()

[116]:
```
runs.var() # it gives variance of all the values in series but the values␣
↪should be numerical
```

[116]:
```
970762.5995402551
```

9.min()

[117]:
```
runs.min() # gives the minimum value from a series
```

[117]:
```
0
```

10.max()

[118]:
```
runs.max() # gives the maximum value from a series
```

[118]:
```
6634
```

11. describe()

[119]:
```
runs.describe() # gives a mathematical summary on a numerical series
```

[119]:
```
count    605.000000
mean     464.428099
std      985.272855
min        0.000000
25%       15.000000
50%       73.000000
75%      326.000000
```

```
max      6634.000000
Name: batsman_run, dtype: float64
```

1.5 Series Indexing

[120]:
```
# if the named indexes are numerical then we can do only positive(+) indexing␣
 ↪but not negative(-) idexing
runs[3]
```

[120]:
```
53
```

[121]:
```
runs[-29]
```

[121]:
```
552
```

[122]:
```
a = [8,7,6,5,4]
b = [10,20,30,40,50]
ans = pd.Series(b,index=a)
ans
```

[122]:
```
8    10
7    20
6    30
5    40
4    50
dtype: int64
```

[123]:
```
ans[-1] # See so in case of names numerical indexes we can do numerical indexes
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key,␣
 ↪method, tolerance)
   3801             try:
-> 3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.
 ↪IndexEngine.get_loc()

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.
 ↪IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTabl.
 ↪get_item()
```

```
pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTabl
 ↪get_item()

KeyError: -1

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1508\3361365256.py in <module>
----> 1 ans[-1] # See so in case of names numerical indexes we can do numerical
 ↪indexes

~\anaconda3\lib\site-packages\pandas\core\series.py in __getitem__(self, key)
    979
    980         elif key_is_scalar:
--> 981             return self._get_value(key)
    982
    983         if is_hashable(key):

~\anaconda3\lib\site-packages\pandas\core\series.py in _get_value(self, label,
 ↪takeable)
   1087
   1088         # Similar to Index.get_value, but we do not fall back to
 ↪positional
-> 1089         loc = self.index.get_loc(label)
   1090         return self.index._get_values_for_loc(self, loc, label)
   1091

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key,
 ↪method, tolerance)
   3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:
-> 3804                 raise KeyError(key) from err
   3805             except TypeError:
   3806                 # If we have a listlike key, _check_indexing_error will
 ↪raise

KeyError: -1
```

[124]:
```
# but if the indexes are string the both the indexing is possible positive and
 ↪negative
```

[125]:
```
runs[-1]
```

[125]: 117

[126]:
```
runs[2]
```

[126]: 4

1.6 Series Slicing

[127]:
```
#in case of slicing both the scenarios positively as well as negatively and␣
 ↪slicing works on index values
runs.head()
```

[127]:
```
batter
A Ashish Reddy    280
A Badoni          161
A Chandila          4
A Chopra           53
A Choudhary        25
Name: batsman_run, dtype: int64
```

[128]:
```
runs['A Ashish Reddy ':'A Chopra']
```

[128]:
```
batter
A Badoni      161
A Chandila      4
A Chopra       53
Name: batsman_run, dtype: int64
```

[129]:
```
runs[-6:-3]
```

[129]:
```
batter
YV Takawale      192
Yash Dayal         0
Yashpal Singh     47
Name: batsman_run, dtype: int64
```

[130]:
```
s[1:4]
```

[130]:
```
1    20
2    30
3    40
Name: sample, dtype: int64
```

[131]:
```
s[-4:-1]
```

[131]:
```
2    30
3    40
4    50
Name: sample, dtype: int64
```

1.6.1 Also one can edit series

[132]:
```
s.head()
```

[132]:
```
0    10
1    20
2    30
3    40
4    50
Name: sample, dtype: int64
```

[133]:
```
s[2] = 2000
```

[134]:
```
s.head()
```

[134]:
```
0      10
1      20
2    2000
3      40
4      50
Name: sample, dtype: int64
```

[135]:
```
s[1:4] = [200,300,400]
```

[136]:
```
s.head()
```

[136]:
```
0     10
1    200
2    300
3    400
4     50
Name: sample, dtype: int64
```

1.7 Series with python functionality

1. len()

[137]:
```
len(s) #gives length of series
```

[137]:
```
6
```

2. type()

[138]:
```
type(s) # tells type of variable s that is a series
```

[138]:
```
pandas.core.series.Series
```

3. dir()

