Pandas part - 01

What is Pandas?

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

- A fast and efficient DataFrame object for data manipulation with integrated indexing;
- Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent data alignment and integrated handling of missing data: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible reshaping and pivoting of data sets;
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets;
- Columns can be inserted and deleted from data structures for size mutability;
- Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets;
- High performance merging and joining of data sets;
- Hierarchical axis indexing provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
- Time series-functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;
- Highly optimized for performance, with critical code paths written in Cython or C.
- Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

Pandas Series

A Pandas Series is like a column in a table. It is a 1-D array holding data of any type.We can easily convert the list, tuple, and dictionary into series using "series' method. The row labels of series are called the index. A Series cannot contain multiple columns.

In [ ]:
```
# importing pandas and numpy
import pandas as pd
import numpy as np
```

Series from Lists

In [ ]:
```
# string
country = ['India', 'Pakistan', 'USA', 'Nepal', 'Srilanka']
pd.Series(country)
```

Out[ ]:
```
0       India
1    Pakistan
2         USA
3       Nepal
4    Srilanka
dtype: object
```

In [ ]:
```
# integer
runs = [31, 24, 26, 78, 100]
runs = pd.Series(runs)
runs
```

Out[ ]:
```
0     31
1     24
2     26
3     78
4    100
dtype: int64
```

In [ ]:
```
# custom Index
marks = [67, 57, 89, 100]
subjects = ['Maths', 'English', 'Science', 'Hindi']

pd.Series(marks, index=subjects)
```

Out[ ]:
```
Maths       67
English     57
Science     89
Hindi      100
dtype: int64
```

In [ ]:
```
# setting a name
pd.Series(marks, index=subjects, name='Dhanraj marks')
```

Out[ ]:
```
Maths       67
English     57
Science     89
Hindi      100
Name: Dhanraj marks, dtype: int64
```

In [ ]:
```
marks = pd.Series(marks, index=subjects, name='Dhanraj marks')
marks
```

Out[ ]:
```
Maths       67
English     57
Science     89
Hindi      100
Name: Dhanraj marks, dtype: int64
```

Series from dictionary

In [ ]:
```
marks = {'maths': 67,
         'english': 57,
         'science': 89,
         'hindi': 100}

marks = pd.Series(marks, name='Dhanraj Marks')
marks
```

Out[ ]:
```
maths       67
english     57
science     89
hindi      100
Name: Dhanraj Marks, dtype: int64
```

Series Attributes

In [ ]:
```
# size
marks.size
```

Out[ ]:
```
4
```

In [ ]:
```
# dtype
marks.dtype
```

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

In [ ]:
```
# name
marks.name
```

Out[ ]:
```
'Dhanraj Marks'
```

In [ ]:
```
# is_unique
pd.Series([1, 1, 2, 3, 4, 5]).is_unique  # 1 repeating
```

Out[ ]:
```
False
```

In [ ]:
```
marks.is_unique  # no repeating value
```

Out[ ]:
```
True
```

In [ ]:
```
# index
marks.index
```

Out[ ]:
```
Index(['maths', 'english', 'science', 'hindi'], dtype='object')
```

In [ ]:
```
runs.index
# range index
```

Out[ ]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [ ]:
```
# values
print(marks)
marks.values
```

Out[ ]:
```
maths       67
english     57
science     89
hindi      100
Name: Dhanraj Marks, dtype: int64
array([ 67,  57,  89, 100], dtype=int64)
```

Series using read_csv (Comma Separated Values)

DataSets

1. bollywood.csv
2. kohli_ipl.csv
3. subs.csv

In [ ]:
```
# with one col
pd.read_csv('subs.csv')
```

Out[ ]:

	Subscribers gained
0	48
1	57
2	40
3	43
4	44
...	...
360	231
361	226
362	155
363	144
364	172

365 rows × 1 columns

In [ ]:
```
# pd.read_csv() -> default store in dataframe not in series
type(pd.read_csv('subs.csv'))
```

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

In [ ]:
```
# convert into series
pd.read_csv('subs.csv', squeeze=True)
```

```
C:\Users\dhanr\AppData\Local\Temp\ipykernel_19676\2376045988.py:2: FutureWarning: The sq
ueeze argument has been deprecated and will be removed in a future version. Append .sque
eze("columns") to the call to squeeze.


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

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

In [ ]:
```
type(pd.read_csv('subs.csv', squeeze=True))
```

```
C:\Users\dhanr\AppData\Local\Temp\ipykernel_19676\2422685784.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.


