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.

- Prudhvi Vardhan Notes

Python Data Analy
pandas.pydata.org
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
overview // get pandas // documentation // community // talk
Analysis Library
VERSION

Pandas Series

A Pandas Series is like a column in a table. It is a 1-D array holding data of any type.

Importing Pandas

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

Series using String

In [6]:
```
# string
country = ['India','Pakistan','USA','Nepal','Srilanka']
pd.Series(country)
```

Out[6]:
```
0       India
1    Pakistan
2         USA
3       Nepal
4    Srilanka
dtype: object
```

In [7]:
```
# integers
marks= [13,24,56,78,100]
pd.Series(marks)
```

Out[7]:
```
0     13
1     24
2     56
3     78
4    100
dtype: int64
```

In [8]:
```
# custom index
marks = [67,57,89,100]
subjects = ['maths','english','science','hindi']

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

Out[8]:
```
maths       67
english     57
science     89
hindi      100
dtype: int64
```

In [10]:
```
# setting a name
marks = pd.Series(marks , index=subjects , name="Jack Marks")
marks
```

Out[10]:
```
maths       67
english     57
science     89
hindi      100
Name: Jack Marks, dtype: int64
```

Series from dictionary

When a Pandas Series is converted to a dictionary using the to_dict() method, the resulting dictionary has the same keys and values as the Series. The index values of the Series become the keys in the dictionary, and the corresponding values become the values in the dictionary.

In [11]:
```
marks = {
    'maths':67,
    'english':57,
    'science':89,
    'hindi':100
}
marks_series = pd.Series(marks,name="jack Marks")
```

In [12]:
```
marks_series
```

Out[12]:
```
maths       67
english     57
science     89
hindi      100
Name: jack Marks, dtype: int64
```

Series Attributes

size: Returns the number of elements in the Series.

In [13]:
```
# size
marks_series.size
```

Out[13]:
```
4
```

dtype: Returns the data type of the values in the Series.

In [14]:
```
# dtype
marks_series.dtype
```

Out[14]:
```
dtype('int64')
```

name: Returns the name of the Series.

In [15]:
```
# name
marks_series.name
```

Out[15]:
```
'jack Marks'
```

unique is an attribute of a Pandas Series that returns an array of the unique values in the Series.

In [16]:
```
# is_unique
marks_series.is_unique
```

Out[16]:
```
True
```

In [17]:
```
pd.Series([1,1,2,3,4,44,2]).is_unique #It gives false because of repetation
```

Out[17]:
```
False
```

index: Returns the index labels of the Series.

In [18]:
```
# index
marks_series.index
```
Out[18]:
```
Index(['maths', 'english', 'science', 'hindi'], dtype='object')
```

values: Returns the data contained in the Series as a NumPy array.

In [19]:
```
# values
marks_series.values
```
Out[19]:
```
array([ 67,  57,  89, 100], dtype=int64)
```

In [20]:
```
type(marks_series.values)
```
Out[20]:
```
numpy.ndarray
```

Series using read_csv

In [21]:
```
# with one col
sub = pd.read_csv("D:\\datascience\\Nitish isr\\Pandas\\subs.csv")
```

Pandas.read_csv

Automatically converts everything into data frames not in series.

In [23]:
```
type(sub)
```
Out[23]:
```
pandas.core.frame.DataFrame
```

In [30]:
```
sub.head(5)
```
Out[30]:

	Subscribers gained
0	48
1	57
2	40
3	43
4	44

To convert data into series,

we have to apply a parameter called as"Squeeze" is equals to True.

In [31]:
```
sub = pd.read_csv("subs.csv",squeeze=True)
```

In [32]:
```
type(sub)
```

Out[32]:
```
pandas.core.series.Series
```

In [33]:
```
sub
```

Out[33]:
```
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 [56]:
```
#With 2 col
kl=pd.read_csv("kohli_ipl.csv",index_col="match_no",squeeze=True)
```

In [57]:
```
kl
```

Out[57]:
```
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 [37]:
```
movies=pd.read_csv( "bollywood.csv", index_col="movie",squeeze=True)
```

In [38]:
```
movies
```

Out[38]:
```
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

head(n): Returns the first n elements of the Series.