[139]:
```
dir(s) # tells what s object contains
```

[139]:
```
['T',
 '_AXIS_LEN',
 '_AXIS_ORDERS',
 '_AXIS_TO_AXIS_NUMBER',
 '_HANDLED_TYPES',
 '__abs__',
 '__add__',
 '__and__',
 '__annotations__',
 '__array__',
 '__array_priority__',
 '__array_ufunc__',
 '__array_wrap__',
 '__bool__',
 '__class__',
 '__contains__',
 '__copy__',
 '__deepcopy__',
 '__delattr__',
 '__delitem__',
 '__dict__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__finalize__',
 '__float__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattr__',
 '__getattribute__',
 '__getitem__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__iand__',
 '__ifloordiv__',
 '__imod__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__int__',
 '__invert__',
```

```
'__ior__',
'__ipow__',
'__isub__',
'__iter__',
'__itruediv__',
'__ixor__',
'__le__',
'__len__',
'__long__',
'__lt__',
'__matmul__',
'__mod__',
'__module__',
'__mul__',
'__ne__',
'__neg__',
'__new__',
'__nonzero__',
'__or__',
'__pos__',
'__pow__',
'__radd__',
'__rand__',
'__rdivmod__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rfloordiv__',
'__rmatmul__',
'__rmod__',
'__rmul__',
'__ror__',
'__round__',
'__rpow__',
'__rsub__',
'__rtruediv__',
'__rxor__',
'__setattr__',
'__setitem__',
'__setstate__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__weakref__',
'__xor__',
```

```
'_accessors',
'_accum_func',
'_add_numeric_operations',
'_agg_by_level',
'_agg_examples_doc',
'_agg_see_also_doc',
'_align_frame',
'_align_series',
'_append',
'_arith_method',
'_as_manager',
'_attrs',
'_binop',
'_can_hold_na',
'_check_inplace_and_allows_duplicate_labels',
'_check_inplace_setting',
'_check_is_chained_assignment_possible',
'_check_label_or_level_ambiguity',
'_check_setitem_copy',
'_clear_item_cache',
'_clip_with_one_bound',
'_clip_with_scalar',
'_cmp_method',
'_consolidate',
'_consolidate_inplace',
'_construct_axes_dict',
'_construct_axes_from_arguments',
'_construct_result',
'_constructor',
'_constructor_expanddim',
'_convert',
'_convert_dtypes',
'_data',
'_dir_additions',
'_dir_deletions',
'_drop_axis',
'_drop_labels_or_levels',
'_duplicated',
'_find_valid_index',
'_flags',
'_get_axis',
'_get_axis_name',
'_get_axis_number',
'_get_axis_resolvers',
'_get_block_manager_axis',
'_get_bool_data',
'_get_cacher',
```

```
'_get_cleaned_column_resolvers',
'_get_index_resolvers',
'_get_label_or_level_values',
'_get_numeric_data',
'_get_value',
'_get_values',
'_get_values_tuple',
'_get_with',
'_gotitem',
'_hidden_attrs',
'_indexed_same',
'_info_axis',
'_info_axis_name',
'_info_axis_number',
'_init_dict',
'_init_mgr',
'_inplace_method',
'_internal_names',
'_internal_names_set',
'_is_cached',
'_is_copy',
'_is_label_or_level_reference',
'_is_label_reference',
'_is_level_reference',
'_is_mixed_type',
'_is_view',
'_item_cache',
'_ixs',
'_logical_func',
'_logical_method',
'_map_values',
'_maybe_update_cacher',
'_memory_usage',
'_metadata',
'_mgr',
'_min_count_stat_function',
'_name',
'_needs_reindex_multi',
'_protect_consolidate',
'_reduce',
'_reindex_axes',
'_reindex_indexer',
'_reindex_multi',
'_reindex_with_indexers',
'_rename',
'_replace_single',
'_repr_data_resource_',
```

```
'_repr_latex_',
'_reset_cache',
'_reset_cacher',
'_set_as_cached',
'_set_axis',
'_set_axis_name',
'_set_axis_nocheck',
'_set_is_copy',
'_set_labels',
'_set_name',
'_set_value',
'_set_values',
'_set_with',
'_set_with_engine',
'_slice',
'_stat_axis',
'_stat_axis_name',
'_stat_axis_number',
'_stat_function',
'_stat_function_ddof',
'_take',
'_take_with_is_copy',
'_typ',
'_update_inplace',
'_validate_dtype',
'_values',
'_where',
'abs',
'add',
'add_prefix',
'add_suffix',
'agg',
'aggregate',
'align',
'all',
'any',
'append',
'apply',
'argmax',
'argmin',
'argsort',
'array',
'asfreq',
'asof',
'astype',
'at',
'at_time',
```

```
'attrs',
'autocorr',
'axes',
'backfill',
'between',
'between_time',
'bfill',
'bool',
'clip',
'combine',
'combine_first',
'compare',
'convert_dtypes',
'copy',
'corr',
'count',
'cov',
'cummax',
'cummin',
'cumprod',
'cumsum',
'describe',
'diff',
'div',
'divide',
'divmod',
'dot',
'drop',
'drop_duplicates',
'droplevel',
'dropna',
'dtype',
'dtypes',
'duplicated',
'empty',
'eq',
'equals',
'ewm',
'expanding',
'explode',
'factorize',
'ffill',
'fillna',
'filter',
'first',
'first_valid_index',
'flags',
```

```
'floordiv',
'ge',
'get',
'groupby',
'gt',
'hasnans',
'head',
'hist',
'iat',
'idxmax',
'idxmin',
'iloc',
'index',
'infer_objects',
'info',
'interpolate',
'is_monotonic',
'is_monotonic_decreasing',
'is_monotonic_increasing',
'is_unique',
'isin',
'isna',
'isnull',
'item',
'items',
'iteritems',
'keys',
'kurt',
'kurtosis',
'last',
'last_valid_index',
'le',
'loc',
'lt',
'mad',
'map',
'mask',
'max',
'mean',
'median',
'memory_usage',
'min',
'mod',
'mode',
'mul',
'multiply',
'name',
```

```
'nbytes',
'ndim',
'ne',
'nlargest',
'notna',
'notnull',
'nsmallest',
'nunique',
'pad',
'pct_change',
'pipe',
'plot',
'pop',
'pow',
'prod',
'product',
'quantile',
'radd',
'rank',
'ravel',
'rdiv',
'rdivmod',
'reindex',
'reindex_like',
'rename',
'rename_axis',
'reorder_levels',
'repeat',
'replace',
'resample',
'reset_index',
'rfloordiv',
'rmod',
'rmul',
'rolling',
'round',
'rpow',
'rsub',
'rtruediv',
'sample',
'searchsorted',
'sem',
'set_axis',
'set_flags',
'shape',
'shift',
'size',
```

```
 'skew',
 'slice_shift',
 'sort_index',
 'sort_values',
 'squeeze',
 'std',
 'sub',
 'subtract',
 'sum',
 'swapaxes',
 'swaplevel',
 'tail',
 'take',
 'to_clipboard',
 'to_csv',
 'to_dict',
 'to_excel',
 'to_frame',
 'to_hdf',
 'to_json',
 'to_latex',
 'to_list',
 'to_markdown',
 'to_numpy',
 'to_period',
 'to_pickle',
 'to_sql',
 'to_string',
 'to_timestamp',
 'to_xarray',
 'transform',
 'transpose',
 'truediv',
 'truncate',
 'tz_convert',
 'tz_localize',
 'unique',
 'unstack',
 'update',
 'value_counts',
 'values',
 'var',
 'view',
 'where',
 'xs']
```

4. sorted()