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

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

In [ ]:
```
subs = pd.read_csv('subs.csv', squeeze=True)
subs
```

```
C:\Users\dhanr\AppData\Local\Temp\ipykernel_19676\2976050358.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.


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

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

In [ ]:
```
# with 2 col
vk = pd.read_csv('kohli_ipl.csv', index_col='match_no',
                 squeeze=True)  # index generate from column
vk
```

```
C:\Users\dhanr\AppData\Local\Temp\ipykernel_19676\4053684352.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.


  vk = pd.read_csv('kohli_ipl.csv', index_col='match_no',
```

Out[ ]:
```
match_no
1       1
2      23
3      13
4      12
5       1
       ..
211     0
212    20
213    73
214    25
215     7
Name: runs, Length: 215, dtype: int64
```

In [ ]:
```
# example
movies = pd.read_csv('bollywood.csv', index_col='movie', squeeze=True)
movies
```

```
C:\Users\dhanr\AppData\Local\Temp\ipykernel_19676\42821923.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.


  movies = pd.read_csv('bollywood.csv', index_col='movie', squeeze=True)
```

Out[ ]:
```
movie
Uri: The Surgical Strike                   Vicky Kaushal
Battalion 609                                Vicky Ahuja
The Accidental Prime Minister (film)         Anupam Kher
Why Cheat India                            Emraan Hashmi
Evening Shadows                         Mona Ambegaonkar
                                                   ...
Hum Tumhare Hain Sanam                    Shah Rukh Khan
Aankhen (2002 film)                     Amitabh Bachchan
Saathiya (film)                             Vivek Oberoi
Company (film)                                Ajay Devgn
Awara Paagal Deewana                        Akshay Kumar
Name: lead, Length: 1500, dtype: object
```

Series Methods

In [ ]:
```
# head and tail
subs.head()  # default -> first five rows
```

Out[ ]:
```
0    48
1    57
2    40
3    43
4    44
Name: Subscribers gained, dtype: int64
```

In [ ]:
```
subs.head(3)  # first 3 rows
```

Out[ ]:
```
0    48
1    57
2    40
Name: Subscribers gained, dtype: int64
```

In [ ]:
```
# tail
subs.tail()  # default -> last 5 rows
```

Out[ ]:
```
360    231
361    226
362    155
363    144
364    172
Name: Subscribers gained, dtype: int64
```

In [ ]:
```
subs.tail(10)  # last 10 rows
```

Out[ ]:
```
355    149
356    156
357    177
358    210
359    209
360    231
361    226
362    155
363    144
364    172
Name: Subscribers gained, dtype: int64
```

In [ ]:
```
# sample -> 1 row randomly
subs.sample()
```

Out[ ]:
```
317    183
Name: Subscribers gained, dtype: int64
```

In [ ]:
```
subs.sample()
```

Out[ ]:
```
155    101
Name: Subscribers gained, dtype: int64
```

In [ ]:
```
# randomly five rows
subs.sample(5)
```

Out[ ]:
```
145    100
294    190
323    196
359    209
362    155
Name: Subscribers gained, dtype: int64
```

In [ ]:
```
# value_counts()
movies.value_counts()
```

Out[ ]:
```
Akshay Kumar        48
Amitabh Bachchan    45
Ajay Devgn          38
Salman Khan         31
Sanjay Dutt         26
                    ..
Diganth              1
Parveen Kaur         1
Seema Azmi           1
Akanksha Puri        1
Edwin Fernandes      1
Name: lead, Length: 566, dtype: int64
```

In [ ]:
```
# sort_values -> temporary sort
vk.sort_values()  # sort in ascending order
```

Out[ ]:
```
match_no
87       0
211      0
207      0
206      0
91       0
      ...
164    100
120    100
123    108
126    109
128    113
Name: runs, Length: 215, dtype: int64
```

In [ ]:
```
# descending order
vk.sort_values(ascending=False)
```

Out[ ]:
```
match_no
128    113
126    109
123    108
164    100
120    100
       ...
93       0
211      0
130      0
8        0
135      0
Name: runs, Length: 215, dtype: int64
```

In [ ]:
```
# top descending row
vk.sort_values(ascending=False).head(1)
```

Out[ ]:
```
match_no
128    113
Name: runs, dtype: int64
```

In [ ]:
```
# print top descending row value
vk.sort_values(ascending=False).head(1).values