In [40]:
```
# Head
sub.head()
```

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

tail(n): Returns the last n elements of the Series.

In [41]:
```
# tail
kl.tail()
```

Out[41]:
```
match_no
211     0
212    20
213    73
214    25
215     7
Name: runs, dtype: int64
```

In [43]:
```
# sample - Gives random data
movies.sample()
```

Out[43]:
```
movie
Enemmy    Sunil Shetty
Name: lead, dtype: object
```

value_counts(): Returns a Series containing the counts of unique values in the Series.

In [44]:
```
# Value Counts
movies.value_counts()
```

Out[44]:
```
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 [45]:
```
#sort_values - temperory changes #### sort_values(): Returns a sorted Series b
kl.sort_values()
```

Out[45]:
```
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 [50]:
```
# method chaining
kl.sort_values(ascending=False).head(1).values[0]
```

Out[50]:
```
113
```

In [55]:
```
# For permanent Changes use Inplace
kl.sort_values(inplace=True)
kl
```

Out[55]:
```
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 [60]:
```
# sort_index -> inplace -> movies

movies.sort_index()
```

Out[60]:
```
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 [61]:
```
movies.sort_index(ascending=False)
```

Out[61]:
```
movie
Zor Lagaa Ke...Haiya!           Meghan Jadhav
Zokkomon                      Darsheel Safary
Zindagi Tere Naam          Mithun Chakraborty
Zindagi Na Milegi Dobara       Hrithik Roshan
Zindagi 50-50                     Veena Malik
                                  ...        
2 States (2014 film)             Arjun Kapoor
1971 (2007 film)               Manoj Bajpayee
1920: The Evil Returns            Vicky Ahuja
1920: London                    Sharman Joshi
1920 (film)                  Rajniesh Duggall
Name: lead, Length: 1500, dtype: object
```

Series Maths Methods

Diffence between Count And Size

Count gives the total number of items present in the series. But only NON missing values but, if we have missing values ,it doesnt count them . But, size gives the total item including missing values

In [62]:
```
# count
kl.count()
```

Out[62]:
```
215
```

sum(): Returns the sum of the values in the Series.

In [66]:
```
# sum -> Product
sub.sum()
```

Out[66]:
```
49510
```

In [67]:
```
sub.product() # Multiply the items
```

Out[67]:
```
0
```

Statical Methods

mean(): Returns the mean value of the Series.

In [68]:
```
# mean


sub.mean()
```

Out[68]:
```
135.64383561643837
```

median(): Returns the median value of the Series.

In [72]:
```
# median
kl.median()
```

Out[72]:
```
24.0
```

mode(): The mode is the value that appears most frequently in the Series.

In [74]:
```
# mode
print(movies.mode())
```
```
0    Akshay Kumar
dtype: object
```

std(): Returns the standard deviation of the values in the Series.

In [71]:
```
# std -> Standard deviation
sub.std()
```
Out[71]:
```
62.67502303725269
```

In [75]:
```
# var -> varience
sub.var()
```
Out[75]:
```
3928.1585127201556
```

min(): Returns the minimum value of the Series.

In [76]:
```
# min
sub.min()
```
Out[76]:
```
33
```

max(): Returns the maximum value of the Series.

In [77]:
```
# max
sub.max()
```
Out[77]:
```
396
```

describe(): Generates descriptive statistics of the Series.

In [79]:
```
# describe
movies.describe()
```
Out[79]:
```
count             1500
unique             566
top       Akshay Kumar
freq                48
Name: lead, dtype: object
```

In [80]:
```
kl.describe()
```

Out[80]:
```
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
```

In [81]:
```
sub.describe()
```

Out[81]:
```
count    365.000000
mean     135.643836
std       62.675023
min       33.000000
25%       88.000000
50%      123.000000
75%      177.000000
max      396.000000
Name: Subscribers gained, dtype: float64
```

Series Indexing

In [83]:
```
# integer indexing
x = pd.Series([12,13,14,35,46,57,58,79,9])
x[1]
```