[140]:
```
sorted(runs) #sorts the series
```

[140]:
```
[0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 2,
 2,
 2,
 2,
 2,
 2,
 2,
 2,
 2,
 2,
```

```
2,
2,
2,
2,
2,
2,
2,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
6,
6,
6,
6,
6,
6,
```

```
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
9,
9,
9,
9,
10,
10,
10,
10,
10,
10,
10,
10,
11,
11,
11,
11,
11,
11,
12,
12,
12,
12,
12,
12,
12,
12,
12,
12,
```

```
12,
13,
13,
13,
13,
13,
14,
14,
14,
15,
15,
15,
15,
15,
15,
15,
15,
15,
16,
16,
16,
17,
18,
18,
18,
18,
18,
18,
18,
18,
18,
18,
19,
19,
19,
19,
19,
19,
19,
19,
20,
20,
20,
20,
20,
20,
20,
21,
```

```
21,
21,
22,
22,
22,
22,
23,
23,
23,
23,
23,
23,
24,
24,
24,
24,
25,
25,
26,
26,
26,
26,
26,
26,
26,
26,
27,
28,
29,
29,
29,
31,
31,
31,
31,
31,
32,
32,
32,
33,
33,
33,
33,
34,
34,
34,
34,
```

```
35,
35,
35,
36,
36,
36,
36,
36,
36,
37,
37,
39,
39,
39,
39,
39,
39,
40,
40,
40,
41,
42,
42,
43,
43,
43,
44,
44,
45,
47,
49,
49,
49,
50,
50,
51,
51,
51,
51,
52,
52,
52,
53,
53,
53,
54,
55,
```

55,
56,
56,
58,
59,
61,
62,
63,
63,
64,
64,
65,
66,
66,
67,
67,
67,
69,
69,
69,
70,
73,
73,
73,
75,
75,
76,
76,
78,
78,
79,
79,
81,
81,
81,
82,
83,
85,
87,
87,
88,
88,
90,
91,
91,
91,
92,

92,
92,
96,
96,
98,
98,
99,
99,
103,
104,
105,
106,
106,
106,
106,
106,
108,
108,
111,
113,
115,
116,
117,
117,
117,
120,
120,
121,
121,
122,
123,
124,
125,
125,
126,
127,
127,
127,
127,
128,
129,
130,
131,
136,
140,
144,
145,
145,

147,
148,
159,
159,
161,
161,
161,
164,
164,
167,
167,
167,
169,
170,
173,
177,
177,
177,
179,
179,
180,
181,
181,
183,
186,
186,
186,
187,
187,
190,
192,
193,
193,
194,
195,
196,
198,
199,
203,
205,
205,
206,
217,
218,
228,
229,
230,

234,
237,
238,
241,
241,
247,
250,
251,
251,
252,
259,
267,
270,
270,
271,
278,
279,
280,
280,
282,
284,
285,
293,
295,
300,
302,
303,
304,
310,
313,
317,
318,
326,
327,
329,
337,
339,
340,
342,
362,
365,
368,
375,
379,
388,
390,
394,

```
397,
401,
404,
405,
409,
417,
423,
424,
460,
476,
503,
505,
506,
511,
514,
522,
527,
527,
531,
532,
538,
547,
549,
552,
577,
584,
604,
618,
647,
654,
663,
667,
672,
676,
688,
724,
731,
738,
739,
768,
795,
798,
831,
833,
880,
886,
910,
```

912,
920,
971,
974,
975,
985,
1000,
1001,
1017,
1025,
1070,
1073,
1079,
1107,
1135,
1150,
1153,
1196,
1207,
1237,
1260,
1291,
1322,
1326,
1329,
1349,
1400,
1406,
1441,
1494,
1496,
1554,
1560,
1588,
1687,
1692,
1695,
1798,
1808,
1870,
1900,
1972,
1977,
2029,
2039,
2069,
2092,

```
 2105,
 2174,
 2181,
 2320,
 2334,
 2335,
 2385,
 2427,
 2427,
 2455,
 2489,
 2495,
 2502,
 2619,
 2644,
 2728,
 2754,
 2767,
 2780,
 2832,
 2848,
 2851,
 2882,
 3222,
 3403,
 3437,
 3526,
 3657,
 3880,
 3895,
 4074,
 4190,
 4217,
 4377,
 4954,
 4978,
 4997,
 5181,
 5536,
 5881,
 5883,
 6244,
 6634]
```

5. max()

[141]:
```
max(runs) # gives maximum of a series
```

[141]: 6634

6. min()

[142]:
```
min(runs) #gives minimum of a series
```

[142]: 0

Membership operator

[143]:
```
# make sure by default membership operator works on index values but if one␣
↪wants to check in values the use "values" attribute
186 in runs.values
```

[143]: True

Arithmetic operators

[144]:
```
runs.head()
```

[144]:
```
batter
A Ashish Reddy    280
A Badoni          161
A Chandila          4
A Chopra           53
A Choudhary        25
Name: batsman_run, dtype: int64
```

[145]:
```
2 * runs
```

[145]:
```
batter
A Ashish Reddy     560
A Badoni           322
A Chandila           8
A Chopra           106
A Choudhary         50
                   ...
Yash Dayal           0
Yashpal Singh       94
Younis Khan          6
Yuvraj Singh      5508
Z Khan             234
Name: batsman_run, Length: 605, dtype: int64
```

Relational Operator

[146]:
```
100 > runs
```

[146]:
```
batter
A Ashish Reddy    False
A Badoni          False
A Chandila         True
A Chopra           True
A Choudhary        True
                  …
Yash Dayal         True
Yashpal Singh      True
Younis Khan        True
Yuvraj Singh      False
Z Khan            False
Name: batsman_run, Length: 605, dtype: bool
```

Boolean Indexing

[147]:
```
runs[100>runs]
```

[147]:
```
batter
A Chandila         4
A Chopra          53
A Choudhary       25
A Dananjaya        4
A Flintoff        62
                  ..
YA Abdulla         0
YS Chahal         37
Yash Dayal         0
Yashpal Singh     47
Younis Khan        3
Name: batsman_run, Length: 335, dtype: int64
```