```

Out[ ]:
```
array([113], dtype=int64)
```

In [ ]:
```
# without numpy array
vk.sort_values(ascending=False).head(1).values[0]
```

Out[ ]:
```
113
```

In [ ]:
```
# permanent sort -> inplace = True
vk.sort_values(inplace=True)
```

In [ ]:
```
# sorted dataset
vk
```

Out[ ]:
```
match_no
87       0
211      0
207      0
206      0
91       0
       ...
164    100
120    100
123    108
126    109
128    113
Name: runs, Length: 215, dtype: int64
```

In [ ]:
```
# sort index ->temporary index
movies
```

Out[ ]:
```
movie
Uri: The Surgical Strike                   Vicky Kaushal
Battalion 609                                Vicky Ahuja
The Accidental Prime Minister (film)         Anupam Kher
Why Cheat India                            Emraan Hashmi
Evening Shadows                         Mona Ambegaonkar
                                                    ...
Hum Tumhare Hain Sanam                    Shah Rukh Khan
Aankhen (2002 film)                      Amitabh Bachchan
Saathiya (film)                              Vivek Oberoi
Company (film)                               Ajay Devgn
Awara Paagal Deewana                        Akshay Kumar
Name: lead, Length: 1500, dtype: object
```

In [ ]:
```
movies.sort_index()
```

Out[ ]:
```
movie
1920 (film)                     Rajniesh Duggall
1920: London                       Sharman Joshi
1920: The Evil Returns               Vicky Ahuja
1971 (2007 film)                   Manoj Bajpayee
2 States (2014 film)                 Arjun Kapoor
                                          ...
Zindagi 50-50                        Veena Malik
Zindagi Na Milegi Dobara          Hrithik Roshan
Zindagi Tere Naam            Mithun Chakraborty
Zokkomon                        Darsheel Safary
Zor Lagaa Ke...Haiya!              Meghan Jadhav
Name: lead, Length: 1500, dtype: object
```

In [ ]:
```
# permanent sort
movies.sort_index(inplace=True)
```

In [ ]:
```
movies
```

Out[ ]:
```
movie
1920 (film)                     Rajniesh Duggall
1920: London                       Sharman Joshi
1920: The Evil Returns               Vicky Ahuja
1971 (2007 film)                   Manoj Bajpayee
2 States (2014 film)                 Arjun Kapoor
                                          ...
Zindagi 50-50                        Veena Malik
Zindagi Na Milegi Dobara          Hrithik Roshan
Zindagi Tere Naam            Mithun Chakraborty
Zokkomon                        Darsheel Safary
Zor Lagaa Ke...Haiya!              Meghan Jadhav
Name: lead, Length: 1500, dtype: object
```

Series Maths Methods

In [ ]:
```
# count -> missing value not count
vk.count()
```

Out[ ]:
```
215
```

In [ ]:
```
# sum
subs.sum()
```

Out[ ]:
```
49510
```

In [ ]:
```
# mean
```

Out[ ]:
```
135.64383561643837
```

In [ ]:
```
# median
vk.median()
```

Out[ ]:
```
24.0
```

In [ ]:
```
# mode
movies.mode()
```

Out[ ]:
```
0    Akshay Kumar
Name: lead, dtype: object
```

In [ ]:
```
# std (standard deviation)
subs.std()
```

Out[ ]:
```
62.67502303725269
```

In [ ]:
```
# var (variance)
subs.var()
```

Out[ ]:
```
3928.1585127201556
```

In [ ]:
```
# min
subs.min()
```

Out[ ]:
```
33
```

In [ ]:
```
# max
subs.max()
```

Out[ ]:
```
396
```

In [ ]:
```
# describe  -> summary of mathematical quantities
vk.describe()
```

Out[ ]:
```
count    215.000000
mean      30.855814
std       26.229801
min        0.000000
25%        9.000000
50%       24.000000
75%       48.000000
max      113.000000
Name: runs, dtype: float64
```

Series Indexing

In [ ]:
```
x = pd.Series([12, 13, 14, 35, 46, 57, 58, 79, 9])
x[1]
```

Out[ ]:
```
13
```

In [ ]:
```
movies
```

Out[ ]:
```
movie
1920 (film)                  Rajniesh Duggall
1920: London                    Sharman Joshi
1920: The Evil Returns            Vicky Ahuja
1971 (2007 film)                Manoj Bajpayee
2 States (2014 film)              Arjun Kapoor
                                          ...
Zindagi 50-50                     Veena Malik
Zindagi Na Milegi Dobara        Hrithik Roshan
Zindagi Tere Naam           Mithun Chakraborty
Zokkomon                       Darsheel Safary