Out[83]:
```
13
```

In [85]:
```
# negative indexing
movies[-1]
```

Out[85]:
```
'Akshay Kumar'
```

In [86]:
```
movies[0]
```

Out[86]:
```
'Vicky Kaushal'
```

In [87]:
```
sub[0]
```

Out[87]:
```
48
```

In [90]:
```
# slicing
kl[4:10]
```

Out[90]:
```
match_no
5      1
6      9
7     34
8      0
9     21
10     3
Name: runs, dtype: int64
```

In [95]:
```
#Negative slicing

sub[-5:]
```

Out[95]:
```
360    231
361    226
362    155
363    144
364    172
Name: Subscribers gained, dtype: int64
```

In [96]:
```
movies[-5:]
```

Out[96]:
```
movie
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, dtype: object
```

In [97]:
```
movies[::2]
```

Out[97]:
```
movie
Uri: The Surgical Strike                   Vicky Kaushal
The Accidental Prime Minister (film)         Anupam Kher
Evening Shadows                         Mona Ambegaonkar
Fraud Saiyaan                               Arshad Warsi
Manikarnika: The Queen of Jhansi          Kangana Ranaut
                                              ...
Raaz (2002 film)                              Dino Morea
Waisa Bhi Hota Hai Part II                  Arshad Warsi
Kaante                                  Amitabh Bachchan
Aankhen (2002 film)                     Amitabh Bachchan
Company (film)                                Ajay Devgn
Name: lead, Length: 750, dtype: object
```

In [98]:
```
# Fancy indexing
kl[[1,8,22,11,2]]
```

Out[98]:
```
match_no
1      1
8      0
22    38
11    10
2     23
Name: runs, dtype: int64
```

In [99]:
```
# Fancy indexing -> indexing with labels
movies
```

Out[99]:
```
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 [100]:
```
movies['Evening Shadows']
```

Out[100]:
```
'Mona Ambegaonkar'
```

Editing the series

In [101]:
```
# using the index number
marks_series
```

Out[101]:
```
maths       67
english     57
science     89
hindi      100
Name: jack Marks, dtype: int64
```

In [102]:
```
marks_series[1]=88
marks_series
```

Out[102]:
```
maths       67
english     88
science     89
hindi      100
Name: jack Marks, dtype: int64
```

In [103]:
```
# we can add data , if it doesnt exist
marks_series['social']=90
marks_series
```

Out[103]:
```
maths       67
english     88
science     89
hindi      100
social      90
Name: jack Marks, dtype: int64
```

In [111]:
```
# using index label
movies
```

Out[111]:
```
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 [114]:
```
movies['Hum Tumhare Hain Sanam'] = 'Jack'
```

In [115]:
```
movies
```

Out[115]:
```
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                              Jack
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 with Python Functionalities

In [117]:
```
# len/type/dir/sorted/max/min
print(len(sub))
print(type(sub))
```

```
365
<class 'pandas.core.series.Series'>
```

In [122]:
```
print(dir(sub))
print(sorted(sub))
```