2 Pandas DataFrame

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

2.0.1 creating a dataframe using list

[148]:
```
l = [[1,2,3,4],[5,6,7,8], [9,10,11,20]]
df = pd.DataFrame(l,columns=['a','b','c','d']) #now since these are unnamed␣
 ↪columns one can use columns parameter to give names to the columns
df
```

[148]:

	a	b	c	d
0	1	2	3	4
1	5	6	7	8
2	9	10	11	20

2.0.2 creating a dataframe using dictionary

[149]:
```
d = {'a':[1,2,3,4],'b':[5,6,7,8],'c':[9,10,11,12]}
sd = pd.DataFrame(d)
sd
```

[149]:

	a	b	c
0	1	5	9
1	2	6	10
2	3	7	11
3	4	8	12

2.0.3 DataFrame using csv file

[150]:
```
insurance = pd.read_csv('insurance_data - insurance_data.csv')
```

[151]:
```
insurance
```

[151]:

	index	PatientID	age	gender	bmi	bloodpressure	diabetic	children	\
0	0	1	39.0	male	23.2	91	Yes	0	
1	1	2	24.0	male	30.1	87	No	0	
2	2	3	NaN	male	33.3	82	Yes	0	
3	3	4	NaN	male	33.7	80	No	0	
4	4	5	NaN	male	34.1	100	No	0	
…	…	…	…	…	…	…	…	…	
1335	1335	1336	44.0	female	35.5	88	Yes	0	
1336	1336	1337	59.0	female	38.1	120	No	1	
1337	1337	1338	30.0	male	34.5	91	Yes	3	
1338	1338	1339	37.0	male	30.4	106	No	0	
1339	1339	1340	30.0	female	47.4	101	No	0	

	smoker	region	claim
0	No	southeast	1121.87
1	No	southeast	1131.51
2	No	southeast	1135.94
3	No	northwest	1136.40
4	No	northwest	1137.01
…	…	…	…
1335	Yes	northwest	55135.40
1336	Yes	northeast	58571.07
1337	Yes	northwest	60021.40

```
1338    Yes  southeast  62592.87
1339    Yes  southeast  63770.43

[1340 rows x 11 columns]
```

2.0.4 DataFrame attributes

1.shape

[152]:
```
insurance.shape # it tells the total no of rows and columns here 1340 is no of␣
↪rows and 11 is no of columns
```

[152]:
```
(1340, 11)
```

2.dtypes

[153]:
```
insurance.dtypes # tells the datatype of each column
```

[153]:
```
index              int64
PatientID          int64
age              float64
gender            object
bmi              float64
bloodpressure      int64
diabetic          object
children           int64
smoker            object
region            object
claim            float64
dtype: object
```

3.index

[154]:
```
insurance.index #-> gives index object
```

[154]:
```
RangeIndex(start=0, stop=1340, step=1)
```

4.columns

[155]:
```
#it gives all the columns of a dataframe in a list
insurance.columns
```

[155]:
```
Index(['index', 'PatientID', 'age', 'gender', 'bmi', 'bloodpressure',
       'diabetic', 'children', 'smoker', 'region', 'claim'],
      dtype='object')
```

5.values

[156]:
```
insurance.values # actually it gives a 2D array of values where each element in␣
↪a 2D array represents a row
```

[156]:
```
array([[0, 1, 39.0, …, 'No', 'southeast', 1121.87],
       [1, 2, 24.0, …, 'No', 'southeast', 1131.51],
       [2, 3, nan, …, 'No', 'southeast', 1135.94],
       …,
       [1337, 1338, 30.0, …, 'Yes', 'northwest', 60021.4],
       [1338, 1339, 37.0, …, 'Yes', 'southeast', 62592.87],
       [1339, 1340, 30.0, …, 'Yes', 'southeast', 63770.43]],
      dtype=object)
```

6.head()

[157]:
```
#-> it displays top 5 rows from top but to see specific no of rows from top␣
↪enter the no as arguement in method
insurance.head(2)
```

[157]:

	index	PatientID	age	gender	bmi	bloodpressure	diabetic	children	smoker	region	claim
0	0	1	39.0	male	23.2	91	Yes	0	No	southeast	1121.87
1	1	2	24.0	male	30.1	87	No	0	No	southeast	1131.51

7.tail()

[158]:
```
#-> it displays top 5 rows from bottom but to see specific no of rows from␣
↪bottom enter the no as arguement in method
insurance.tail(3)
```

[158]:

	index	PatientID	age	gender	bmi	bloodpressure	diabetic	children	smoker	region	claim
1337	1337	1338	30.0	male	34.5	91	Yes	3	Yes	northwest	60021.40
1338	1338	1339	37.0	male	30.4	106	No	0	Yes	southeast	62592.87
1339	1339	1340	30.0	female	47.4	101	No	0	Yes	southeast	63770.43

8.sample()

[159]:
```
insurance.sample(7)  # it returns a random rows from dataframe by default but␣
↪to get any amout of random rows
#enter the no as arguements in method
```

[159]:
```
      index  PatientID   age  gender   bmi  bloodpressure diabetic  children  \
468     468        469  38.0  female  27.4             81       No         1
765     765        766  26.0  female  32.7             90      Yes         0
321     321        322  43.0  female  31.4             99       No         4
418     418        419  28.0  female  43.3             90       No         2
1131   1131       1132  43.0  female  27.6            107       No         2
245     245        246  23.0    male  31.4             88       No         1
1036   1036       1037  38.0    male  29.8             87      Yes         0

     smoker     region     claim
468      No  southwest   6496.89
765      No  northeast  10923.93
321      No  southeast   4561.19
418      No  southeast   5846.92
1131    Yes  northwest  24535.70
245      No  southwest   3659.35
1036    Yes  northeast  18648.42
```

9.info()