Zor Lagaa Ke...Haiya!            Meghan Jadhav
Name: lead, Length: 1500, dtype: object
```

In [ ]:
```
# fetch value
movies[0]
```

Out[ ]:
```
'Rajniesh Duggall'
```

In [ ]:
```
# another method for fetch value
movies['1920 (film)']
```

Out[ ]:
```
'Rajniesh Duggall'
```

In [ ]:
```
# slicing -> runs from 5th match to 16th math
vk[5:16]
```

Out[ ]:
```
match_no
93     0
8      0
130    0
135    0
106    1
113    1
77     1
75     1
1      1
174    1
5      1
Name: runs, dtype: int64
```

In [ ]:
```
# negative sllicing -> last 5 movies
vk[-5:]
```

Out[ ]:
```
match_no
164    100
120    100
123    108
126    109
128    113
Name: runs, dtype: int64
```

In [ ]:
```
movies[-5:]
```

Out[ ]:
```
movie
Zindagi 50-50                     Veena Malik
Zindagi Na Milegi Dobara        Hrithik Roshan
Zindagi Tere Naam           Mithun Chakraborty
Zokkomon                       Darsheel Safary
Zor Lagaa Ke...Haiya!            Meghan Jadhav
Name: lead, dtype: object
```

In [ ]:
```
# skip 1 row
movies[::2]
```

Out[ ]:
```
movie
1920 (film)                 Rajniesh Duggall
1920: The Evil Returns           Vicky Ahuja
2 States (2014 film)           Arjun Kapoor
3 A.M. (2014 film)            Salil Acharya
3 Idiots                         Aamir Khan
                                     ...
Zero (2018 film)             Shah Rukh Khan
Zila Ghaziabad                 Vivek Oberoi
Zindaggi Rocks                 Sushmita Sen
Zindagi Na Milegi Dobara      Hrithik Roshan
Zokkomon                     Darsheel Safary
Name: lead, Length: 750, dtype: object
```

In [ ]:
```
# fancy indexing
vk[[1, 3, 4, 5]]
```

Out[ ]:
```
match_no
1     1
3    13
4    12
5     1
Name: runs, dtype: int64
```

Editing Series

In [ ]:
```
# using indexing
marks
```

Out[ ]:
```
maths       67
english     57
science     89
hindi      100
Name: Dhanraj Marks, dtype: int64
```

In [ ]:
```
marks[1] = 90
marks
```

Out[ ]:
```
maths       67
english     90
science     89
hindi      100
Name: Dhanraj Marks, dtype: int64
```

In [ ]:
```
# what if an index does not exist
marks['sst'] = 75
marks
```

Out[ ]:
```
maths       67
english     90
science     89
hindi      100
sst         75
Name: Dhanraj Marks, dtype: int64
```

In [ ]:
```
# slicing
runs
```

Out[ ]:
```
0     31
1     24
2     26
3     78
4    100
dtype: int64
```

In [ ]:
```
runs[2:4] = [100, 100]
runs
```

Out[ ]:
```
0     31
1     24
2    100
3    100
4    100
dtype: int64
```

In [ ]:
```
# fancy indexing
runs[[0, 3, 4]] = [0, 0, 0]
runs
```

Out[ ]:
```
0      0
1     24
2    100
3      0
4      0
dtype: int64
```

In [ ]:
```
# example
movies
```

Out[ ]:
```
movie
1920 (film)                     Rajniesh Duggall
1920: London                       Sharman Joshi
1920: The Evil Returns               Vicky Ahuja
1971 (2007 film)                  Manoj Bajpayee
2 States (2014 film)                Arjun Kapoor
                                             ...
Zindagi 50-50                        Veena Malik
Zindagi Na Milegi Dobara          Hrithik Roshan
Zindagi Tere Naam           Mithun Chakraborty
Zokkomon                       Darsheel Safary
Zor Lagaa Ke...Haiya!            Meghan Jadhav
Name: lead, Length: 1500, dtype: object
```

In [ ]:
```
movies['2 States (2014 film)'] = 'aliya bhatt'
movies
```

Out[ ]:
```
movie
1920 (film)                     Rajniesh Duggall
1920: London                       Sharman Joshi
1920: The Evil Returns               Vicky Ahuja
1971 (2007 film)                  Manoj Bajpayee
2 States (2014 film)                 aliya bhatt
                                             ...
Zindagi 50-50                        Veena Malik
Zindagi Na Milegi Dobara          Hrithik Roshan
Zindagi Tere Naam           Mithun Chakraborty
Zokkomon                       Darsheel Safary
Zor Lagaa Ke...Haiya!            Meghan Jadhav
Name: lead, Length: 1500, dtype: object
```

Series with Python Functionalities

In [ ]:
```
# len
print('len:', len(subs))
print('type:', type(subs))
print('dir:', dir(subs))
print('sorted:', sorted(subs))
print('min:', min(subs))
max(subs))
```