```
['T', '_AXIS_LEN', '_AXIS_ORDERS', '_AXIS_REVERSED', '_AXIS_TO_AXIS_NUMBER',
'_HANDLED_TYPES', '__abs__', '__add__', '__and__', '__annotations__', '__arra
y__', '__array_priority__', '__array_ufunc__', '__array_wrap__', '__bool__',
'__class__', '__contains__', '__copy__', '__deepcopy__', '__delattr__', '__de
litem__', '__dict__', '__dir__', '__divmod__', '__doc__', '__eq__', '__finali
ze__', '__float__', '__floordiv__', '__format__', '__ge__', '__getattr__', '_
_getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iad
d__', '__iand__', '__ifloordiv__', '__imod__', '__imul__', '__init__', '__ini
t_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__', '__trued
iv__', '__weakref__', '__xor__', '_accessors', '_accum_func', '_add_numeric_o
perations', '_agg_by_level', '_agg_examples_doc', '_agg_see_also_doc', '_alig
n_frame', '_align_series', '_arith_method', '_as_manager', '_attrs', '_bino
p', '_can_hold_na', '_check_inplace_and_allows_duplicate_labels', '_check_inp
lace_setting', '_check_is_chained_assignment_possible', '_check_label_or_leve
l_ambiguity', '_check_setitem_copy', '_clear_item_cache', '_clip_with_one_bou
nd', '_clip_with_scalar', '_cmp_method', '_consolidate', '_consolidate_inplac
e', '_construct_axes_dict', '_construct_axes_from_arguments', '_construct_res
ult', '_constructor', '_constructor_expanddim', '_convert', '_convert_dtype
s', '_data', '_dir_additions', '_dir_deletions', '_drop_axis', '_drop_labels_
or_levels', '_duplicated', '_find_valid_index', '_flags', '_from_mgr', '_get_
axis', '_get_axis_name', '_get_axis_number', '_get_axis_resolvers', '_get_blo
ck_manager_axis', '_get_bool_data', '_get_cacher', '_get_cleaned_column_resol
vers', '_get_index_resolvers', '_get_label_or_level_values', '_get_numeric_da
ta', '_get_value', '_get_values', '_get_values_tuple', '_get_with', '_gotite
m', '_hidden_attrs', '_index', '_indexed_same', '_info_axis', '_info_axis_nam
e', '_info_axis_number', '_init_dict', '_init_mgr', '_inplace_method', '_inte
rnal_names', '_internal_names_set', '_is_cached', '_is_copy', '_is_label_or_l
evel_reference', '_is_label_reference', '_is_level_reference', '_is_mixed_typ
e', '_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_consolid
ate', '_reduce', '_reindex_axes', '_reindex_indexer', '_reindex_multi', '_rei
ndex_with_indexers', '_replace_single', '_repr_data_resource_', '_repr_latex
_', '_reset_cache', '_reset_cacher', '_set_as_cached', '_set_axis', '_set_axi
s_name', '_set_axis_nocheck', '_set_is_copy', '_set_labels', '_set_name', '_s
et_value', '_set_values', '_set_with', '_set_with_engine', '_slice', '_stat_a
xis', '_stat_axis_name', '_stat_axis_number', '_stat_function', '_stat_functi
on_ddof', '_take_with_is_copy', '_typ', '_update_inplace', '_validate_dtype',
'_values', '_where', 'abs', 'add', 'add_prefix', 'add_suffix', 'agg', 'aggreg
ate', 'align', 'all', 'any', 'append', 'apply', 'argmax', 'argmin', 'argsor
t', 'array', 'asfreq', 'asof', 'astype', 'at', 'at_time', 'attrs', 'autocor
r', 'axes', 'backfill', 'between', 'between_time', 'bfill', 'bool', 'clip',
'combine', 'combine_first', 'compare', 'convert_dtypes', 'copy', 'corr', 'cou
nt', 'cov', 'cummax', 'cummin', 'cumprod', 'cumsum', 'describe', 'diff', 'di
v', 'divide', 'divmod', 'dot', 'drop', 'drop_duplicates', 'droplevel', 'dropn
a', 'dtype', 'dtypes', 'duplicated', 'empty', 'eq', 'equals', 'ewm', 'expandi
ng', 'explode', 'factorize', 'ffill', 'fillna', 'filter', 'first', 'first_val
id_index', 'flags', 'floordiv', 'ge', 'get', 'groupby', 'gt', 'hasnans', 'hea
d', 'hist', 'iat', 'idxmax', 'idxmin', 'iloc', 'index', 'infer_objects', 'int
erpolate', 'is_monotonic', 'is_monotonic_decreasing', 'is_monotonic_increasin
g', 'is_unique', 'isin', 'isna', 'isnull', 'item', 'items', 'iteritems', 'key
s', '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', 'notn
ull', 'nsmallest', 'nunique', 'pad', 'pct_change', 'pipe', 'plot', 'pop', 'po
w', 'prod', 'product', 'quantile', 'radd', 'rank', 'ravel', 'rdiv', 'rdivmo
d', 'reindex', 'reindex_like', 'rename', 'rename_axis', 'reorder_levels', 're
peat', 'replace', 'resample', 'reset_index', 'rfloordiv', 'rmod', 'rmul', 'ro
lling', 'round', 'rpow', 'rsub', 'rtruediv', 'sample', 'searchsorted', 'sem',
'set_axis', 'set_flags', 'shape', 'shift', 'size', 'skew', 'slice_shift', 'so
rt_index', 'sort_values', 'squeeze', 'std', 'sub', 'subtract', 'sum', 'swapax
es', 'swaplevel', 'tail', 'take', 'to_clipboard', 'to_csv', 'to_dict', 'to_ex
cel', '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', 'va
r', 'view', 'where', 'xs']
[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, 6
6, 66, 66, 66, 67, 68, 70, 70, 70, 71, 71, 72, 72, 72, 72, 72, 73, 74, 74, 7
5, 76, 76, 76, 76, 77, 77, 78, 78, 78, 79, 79, 80, 80, 80, 81, 81, 82, 82, 8
3, 83, 83, 84, 84, 84, 85, 86, 86, 86, 87, 87, 87, 87, 88, 88, 88, 88, 88, 8
9, 89, 89, 90, 90, 90, 90, 91, 92, 92, 92, 93, 93, 93, 93, 95, 95, 96, 96, 9
6, 96, 97, 97, 98, 98, 99, 99, 100, 100, 100, 101, 101, 101, 102, 102, 103, 1
03, 104, 104, 104, 105, 105, 105, 105, 105, 105, 105, 105, 105, 108, 108, 10
8, 108, 108, 108, 109, 109, 110, 110, 110, 111, 111, 112, 113, 113, 113, 114,
114, 114, 114, 115, 115, 115, 115, 117, 117, 117, 118, 118, 119, 119, 119, 11
9, 120, 122, 123, 123, 123, 123, 123, 124, 125, 126, 127, 128, 128, 129, 130,
131, 131, 132, 132, 134, 134, 134, 135, 135, 136, 136, 136, 137, 138, 138, 13
8, 139, 140, 144, 145, 146, 146, 146, 146, 147, 149, 150, 150, 150, 150, 151,
152, 152, 152, 153, 153, 153, 154, 154, 154, 155, 155, 156, 156, 156, 156, 15
7, 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, 174, 175, 17
5, 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, 19
6, 196, 196, 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, 22
8, 229, 230, 231, 233, 236, 236, 237, 241, 243, 244, 245, 247, 249, 254, 254,
258, 259, 259, 261, 261, 265, 267, 268, 269, 276, 276, 290, 295, 301, 306, 31
2, 396]
```

In [123]:
```
print(min(sub))
print(max(sub))
```