[160]:
```
insurance.info() # iprovides a high level information about dataFrame/column␣
↪names/no-of-rows/missing-value-information/datatypes/memory-used
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1340 entries, 0 to 1339
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   index           1340 non-null   int64
 1   PatientID       1340 non-null   int64
 2   age             1335 non-null   float64
 3   gender          1340 non-null   object
 4   bmi             1340 non-null   float64
 5   bloodpressure   1340 non-null   int64
 6   diabetic        1340 non-null   object
 7   children        1340 non-null   int64
 8   smoker          1340 non-null   object
 9   region          1337 non-null   object
 10  claim           1340 non-null   float64
dtypes: float64(3), int64(4), object(4)
memory usage: 115.3+ KB
```

[161]:
```
insurance.describe() # gives a mathematical summary on all the numerical␣
↪columns by default
```

[161]:
```
              index    PatientID          age          bmi  bloodpressure  \
count  1340.000000  1340.000000  1335.000000  1340.000000    1340.000000
```

```
mean    669.500000   670.500000    38.078652    30.668955      94.157463
std     386.968991   386.968991    11.102924     6.106735      11.434712
min       0.000000     1.000000    18.000000    16.000000      80.000000
25%     334.750000   335.750000    29.000000    26.275000      86.000000
50%     669.500000   670.500000    38.000000    30.400000      92.000000
75%    1004.250000  1005.250000    47.000000    34.700000      99.000000
max    1339.000000  1340.000000    60.000000    53.100000     140.000000

          children         claim
count  1340.000000   1340.000000
mean      1.093284  13252.745642
std       1.205334  12109.609288
min       0.000000   1121.870000
25%       0.000000   4719.685000
50%       1.000000   9369.615000
75%       2.000000  16604.305000
max       5.000000  63770.430000
```

10.isnull in conjunction with sum()

[162]:
```
insurance.isnull().sum() # this code gives the total no of missing values in␣
↪each column
```

[162]:
```
index           0
PatientID       0
age             5
gender          0
bmi             0
bloodpressure   0
diabetic        0
children        0
smoker          0
region          3
claim           0
dtype: int64
```

11.duplicated() in conjunction with sum()

[164]:
```
insurance.duplicated().sum() # this code tells whether rows in a dataframe are␣
↪duplicated or not
```

[164]:
```
0
```

12.rename()

[165]:
```
insurance.columns
```

[165]:
```
Index(['index', 'PatientID', 'age', 'gender', 'bmi', 'bloodpressure',
       'diabetic', 'children', 'smoker', 'region', 'claim'],
      dtype='object')
```

[169]:
```
insurance.rename(columns={'age':'patient_age','bmi':'patient_bmi'}) # used to␣
 ↪rename columns but these are not permanent changes,
#to make it permanent use inplace=true
```

[169]:

	index	PatientID	patient_age	gender	patient_bmi	bloodpressure	\
0	0	1	39.0	male	23.2	91	
1	1	2	24.0	male	30.1	87	
2	2	3	NaN	male	33.3	82	
3	3	4	NaN	male	33.7	80	
4	4	5	NaN	male	34.1	100	
…	…	…	…	…	…	…	
1335	1335	1336	44.0	female	35.5	88	
1336	1336	1337	59.0	female	38.1	120	
1337	1337	1338	30.0	male	34.5	91	
1338	1338	1339	37.0	male	30.4	106	
1339	1339	1340	30.0	female	47.4	101	

	diabetic	children	smoker	region	claim
0	Yes	0	No	southeast	1121.87
1	No	0	No	southeast	1131.51
2	Yes	0	No	southeast	1135.94
3	No	0	No	northwest	1136.40
4	No	0	No	northwest	1137.01
…	…	…	…	…	…
1335	Yes	0	Yes	northwest	55135.40
1336	No	1	Yes	northeast	58571.07
1337	Yes	3	Yes	northwest	60021.40
1338	No	0	Yes	southeast	62592.87
1339	No	0	Yes	southeast	63770.43

[1340 rows x 11 columns]

13.sum()

[170]:
```
insurance.sum() # this code will do sum columnwise , obviously this seems not␣
 ↪to be very logical but can be useful sometimes
```

```
C:\Users\malho\AppData\Local\Temp\ipykernel_1508\2753369480.py:1: FutureWarning:
The default value of numeric_only in DataFrame.sum is deprecated. In a future
version, it will default to False. In addition, specifying 'numeric_only=None'
is deprecated. Select only valid columns or specify the value of numeric_only to
silence this warning.
  insurance.sum()
```

[170]:
```
index                                                      897130
PatientID                                                  898470
age                                                       50835.0
gender          malemalemalemalemalemalemalemalemalemalema…
bmi                                                       41096.4
bloodpressure                                              126171
diabetic        YesNoYesNoNoYesYesNoNoNoYesYesNoNoYesYesNoNoYe…
children                                                     1465
smoker          NoNoNoNoNoNoNoNoNoNoNoNoNoNoNoNoNoNoNoNoNoNo…
claim                                                17758679.16
dtype: object
```

[171]:
```
# also to sum row-wise one can use axis parameter
insurance.sum(axis=1)
```

```
C:\Users\malho\AppData\Local\Temp\ipykernel_1508\3021862499.py:2: FutureWarning:
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only valid
columns before calling the reduction.
  insurance.sum(axis=1)
```

[171]:
```
0        1276.07
1        1275.61
2        1256.24
3        1257.10
4        1280.11
          …
1335    57973.90
1336    61462.17
1337    62854.90
1338    65443.27
1339    66627.83
Length: 1340, dtype: float64
```

14.min()

[173]:
```
insurance.min() #
```

```
C:\Users\malho\AppData\Local\Temp\ipykernel_1508\1058992162.py:1: FutureWarning:
The default value of numeric_only in DataFrame.min is deprecated. In a future
version, it will default to False. In addition, specifying 'numeric_only=None'
is deprecated. Select only valid columns or specify the value of numeric_only to
silence this warning.
  insurance.min()
```

[173]:
```
index                 0
PatientID             1
age                18.0
```

```
gender            female
bmi                 16.0
bloodpressure         80
diabetic              No
children               0
smoker                No
claim            1121.87
dtype: object
```

15.max()

[ ]:
```
# for min and max functions these functions will work on strings on the basis␣
↪of ascii values
```