```
len: 365
type: <class 'pandas.core.series.Series'>
dir: ['T', '_AXIS_LEN', '_AXIS_ORDERS', '_AXIS_TO_AXIS_NUMBER', '_HANDLED_TYPES', '__abs
__', '__add__', '__and__', '__annotations__', '__array__', '__array_priority__', '__arra
y_ufunc__', '__array_wrap__', '__bool__', '__class__', '__contains__', '__copy__', '__de
epcopy__', '__delattr__', '__delitem__', '__dict__', '__dir__', '__divmod__', '__doc__',
'__eq__', '__finalize__', '__float__', '__floordiv__', '__format__', '__ge__', '__getatt
r__', '__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__', '__m
odule__', '__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__', '__setsta
te__', '__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_al
lows_duplicate_labels', '_check_inplace_setting', '_check_is_chained_assignment_possibl
e', '_check_label_or_level_ambiguity', '_check_setitem_copy', '_clear_item_cache', '_cli
p_with_one_bound', '_clip_with_scalar', '_cmp_method', '_consolidate', '_consolidate_inp
lace', '_construct_axes_dict', '_construct_axes_from_arguments', '_construct_result', '_
constructor', '_constructor_expanddim', '_convert', '_convert_dtypes', '_data', '_dir_ad
ditions', '_dir_deletions', '_drop_axis', '_drop_labels_or_levels', '_duplicated', '_fin
d_valid_index', '_flags', '_from_mgr', '_get_axis', '_get_axis_name', '_get_axis_numbe
r', '_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', '_go
titem', '_hidden_attrs', '_indexed_same', '_info_axis', '_info_axis_name', '_info_axis_n
umber', '_init_dict', '_init_mgr', '_inplace_method', '_internal_names', '_internal_name
s_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_fu
nc', '_logical_method', '_map_values', '_maybe_update_cacher', '_memory_usage', '_metada
ta', '_mgr', '_min_count_stat_function', '_name', '_needs_reindex_multi', '_protect_cons
olidate', '_reduce', '_reindex_axes', '_reindex_indexer', '_reindex_multi', '_reindex_wi
th_indexers', '_rename', '_replace_single', '_repr_data_resource_', '_repr_latex_', '_re
set_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', '_se
t_with', '_set_with_engine', '_slice', '_stat_axis', '_stat_axis_name', '_stat_axis_numb
er', '_stat_function', '_stat_function_ddof', '_take_with_is_copy', '_typ', '_update_inp
lace', '_validate_dtype', '_values', '_where', 'abs', 'add', 'add_prefix', 'add_suffix',
'agg', 'aggregate', 'align', 'all', 'any', 'append', 'apply', 'argmax', 'argmin', 'argso
rt', 'array', 'asfreq', 'asof', 'astype', 'at', 'at_time', 'attrs', 'autocorr', 'axes',
'backfill', 'between', 'between_time', 'bfill', 'bool', 'clip', 'combine', 'combine_firs
t', 'compare', 'convert_dtypes', 'copy', 'corr', 'count', 'cov', 'cummax', 'cummin', 'cu
mprod', 'cumsum', 'describe', 'diff', 'div', 'divide', 'divmod', 'dot', 'drop', 'drop_du
plicates', 'droplevel', 'dropna', 'dtype', 'dtypes', 'duplicated', 'empty', 'eq', 'equal
s', 'ewm', 'expanding', 'explode', 'factorize', 'ffill', 'fillna', 'filter', 'first', 'f
irst_valid_index', 'flags', 'floordiv', 'ge', 'get', 'groupby', 'gt', 'hasnans', 'head',
'hist', 'iat', 'idxmax', 'idxmin', 'iloc', 'index', 'infer_objects', 'info', 'interpolat
e', 'is_monotonic', 'is_monotonic_decreasing', 'is_monotonic_increasing', 'is_unique',
'isin', 'isna', 'isnull', 'item', 'items', 'iteritems', 'keys', 'kurt', 'kurtosis', 'las