```
33
396
```

In [125]:
```
# type conversion
list(marks_series)
```

Out[125]:
```
[67, 88, 89, 100, 90]
```

In [126]:
```
dict(marks_series)
```
Out[126]:
```
{'maths': 67, 'english': 88, 'science': 89, 'hindi': 100, 'social': 90}
```

In [129]:
```
# membership operator
'Hum Tumhare Hain Sanam' in movies # In oprator only searches in index values
```
Out[129]:
```
True
```

In [133]:
```
"Jack" in movies.values
```
Out[133]:
```
True
```

In [138]:
```
# looping
for i in movies:
    print(i)
```
```
Vicky Kaushal
Vicky Ahuja
Anupam Kher
Emraan Hashmi
Mona Ambegaonkar
Geetika Vidya Ohlyan
Arshad Warsi
Radhika Apte
Kangana Ranaut
Nawazuddin Siddiqui
Ali Asgar
Ranveer Singh
Prit Kamani
Ajay Devgn
Sushant Singh Rajput
Amitabh Bachchan
Abhimanyu Dasani
Talha Arshad Reshi
Nawazuddin Siddiqui
```

In [139]:
```
for i in movies.index:
    print(i)
```

```
Uri: The Surgical Strike
Battalion 609
The Accidental Prime Minister (film)
Why Cheat India
Evening Shadows
Soni (film)
Fraud Saiyaan
Bombairiya
Manikarnika: The Queen of Jhansi
Thackeray (film)
Amavas
Gully Boy
Hum Chaar
Total Dhamaal
Sonchiriya
Badla (2019 film)
Mard Ko Dard Nahi Hota
Hamid (film)
Photograph (film)
```

In [140]:
```
# Arthematic Operators (Broadcasting)
100-marks_series
```

Out[140]:
```
maths      33
english    12
science    11
hindi       0
social     10
Name: jack Marks, dtype: int64
```

In [141]:
```
100+marks_series
```

Out[141]:
```
maths      167
english    188
science    189
hindi      200
social     190
Name: jack Marks, dtype: int64
```

In [143]:
```
# Relational operators
kl>=50
```

Out[143]:
```
match_no
1      False
2      False
3      False