[174]:
```
insurance.max() # gives maximum value for each column
```

```
C:\Users\malho\AppData\Local\Temp\ipykernel_1508\2140407137.py:1: FutureWarning:
The default value of numeric_only in DataFrame.max is deprecated. In a future
version, it will default to False. In addition, specifying 'numeric_only=None'
is deprecated. Select only valid columns or specify the value of numeric_only to
silence this warning.
  insurance.max()
```

[174]:
```
index               1339
PatientID           1340
age                 60.0
gender              male
bmi                 53.1
bloodpressure        140
diabetic             Yes
children               5
smoker               Yes
claim            63770.43
dtype: object
```

16.mean()

[176]:
```
insurance.mean() # so this method will give mean for only numerical columns only
```

```
C:\Users\malho\AppData\Local\Temp\ipykernel_1508\997434782.py:1: FutureWarning:
The default value of numeric_only in DataFrame.mean is deprecated. In a future
version, it will default to False. In addition, specifying 'numeric_only=None'
is deprecated. Select only valid columns or specify the value of numeric_only to
silence this warning.
  insurance.mean() # so this method will give mean for only numerical values
```

[176]:
```
index             669.500000
PatientID         670.500000
```

```
age                  38.078652
bmi                  30.668955
bloodpressure        94.157463
children              1.093284
claim             13252.745642
dtype: float64
```

17.var()

[177]:
```
power = pd.read_csv('PowerGeneration - PowerGeneration.csv')
```

[178]:
```
power.head()
```

[178]:

	Dates	Power Station	Monitored Cap.(MW)	\
0	2017-09-01	Delhi	2235.4	
1	2017-09-01	Haryana	2720.0	
2	2017-09-01	Himachal Pradesh	3378.0	
3	2017-09-01	Jammu and Kashmir	1285.0	
4	2017-09-01	Punjab	3826.3	

	Total Cap. Under Maintenace (MW)	Planned Maintanence (MW)	\
0	135.00	0.00	
1	2470.00	0.00	
2	379.00	0.00	
3	150.00	0.00	
4	2697.65	77.65	

	Forced Maintanence(MW)	Other Reasons (MW)	Programme or Expected(MU)	\
0	135.0	0	13	
1	2470.0	0	28	
2	231.0	0	40	
3	0.0	0	14	
4	2620.0	0	39	

	Actual(MU)	Excess(+) / Shortfall (-)	Deviation
0	18	5.00	0.0
1	7	-21.80	0.0
2	46	5.63	0.0
3	23	9.43	0.0
4	17	-21.69	0.0

[179]:
```
power.var() # so this method will give variance for only numerical columns only
```

```
C:\Users\malho\AppData\Local\Temp\ipykernel_1508\34566881.py:1: FutureWarning:
The default value of numeric_only in DataFrame.var is deprecated. In a future
version, it will default to False. In addition, specifying 'numeric_only=None'
is deprecated. Select only valid columns or specify the value of numeric_only to
```

```
silence this warning.
  power.var()
```

[179]:
```
Monitored Cap.(MW)                  6.863863e+06
Total Cap. Under Maintenace (MW)    7.069308e+05
Planned Maintanence (MW)            4.168298e+04
Forced Maintanence(MW)              4.311783e+05
Other Reasons (MW)                  2.260050e+04
Programme or Expected(MU)           1.214681e+03
Actual(MU)                          1.467415e+03
Excess(+) / Shortfall (-)           9.443423e+01
Deviation                           1.245411e+03
dtype: float64
```

18.std()

[182]:
```
power.std() # so this method will give Standard Deviation for only numerical␣
 ↪columns only
```

```
C:\Users\malho\AppData\Local\Temp\ipykernel_1508\600398417.py:1: FutureWarning:
The default value of numeric_only in DataFrame.std is deprecated. In a future
version, it will default to False. In addition, specifying 'numeric_only=None'
is deprecated. Select only valid columns or specify the value of numeric_only to
silence this warning.
  power.std() # so this method will give Standard Deviation for only numerical
columns only
```

[182]:
```
Monitored Cap.(MW)                  2619.897504
Total Cap. Under Maintenace (MW)     840.791783
Planned Maintanence (MW)             204.164108
Forced Maintanence(MW)               656.641688
Other Reasons (MW)                   150.334633
Programme or Expected(MU)             34.852273
Actual(MU)                            38.306851
Excess(+) / Shortfall (-)              9.717728
Deviation                             35.290388
dtype: float64
```

2.1 Selecting columns from DataFrame

2.1.1 1. Single column

[183]:
```
power.columns # Selecting a single column will always give a series
```

[183]:
```
Index(['Dates', 'Power Station', 'Monitored Cap.(MW)',
       'Total Cap. Under Maintenace (MW)', 'Planned Maintanence (MW)',
       'Forced Maintanence(MW)', 'Other Reasons (MW)',
       'Programme or Expected(MU)', 'Actual(MU)', 'Excess(+) / Shortfall (-)',
```

```
       'Deviation'],
      dtype='object')
```

[184]:
```
power['Dates']
```

[184]:
```
0         2017-09-01
1         2017-09-01
2         2017-09-01
3         2017-09-01
4         2017-09-01
             …
345268    2022-04-13
345269    2022-04-13
345270    2022-04-13
345271    2022-04-13
345272    2022-04-13
Name: Dates, Length: 345273, dtype: object
```

[185]:
```
insurance['bmi']
```

[185]:
```
0       23.2
1       30.1
2       33.3
3       33.7
4       34.1
        …
1335    35.5
1336    38.1
1337    34.5
1338    30.4
1339    47.4
Name: bmi, Length: 1340, dtype: float64
```

2.1.2 2. Multiple columns

[186]:
```
insurance[['bmi','bloodpressure','age']] # always remember while selecting more than 1 column for indexing - because now now it becomes 2D
# to pass the column names in a 2D list just as seen in this code
```

[186]:
```
        bmi  bloodpressure   age
0     23.2             91  39.0
1     30.1             87  24.0
2     33.3             82   NaN
3     33.7             80   NaN
4     34.1            100   NaN
…     …              …    …
1335  35.5             88  44.0
```

```
1336  38.1            120  59.0
1337  34.5             91  30.0
1338  30.4            106  37.0
1339  47.4            101  30.0

[1340 rows x 3 columns]
```