t', 'last_valid_index', 'le', 'loc', 'lt', 'mad', 'map', 'mask', 'max', 'mean', 'media
n', 'memory_usage', 'min', 'mod', 'mode', 'mul', 'multiply', 'name', 'nbytes', 'ndim',
'ne', 'nlargest', 'notna', 'notnull', 'nsmallest', 'nunique', 'pad', 'pct_change', 'pip
e', 'plot', 'pop', 'pow', 'prod', 'product', 'quantile', 'radd', 'rank', 'ravel', 'rdi
v', 'rdivmod', 'reindex', 'reindex_like', 'rename', 'rename_axis', 'reorder_levels', 're
peat', 'replace', 'resample', 'reset_index', 'rfloordiv', 'rmod', 'rmul', 'rolling', 'ro
und', 'rpow', 'rsub', 'rtruediv', 'sample', 'searchsorted', 'sem', 'set_axis', 'set_flag
s', 'shape', 'shift', 'size', 'skew', 'slice_shift', 'sort_index', 'sort_values', 'squee
ze', 'std', 'sub', 'subtract', 'sum', 'swapaxes', 'swaplevel', 'tail', 'take', 'to_clipb
oard', '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']
sorted: [33, 33, 35, 37, 39, 40, 40, 40, 40, 42, 42, 43, 44, 44, 44, 45, 46, 46, 48, 49,
49, 49, 49, 50, 50, 50, 51, 54, 56, 56, 56, 56, 57, 61, 62, 64, 65, 65, 66, 66, 66, 66,
67, 68, 70, 70, 70, 71, 71, 72, 72, 72, 72, 72, 73, 74, 74, 75, 76, 76, 76, 76, 77, 77,
78, 78, 78, 79, 79, 80, 80, 80, 81, 81, 82, 82, 83, 83, 83, 84, 84, 84, 85, 86, 86, 86,
87, 87, 87, 87, 88, 88, 88, 88, 88, 89, 89, 89, 90, 90, 90, 90, 91, 92, 92, 92, 93, 93,
93, 93, 95, 95, 96, 96, 96, 96, 97, 97, 98, 98, 99, 99, 100, 100, 100, 101, 101, 101, 10
2, 102, 103, 103, 104, 104, 104, 105, 105, 105, 105, 105, 105, 105, 105, 105, 105, 108, 108,
108, 108, 108, 108, 109, 109, 110, 110, 110, 111, 111, 112, 113, 113, 113, 114, 114, 11
4, 114, 115, 115, 115, 115, 117, 117, 117, 118, 118, 119, 119, 119, 119, 120, 122, 123,
123, 123, 123, 123, 124, 125, 126, 127, 128, 128, 129, 130, 131, 131, 132, 132, 134, 13
4, 134, 135, 135, 136, 136, 136, 137, 138, 138, 138, 139, 140, 144, 145, 146, 146, 146,
146, 147, 149, 150, 150, 150, 150, 151, 152, 152, 152, 153, 153, 153, 154, 154, 154, 15
5, 155, 156, 156, 156, 156, 157, 157, 157, 157, 158, 158, 159, 159, 160, 160, 160, 160,
162, 164, 166, 167, 167, 168, 170, 170, 170, 170, 171, 172, 172, 173, 173, 173, 174, 17
4, 175, 175, 176, 176, 177, 178, 179, 179, 180, 180, 180, 182, 183, 183, 183, 184, 184,
184, 185, 185, 185, 185, 186, 186, 186, 188, 189, 190, 190, 192, 192, 192, 196, 196, 19
6, 197, 197, 202, 202, 202, 203, 204, 206, 207, 209, 210, 210, 211, 212, 213, 214, 216,
219, 220, 221, 221, 222, 222, 224, 225, 225, 226, 227, 228, 229, 230, 231, 233, 236, 23
6, 237, 241, 243, 244, 245, 247, 249, 254, 254, 258, 259, 259, 261, 261, 265, 267, 268,
269, 276, 276, 290, 295, 301, 306, 312, 396]
min: 33
max: 396
```

In [ ]:
```
# type conversion
marks
```

Out[ ]:
```
maths       67
english     90
science     89
hindi      100
sst         75
Name: Dhanraj Marks, dtype: int64
```

In [ ]:
```
print(list(marks))  # converting into list
dict(marks)  # converting into dictionary
```

Out[ ]:
```
[67, 90, 89, 100, 75]
{'maths': 67, 'english': 90, 'science': 89, 'hindi': 100, 'sst': 75}
```

In [ ]:
```
# membership operator
'2 States (2014 film)' in movies # search in index
```

Out[ ]:
```
True
```

In [ ]:
```
# search in values
'aliya bhatt' in movies.values
```

Out[ ]:
```
True
```

In [ ]:
```
# looping -> print values
for i in movies[:5]:
    print(i)
```

