pandas

August 28, 2023

1 Pandas for Data Science

[1]:
```
import pandas as pd
```

1.1 Access a file :

[2]:
```
df = pd.read_csv("weather_data.csv")        # access all deta from this fiel
df
```

[2]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

1.2 Access row and column :

[3]:
```
r , c = df.shape        # df.shape --> row and comolmn
```

1.3 Head and tail :

[4]:
```
df.head(2)      # 1st 2 row
```

[4]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny

[5]:
```
df.tail(2)      # last 2 row
```

[5]:

	day	temperature	windspeed	event
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

1.4 Specific row access with index:

[6]:
```
df[1:3]          # row index 1 -3
```

[6]:

	day	temperature	windspeed	event
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow

[7]:
```
df[:3]           # row index start -3
```

[7]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow

[8]:
```
df[2:]           # row index 2 - end
```

[8]:

	day	temperature	windspeed	event
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

1.5 Column :

[9]:
```
df.columns       # all columns
```

[9]:
```
Index(['day', 'temperature', 'windspeed', 'event'], dtype='object')
```

1.6 Specific column access:

[10]:
```
df.day           # access day column
```

[10]:
```
0     1/1/2017
1     1/4/2017
2     1/5/2017
3     1/6/2017
4     1/7/2017
5     1/8/2017
6     1/9/2017
7    1/10/2017
8    1/11/2017
Name: day, dtype: object
```

[11]:
```
df["day"]       # access day column
```

[11]:
```
0     1/1/2017
1     1/4/2017
2     1/5/2017
3     1/6/2017
4     1/7/2017
5     1/8/2017
6     1/9/2017
7    1/10/2017
8    1/11/2017
Name: day, dtype: object
```

1.7 Type:

[12]:
```
type(df["day"]) # pandas.core.series.series
```

[12]:
```
pandas.core.series.Series
```

1.8 Access specific column:

[13]:
```
df[['day','temperature']]       # access 2 column
```

[13]:

	day	temperature
0	1/1/2017	32.0
1	1/4/2017	NaN
2	1/5/2017	28.0
3	1/6/2017	NaN
4	1/7/2017	32.0
5	1/8/2017	NaN
6	1/9/2017	NaN
7	1/10/2017	34.0
8	1/11/2017	40.0

1.9 Operation on DataFrame:

[14]:
```
df['temperature'].max()         # return max vlaue of temperature column
```

[14]:
```
40.0
```

[15]:
```
x = df[df["temperature"] > 32]  # conditon apply
df['day'][df['temperature'] == df['temperature'].max()]
                                # pritns the day when your temperature is␣
 ↪maximum
```

[15]:
```
8    1/11/2017
Name: day, dtype: object
```

[16]:
```
x = df[['day' , 'temperature']][df['temperature'] == df['temperature'].max()]
print(x)                          # print day along with temperature when temp max
```

```
         day  temperature
8  1/11/2017         40.0
```

1.10 index: (you can set any column as index)

[17]:
```
df.index              # index 0 - 6 and step 1
```

[17]:
```
RangeIndex(start=0, stop=9, step=1)
```

[18]:
```
df.set_index('day') # set index day column
```

[18]:

day	temperature	windspeed	event
1/1/2017	32.0	6.0	Rain
1/4/2017	NaN	9.0	Sunny
1/5/2017	28.0	NaN	Snow
1/6/2017	NaN	7.0	NaN
1/7/2017	32.0	NaN	Rain
1/8/2017	NaN	NaN	Sunny
1/9/2017	NaN	NaN	NaN
1/10/2017	34.0	8.0	Cloudy
1/11/2017	40.0	12.0	Sunny

[19]:
```
df.set_index('day', inplace=True)    # set index not 0 - 6. it set date index
print(df)
```

day	temperature	windspeed	event
1/1/2017	32.0	6.0	Rain
1/4/2017	NaN	9.0	Sunny
1/5/2017	28.0	NaN	Snow
1/6/2017	NaN	7.0	NaN
1/7/2017	32.0	NaN	Rain
1/8/2017	NaN	NaN	Sunny
1/9/2017	NaN	NaN	NaN
1/10/2017	34.0	8.0	Cloudy
1/11/2017	40.0	12.0	Sunny

[20]:
```
df.reset_index(inplace=True)         # reset this indeing
```

2 Different ways of creationg dataFrame

2.1 Using CSV:

[21]:
```
csv = pd.read_csv("exel.csv")       # read csv file into csv variable
```

2.2 Using Exel:

[22]:
```
exel = pd.read_excel("exel.xlsx")   # read exel file into exel variable
```

2.3 Using python dictionary:

[23]:
```
weather_data = {
     'day': ['1/1/2017', '1/2/2017', '1/3/2017'],
     'temperature': [32, 35,28],
     'windspeed': [6,7,2],
     'event': ['Rain', 'Sunny', 'Snow']
}
dictionary = pd.DataFrame(weather_data)     # read dictionary as dataFrame into␣
 ↪dictionary variable
dictionary
```