[187]:
```
power[['Dates','Deviation']]
```

[187]:
```
             Dates  Deviation
0       2017-09-01        0.0
1       2017-09-01        0.0
2       2017-09-01        0.0
3       2017-09-01        0.0
4       2017-09-01        0.0
…              …        …
345268  2022-04-13        0.0
345269  2022-04-13        0.0
345270  2022-04-13        0.0
345271  2022-04-13        0.0
345272  2022-04-13        0.0

[345273 rows x 2 columns]
```

2.2 Selecting rows from DataFrame using Indexing/Slicing

2.2.1 1. Indexing

i. Using iloc

- this is used for internal indexes that are given by pandas liabrary

[189]:
```
play = pd.read_csv('googleplaystore.csv')
```

[190]:
```
play.head()
```

[190]:
```
                                                App        Category  Rating  \
0     Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN     4.1
1                                 Coloring book moana  ART_AND_DESIGN     3.9
2  U Launcher Lite - FREE Live Cool Themes, Hide …  ART_AND_DESIGN     4.7
3                                Sketch - Draw & Paint  ART_AND_DESIGN     4.5
4              Pixel Draw - Number Art Coloring Book  ART_AND_DESIGN     4.3

  Reviews  Size     Installs  Type Price Content Rating  \
0     159   19M      10,000+  Free     0       Everyone
1     967   14M     500,000+  Free     0       Everyone
2   87510  8.7M   5,000,000+  Free     0       Everyone
3  215644   25M  50,000,000+  Free     0           Teen
```

```
4     967  2.8M     100,000+  Free     0       Everyone

                      Genres      Last Updated         Current Ver  \
0               Art & Design   January 7, 2018               1.0.0
1  Art & Design;Pretend Play  January 15, 2018               2.0.0
2               Art & Design    August 1, 2018               1.2.4
3               Art & Design      June 8, 2018  Varies with device
4    Art & Design;Creativity     June 20, 2018                 1.1

    Android Ver
0  4.0.3 and up
1  4.0.3 and up
2  4.0.3 and up
3    4.2 and up
4    4.4 and up
```

Syntax = variablename.iloc[rowindex,columnindex]

[216]:
```
play.iloc[3 , 4] # always remember if we use simple indexing to fetch data it
 ↪will return a series in case of a single value and if values for both row
# and column is given then only a single value and in case of single value the
 ↪indexes will be columns and
# value will be values  of that particular row and column as given in the code
 ↪so here from 3rd row and 4th column the value is shown but if only  if
# a single value is given then it means that a single row and all the columns
```

[216]:
```
'50,000,000+'
```

[199]:
```
play.iloc[5]
```

[199]:
```
App                Paper flowers instructions
Category                         ART_AND_DESIGN
Rating                                      4.4
Reviews                                     167
Size                                       5.6M
Installs                                 50,000+
Type                                        Free
Price                                          0
Content Rating                          Everyone
Genres                             Art & Design
Last Updated                    March 26, 2017
Current Ver                                 1.0
Android Ver                       2.3 and up
Name: 5, dtype: object
```

ii. Using loc

- this is used for names indexes that are given by users

[210]:
```
play.set_index('App',inplace=True)
```

[213]:
```
play.index
```

[213]:
```
Index(['Photo Editor & Candy Camera & Grid & ScrapBook', 'Coloring book moana',
       'U Launcher Lite - FREE Live Cool Themes, Hide Apps',
       'Sketch - Draw & Paint', 'Pixel Draw - Number Art Coloring Book',
       'Paper flowers instructions', 'Smoke Effect Photo Maker - Smoke Editor',
       'Infinite Painter', 'Garden Coloring Book',
       'Kids Paint Free - Drawing Fun',
       …
       'payermonstationnement.fr', 'FR Tides', 'Chemin (fr)', 'FR Calculator',
       'FR Forms', 'Sya9a Maroc - FR', 'Fr. Mike Schmitz Audio Teachings',
       'Parkinson Exercices FR', 'The SCP Foundation DB fr nn5n',
       'iHoroscope - 2018 Daily Horoscope & Astrology'],
      dtype='object', name='App', length=10841)
```

[215]:
```
play.loc['Photo Editor & Candy Camera & Grid & ScrapBook','Size']
# always remember if we use simple indexing to fetch data it will return a␣
 ↪series in case of a single value and if values for both row
# and column is given then only a single value and in case of single value the␣
 ↪indexes will be columns and
# value will be values  of that particular row and column as given in the code␣
 ↪so here from 3rd row and 4th column the value is shown but if only  if
# a single value is given then it means that a single row and all the columns
```

[215]:
```
'19M'
```

[219]:
```
play.loc['Kids Paint Free - Drawing Fun']
```

[219]:
```
Category                   ART_AND_DESIGN
Rating                                4.7
Reviews                               121
Size                                 3.1M
Installs                         10,000+
Type                                Free
Price                                  0
Content Rating                   Everyone
Genres            Art & Design;Creativity
Last Updated                July 3, 2018
Current Ver                           2.8
Android Ver                 4.0.3 and up
Name: Kids Paint Free - Drawing Fun, dtype: object
```

2.2.2 2. Slicing

i. using iloc

[221]:
```
insurance.iloc[2:10,2:5] # it always return a dataframe, the value before comma is row slicing and the value after comma is column slicing . here
# slicing is same as that in python list
```

[221]:
```
    age gender   bmi
2   NaN   male  33.3
3   NaN   male  33.7
4   NaN   male  34.1
5   NaN   male  34.4
6   NaN   male  37.3
7  19.0   male  41.1
8  20.0   male  43.0
9  30.0   male  53.1
```