```
Rajniesh Duggall
Sharman Joshi
Vicky Ahuja
Manoj Bajpayee
aliya bhatt
```

In [ ]:
```
# looping -> print index
...es[:5].index:
```

```
    print(i)
```

```
1920 (film)
1920: London
1920: The Evil Returns
1971 (2007 film)
2 States (2014 film)
```

In [ ]:
```
# Arithmetic Operators ->(Broadcasting)
marks
```

Out[ ]:
```
maths       67
english     90
science     89
hindi      100
sst         75
Name: Dhanraj Marks, dtype: int64
```

In [ ]:
```
# example 1
100 - marks
```

Out[ ]:
```
maths      33
english    10
science    11
hindi       0
sst        25
Name: Dhanraj Marks, dtype: int64
```

In [ ]:
```
# example 2
100 + marks
```

Out[ ]:
```
maths      167
english    190
science    189
hindi      200
sst        175
Name: Dhanraj Marks, dtype: int64
```

In [ ]:
```
# example 3
100 * marks
```

Out[ ]:
```
maths       6700
english     9000
science     8900
hindi      10000
sst         7500
Name: Dhanraj Marks, dtype: int64
```

In [ ]:
```
# example 4
100 / marks
```

Out[ ]:
```
maths      1.492537
english    1.111111
science    1.123596
hindi      1.000000
sst        1.333333
Name: Dhanraj Marks, dtype: float64
```

In [ ]:
```
# example 5
marks + 100
```

Out[ ]:
```
maths      167
english    190
science    189
hindi      200
sst        175
Name: Dhanraj Marks, dtype: int64
```

In [ ]:
```
# Relational Operators
vk
```

Out[ ]:
```
match_no
87       0
211      0
207      0
206      0
91       0
      ...
164    100
120    100
123    108
126    109
128    113
Name: runs, Length: 215, dtype: int64
```

In [ ]:
```
vk >= 50
```

Out[ ]:
```
match_no
87     False
211    False
207    False
206    False
91     False
       ...
164     True