4      False
5      False
       ...  
211    False
212    False
213     True
214    False
215    False
Name: runs, Length: 215, dtype: bool
```

Boolean Indexing on Series

In [146]:
```
# Find no of 50's and 100's scored by kohli
kl[kl>=50].size
```

Out[146]:
```
50
```

In [148]:
```
# find number of ducks
kl[kl == 0].size
```

Out[148]:
```
9
```

In [149]:
```
# Count number of day when I had more than 200 subs a day
sub[sub>=200].size
```

Out[149]:
```
59
```

In [159]:
```
# find actors who have done more than 20 movies
num_mov=movies.value_counts()
num_mov[num_mov>=20]
```

Out[159]:
```
Akshay Kumar        48
Amitabh Bachchan    45
Ajay Devgn          38
Salman Khan         31
Sanjay Dutt         26
Shah Rukh Khan      21
Emraan Hashmi       21
Name: lead, dtype: int64
```

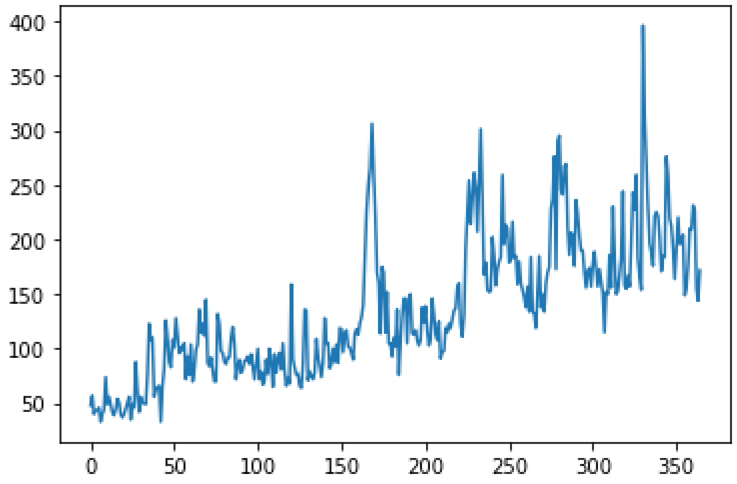
In [160]:
```
num_mov[num_mov>=20].size
```

Out[160]:
```
7
```

Plotting Graphs on Series

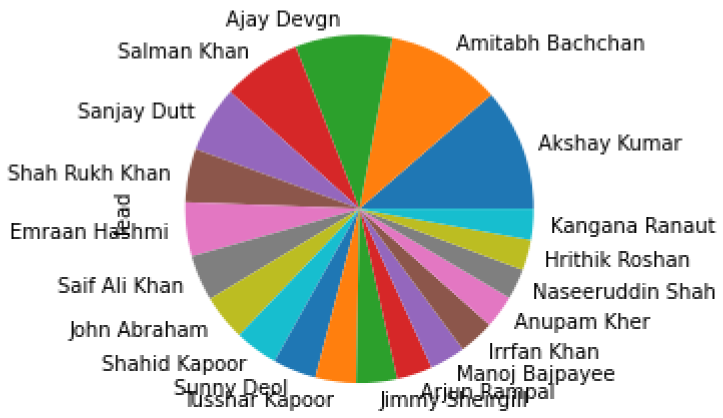
In [162]:
```
sub.plot()
```

Out[162]:
```
<AxesSubplot:>
```



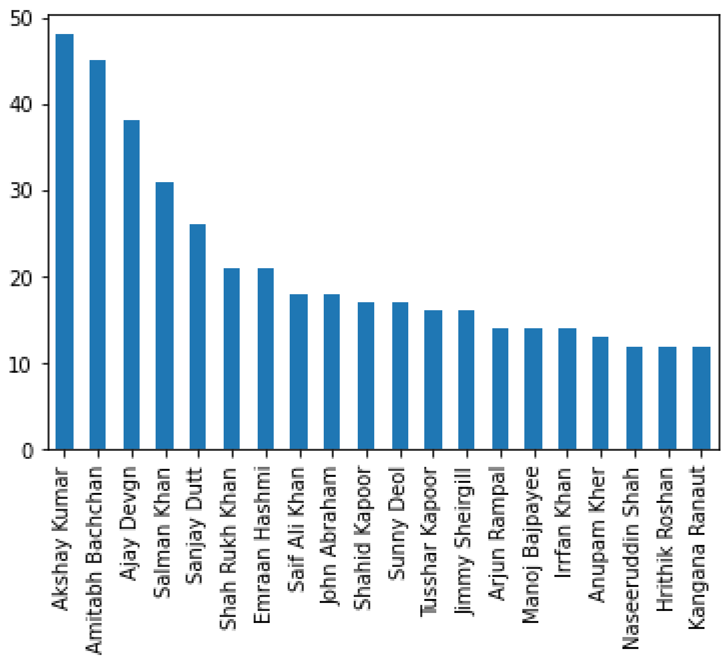
In [164]:
```
movies.value_counts().head(20).plot(kind="pie")
```

Out[164]:
```
<AxesSubplot:ylabel='lead'>
```



In [165]:
```
movies.value_counts().head(20).plot(kind="bar")
```

Out[165]:
```
<AxesSubplot:>
```



Some Important Series Methods

In [166]:
```
# astype
# between
# clip
# drop_duplicates
# isnull
# dropna
# fillna
# isin
# apply
# copy
```

In [175]:
```
# astype
import sys
sys.getsizeof(kl)
```

Out[175]:
```
11752
```

In [176]:
```
kl
```

Out[176]:
```
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 [177]:
```
(kl.astype("int16"))
```