[225]:
```
play.head()
```

[225]:
```
                                                  Category  Rating  \
App
Photo Editor & Candy Camera & Grid & ScrapBook      ART_AND_DESIGN     4.1
Coloring book moana                                ART_AND_DESIGN     3.9
U Launcher Lite - FREE Live Cool Themes, Hide Apps  ART_AND_DESIGN     4.7
Sketch - Draw & Paint                              ART_AND_DESIGN     4.5
Pixel Draw - Number Art Coloring Book              ART_AND_DESIGN     4.3

                                                 Reviews  Size     Installs  \
App
Photo Editor & Candy Camera & Grid & ScrapBook       159   19M      10,000+
Coloring book moana                                  967   14M     500,000+
U Launcher Lite - FREE Live Cool Themes, Hide Apps  87510  8.7M   5,000,000+
Sketch - Draw & Paint                              215644   25M  50,000,000+
Pixel Draw - Number Art Coloring Book                967  2.8M     100,000+

                                                 Type Price Content Rating  \
App
Photo Editor & Candy Camera & Grid & ScrapBook      Free     0       Everyone
Coloring book moana                                Free     0       Everyone
U Launcher Lite - FREE Live Cool Themes, Hide Apps  Free     0       Everyone
Sketch - Draw & Paint                              Free     0           Teen
Pixel Draw - Number Art Coloring Book              Free     0       Everyone

                                                                      Genres  \
App
Photo Editor & Candy Camera & Grid & ScrapBook                   Art & Design
Coloring book moana                                Art & Design;Pretend Play
U Launcher Lite - FREE Live Cool Themes, Hide Apps               Art & Design
Sketch - Draw & Paint                                            Art & Design
```

```
Pixel Draw - Number Art Coloring Book                 Art & Design;Creativity

                                                        Last Updated  \
App
Photo Editor & Candy Camera & Grid & ScrapBook       January 7, 2018
Coloring book moana                                 January 15, 2018
U Launcher Lite - FREE Live Cool Themes, Hide Apps    August 1, 2018
Sketch - Draw & Paint                                   June 8, 2018
Pixel Draw - Number Art Coloring Book                  June 20, 2018

                                                           Current Ver  \
App
Photo Editor & Candy Camera & Grid & ScrapBook                   1.0.0
Coloring book moana                                              2.0.0
U Launcher Lite - FREE Live Cool Themes, Hide Apps                1.2.4
Sketch - Draw & Paint                                Varies with device
Pixel Draw - Number Art Coloring Book                              1.1

                                                      Android Ver
App
Photo Editor & Candy Camera & Grid & ScrapBook      4.0.3 and up
Coloring book moana                                 4.0.3 and up
U Launcher Lite - FREE Live Cool Themes, Hide Apps  4.0.3 and up
Sketch - Draw & Paint                                 4.2 and up
Pixel Draw - Number Art Coloring Book                 4.4 and up
```

ii. loc

[227]:
```
play.loc['Photo Editor & Candy Camera & Grid & ScrapBook':'Sketch - Draw &␣
 ↪Paint','Category':'Price'] # same as python list slicing and iloc slicing
# rather here we use named indexes
```

[227]:
```
                                                       Category  Rating  \
App
Photo Editor & Candy Camera & Grid & ScrapBook      ART_AND_DESIGN     4.1
Coloring book moana                                 ART_AND_DESIGN     3.9
U Launcher Lite - FREE Live Cool Themes, Hide Apps  ART_AND_DESIGN     4.7
Sketch - Draw & Paint                               ART_AND_DESIGN     4.5

                                                    Reviews  Size     Installs  \
App
Photo Editor & Candy Camera & Grid & ScrapBook          159   19M      10,000+
Coloring book moana                                     967   14M      500,000+
U Launcher Lite - FREE Live Cool Themes, Hide Apps     87510  8.7M   5,000,000+
Sketch - Draw & Paint                                215644   25M  50,000,000+

                                                    Type Price
```

```
App
Photo Editor & Candy Camera & Grid & ScrapBook      Free     0
Coloring book moana                                 Free     0
U Launcher Lite - FREE Live Cool Themes, Hide Apps  Free     0
Sketch - Draw & Paint                               Free     0
```

2.3 Using Fancy indexing

[223]:
```
iris = pd.read_csv('iris.csv')
```

[224]:
```
iris.head()
```

[224]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

[230]:
```
iris.iloc[[1,4,20],[1,4]] # here inside separate lists we can provide our own␣
↪indexes
```

[230]:

	SepalLengthCm	PetalWidthCm
1	4.9	0.2
4	5.0	0.2
20	5.4	0.2

[254]:
```
movies = pd.read_csv('movies.csv')
movies.set_index('title_x',inplace=True)
```

[255]:
```
movies.loc[['Why Cheat␣
↪India','Daaka','Humsafar'],['poster_path','wiki_link','title_y']] # here␣
↪inside separate lists we can provide our own indexes
```

[255]:

title_x	poster_path	wiki_link	title_y
Why Cheat India	https://upload.wikimedia.org/wikipedia/en/thum…	https://en.wikipedia.org/wiki/Why_Cheat_India	
Daaka	https://upload.wikimedia.org/wikipedia/en/thum…	https://en.wikipedia.org/wiki/Daaka	
Humsafar	https://upload.wikimedia.org/wikipedia/en/thum…	https://en.wikipedia.org/wiki/Humsafar	

```
title_x
Why Cheat India  Why Cheat India
Daaka                       Daaka
Humsafar                 Humsafar
```

2.4 Adding New Columns

[257]:
```
iris['country'] = 'India'
```

[258]:
```
iris.head() # just for sample and also we can make custom series and make a␣
↪column out of it in same way
```

[258]:
```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species  \
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa

  country
0   India
1   India
2   India
3   India
4   India
```

2.4.1 Pandas astype

- this is used to change the datatype of a column as it can store memory

[259]:
```
iris.info() # here species
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
 6   country        150 non-null    object
dtypes: float64(4), int64(1), object(2)
memory usage: 8.3+ KB
```

[267]:
```
iris['Species'] = iris['Species'].astype('category') # this si how we will have␣
↪to reassign the column to see changes
```

[268]:
```
iris.info() # now compare Species column above and here we have saved some space
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    category
dtypes: category(1), float64(4), int64(1)
memory usage: 6.3 KB
```