120     True
123     True
126     True
128     True
Name: runs, Length: 215, dtype: bool
```

In [ ]:
```
vk[vk >= 50]
```

Out[ ]:
```
match_no
15      50
182     50
197     51
103     51
71      51
122     52
198     53
131     54
129     54
137     55
44      56
85      56
80      57
57      57
144     57
141     58
209     58
34      58
73      58
132     62
104     62
134     64
74      65
45      67
162     67
97      67
148     68
52      70
152     70
41      71
175     72
188     72
68      73
213     73
99      73
127     75
116     75
117     79
119     80
110     82
160     84
178     90
145     92
81      93
82      99
164    100
120    100
123    108
126    109
128    113
Name: runs, dtype: int64
```

Boolean Indexing On Series

In [ ]:
```
# find number of 50's or more than 50's scored by kohli
vk[vk >= 50].size
```

Out[ ]:
```
50
```

In [ ]:
```
# find number of ducks( 0 runs)
vk[vk == 0].size
```

Out[ ]:
```
9
```

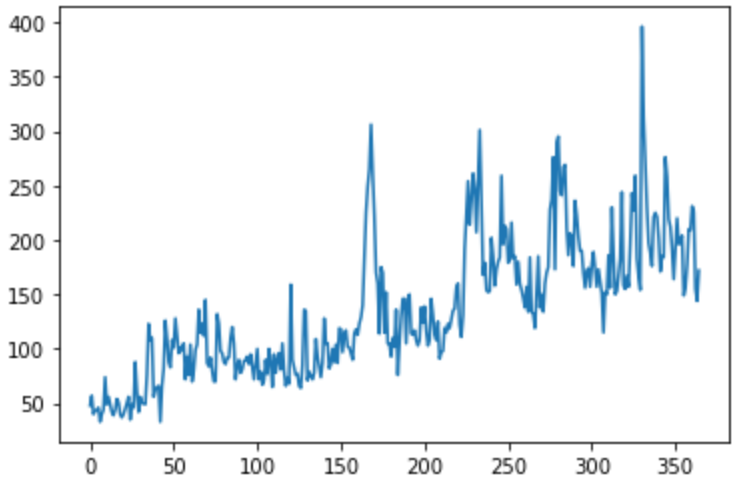
In [ ]:
```
# find actors who have done more than 20 movies
num = movies.value_counts()
num[num > 20]
```

Out[ ]:
```
Akshay Kumar         48
Amitabh Bachchan     45
Ajay Devgn           38
Salman Khan          31
Sanjay Dutt          26
Shah Rukh Khan       22
Emraan Hashmi        21
Name: lead, dtype: int64
```

Plotting Graphs on Series

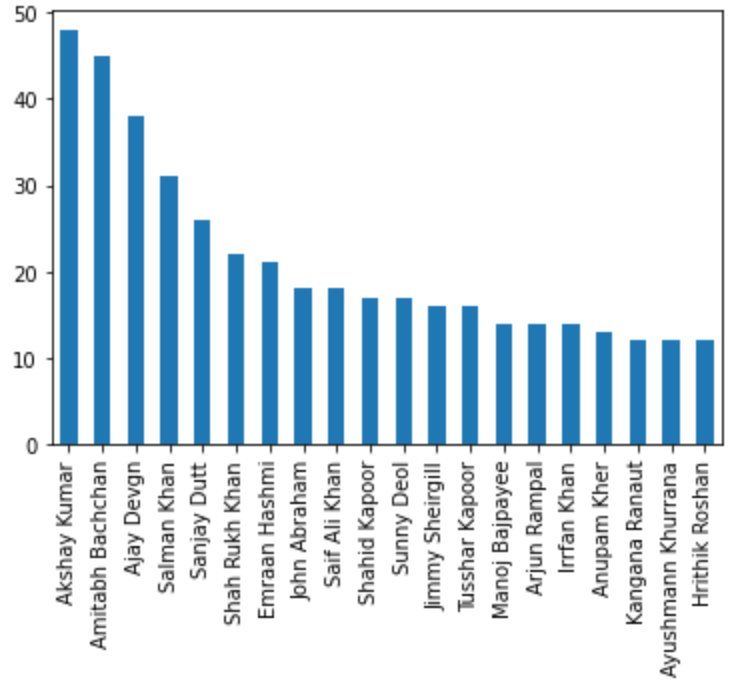
In [ ]:
```
subs.plot()
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]:
```
# bar chart
movies.value_counts().head(20).plot(kind='bar')
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]:
```
# pie chart
movies.value_counts().head(20).plot(kind='pie')
```

Out[ ]:
```
<AxesSubplot:ylabel='lead'>
```