Out[177]:
```
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: int16
```

In [178]:
```
sys.getsizeof(kl.astype("int16"))
```

Out[178]:
```
10462
```

In [181]:
```
# between
kl[kl.between(50,60)]
```

Out[181]:
```
match_no
15     50
34     58
44     56
57     57
71     51
73     58
80     57
85     56
103    51
122    52
129    54
131    54
137    55
141    58
144    57
182    50
197    51
198    53
209    58
Name: runs, dtype: int64
```

In [182]:
```
kl[kl.between(50,60)].size
```

Out[182]:
```
19
```

In [183]:
```
# clip
sub.clip(100,200)
```

Out[183]:
```
0      100
1      100
2      100
3      100
4      100
      ...
360    200
361    200
362    155
363    144
364    172
Name: Subscribers gained, Length: 365, dtype: int64
```

In [186]:
```
# drop duplicates #### drop_duplicates(): Returns a Series with duplicates rem

dele = pd.Series([1,2,33,3,3,3,1,23,33,22,33,11])
dele
```

Out[186]:
```
0      1
1      2
2     33
3      3
4      3
5      3
6      1
7     23
8     33
9     22
10    33
11    11
dtype: int64
```

In [188]:
```
dele.drop_duplicates()
```

Out[188]:
```
0      1
1      2
2     33
3      3
7     23
9     22
11    11
dtype: int64
```

In [189]:
```
dele.drop_duplicates(keep='last')
```

Out[189]:
```
1      2
5      3
6      1
7     23
9     22
10    33
11    11
dtype: int64
```

In [190]:
```
movies.drop_duplicates()
```

Out[190]:
```
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
                                                 ...
Rules: Pyaar Ka Superhit Formula                     Tanuja
Right Here Right Now (film)                           Ankit
Talaash: The Hunt Begins...                   Rakhee Gulzar
The Pink Mirror                             Edwin Fernandes
Hum Tumhare Hain Sanam                                 Jack
Name: lead, Length: 567, dtype: object
```

In [191]:
```
dele.duplicated().sum()
```

Out[191]:
```
5
```

In [193]:
```
kl.duplicated().sum()
```

Out[193]:
```
137
```

In [194]:
```
dele.count()
```

Out[194]:
```
12
```

isin(values): Returns a boolean Series indicating whether each element in the Series is in the provided values

In [198]:
```
# isnull

kl.isnull().sum()
```

Out[198]:
```
0
```

In [199]:
```
dele.isnull().sum()
```

Out[199]:
```
0
```

In [200]:
```
# dropna

dele.dropna()
```

Out[200]:
```
0      1
1      2
2     33
3      3
4      3
5      3
6      1
7     23
8     33
9     22
10    33
11    11
dtype: int64
```

In [202]:
```
# fillna

dele.fillna(0)
dele.fillna(dele.mean())
```

Out[202]:
```
0      1
1      2
2     33
3      3
4      3
5      3
6      1
7     23
8     33
9     22
10    33
11    11
dtype: int64
```

In [205]:
```
# isin

kl
```

Out[205]:
```
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 [207]:
```
kl[(kl==49) | (kl==99)]
```

Out[207]:
```
match_no
82    99
86    49
Name: runs, dtype: int64
```

In [209]:
```
kl[kl.isin([49,99])]
```

Out[209]:
```
match_no
82    99
86    49
Name: runs, dtype: int64
```

In [210]:
```
# apply

movies
```

Out[210]:
```
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                               Jack
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 [212]:
```
movies.apply(lambda x:x.split()) # split name in to two using Lambda function
```

Out[212]:
```
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                                [Jack]
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 [213]:
```
movies.apply(lambda x:x.split()[0]) # select first word
```

Out[213]:
```
movie
Uri: The Surgical Strike                      Vicky
Battalion 609                                 Vicky
The Accidental Prime Minister (film)         Anupam
Why Cheat India                              Emraan
Evening Shadows                                Mona
                                             ...
Hum Tumhare Hain Sanam                         Jack
Aankhen (2002 film)                         Amitabh
Saathiya (film)                               Vivek
Company (film)                                 Ajay
Awara Paagal Deewana                         Akshay
Name: lead, Length: 1500, dtype: object
```

In [214]:
```
movies.apply(lambda x:x.split()[0].upper()) # Upper case
```

Out[214]:
```
movie
Uri: The Surgical Strike                      VICKY
Battalion 609                                 VICKY
The Accidental Prime Minister (film)         ANUPAM
Why Cheat India                              EMRAAN
Evening Shadows                                MONA
                                             ...
Hum Tumhare Hain Sanam                         JACK
Aankhen (2002 film)                         AMITABH
Saathiya (film)                               VIVEK
Company (film)                                 AJAY
Awara Paagal Deewana                         AKSHAY
Name: lead, Length: 1500, dtype: object
```

In [215]:
```
sub
```

Out[215]:
```
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 [216]:
```
sub.mean()
```

Out[216]:
```
135.64383561643837
```

In [217]:
```
sub.apply(lambda x:'good day' if x > sub.mean() else 'bad day')
```

Out[217]:
```
0       bad day
1       bad day
2       bad day
3       bad day
4       bad day
          ...
360    good day
361    good day
362    good day
363    good day
364    good day
Name: Subscribers gained, Length: 365, dtype: object
```

In [229]:
```
# Copy

kl
```

Out[229]:
```
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 [230]:
```
new = kl.head()
```

In [231]:
```
new[1]=100
```

In [232]:
```
new
```

Out[232]:
```
match_no
1    100
2     23
3     13
4     12
5      1
Name: runs, dtype: int64
```

In [233]:
```
kl
```

Out[233]:
```
match_no
1      100
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 [240]:
```
new = kl.head(5).copy()
```

In [241]:
```
new[1]=20
```

In [242]:
```
new
```

Out[242]:
```
match_no
1    20
2    23
3    13
4    12
5     1
Name: runs, dtype: int64
```

In [250]:
```
kl
```

Out[250]:
```
match_no
1      100
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 [ ]:
```

```

In [ ]:
```

```