[23]:

	day	temperature	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow

2.4 Using tuple list:

[24]:
```
weather_data = [
     ('1/1/2017', 32, 6, 'Rain'),
     ('1/2/2017', 35, 7, 'Sunny'),
     ('1/3/2017', 28, 2, 'Snow')
]
tuple_data = pd.DataFrame(weather_data , columns=["day" , "temp" , "windspeed"␣
 ↪, "event"])
tuple_data
```

[24]:

	day	temp	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow

2.5 List dictionary:

[25]:
```
weather_data = [
    {'day': '1/1/2017', 'temperature': 32, 'windspeed': 6, 'event': 'Rain'},
    {'day': '1/2/2017', 'temperature': 35, 'windspeed' : 7, 'event': 'Sunny'},
    {'day': '1/3/2017', 'temperature': 28, 'windspeed': 2, 'event': 'Snow'},
 ]
dictionary = pd.DataFrame(weather_data)     # creating dictionary list dataFrame
dictionary
```

[25]:

	day	temperature	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow

3 Reading and witing cvs and exel file

3.1 Read from csv file:

[26]:
```
df = pd.read_csv("stock_data.csv" , skiprows=1)     # import all contant into␣
 ↪df and skip 1st row for (skiprows = 1)
df
```

[26]:

	GOOGL	27.82	87	845	larry page
0	WMT	4.61	484	65	n.a.
1	MSFT	-1	85	64	bill gates
2	RIL	not available	50	1023	mukesh ambani
3	TATA	5.6	-1	n.a.	ratan tata

[27]:
```
df = pd.read_csv("stock_data.csv" , header=1)       # import all contant into␣
 ↪df and skip 1st row for (header = 1)
df
```

[27]:

	GOOGL	27.82	87	845	larry page
0	WMT	4.61	484	65	n.a.
1	MSFT	-1	85	64	bill gates
2	RIL	not available	50	1023	mukesh ambani
3	TATA	5.6	-1	n.a.	ratan tata

[28]:
```
df = pd.read_csv("stock_data.csv" , header=None)    # header will indexd with␣
 ↪start 0
df
```

[28]:

	0	1	2	3	4
0	tickers	eps	revenue	price	people
1	GOOGL	27.82	87	845	larry page
2	WMT	4.61	484	65	n.a.

```
3     MSFT             -1       85     64     bill gates
4     RIL   not available       50   1023  mukesh ambani
5     TATA            5.6       -1   n.a.     ratan tata
```

[29]:
```
df = pd.read_csv("stock_data.csv" , header=None ,  names=["tikers" , "eps" ,␣
 ↪"revenue" , "price" ])
# replace header index with this
df
```

[29]:
```
                tikers      eps revenue          price
tickers            eps  revenue   price         people
GOOGL            27.82       87     845     larry page
WMT               4.61      484      65           n.a.
MSFT                -1       85      64     bill gates
RIL      not available       50    1023  mukesh ambani
TATA               5.6       -1    n.a.     ratan tata
```

3.2 Read specific number of rows:

[30]:
```
df = pd.read_csv("stock_data.csv" , nrows=3)        # first 3 rows excluding␣
 ↪your headers
df
```

[30]:
```
  tickers    eps  revenue  price      people
0   GOOGL  27.82       87    845  larry page
1     WMT   4.61      484     65        n.a.
2    MSFT  -1.00       85     64  bill gates
```

3.3 NaN values:

[31]:
```
df = pd.read_csv("stock_data.csv" , na_values=["not available" , "n.a."])
# in will replace "not available" and "n.a. " text to [NaN]
df
```

[31]:
```
  tickers    eps  revenue   price         people
0   GOOGL  27.82       87   845.0     larry page
1     WMT   4.61      484    65.0            NaN
2    MSFT  -1.00       85    64.0     bill gates
3    RIL     NaN       50  1023.0  mukesh ambani
4    TATA   5.60       -1     NaN     ratan tata
```

[32]:
```
df = pd.read_csv("stock_data.csv" , na_values={     # set specific column NaN␣
 ↪values
    "esp" : ["not available" , "n.a. " ],
    "revenue" : ["not available" , "n.a. " , -1] ,
    "people" : ["not availabele" , "n.a. "]
```

```
})
df
```

[32]:

	tickers	eps	revenue	price	people
0	GOOGL	27.82	87.0	845	larry page
1	WMT	4.61	484.0	65	n.a.
2	MSFT	-1	85.0	64	bill gates
3	RIL	not available	50.0	1023	mukesh ambani
4	TATA	5.6	NaN	n.a.	ratan tata

3.4 Write into csv file:

[33]:
```
df.to_csv("new.csv")                  # create a new.csv file
df.to_csv("new.csv" , index=False)    # create a new.csv file without row index
df.to_csv("new.csv" , columns=["eps" , "price"])
                                      # create a new.csv file with 2 column
                                      # header = False --> skip header when create
```

3.5 Read from exel file:

[34]:
```
df = pd.read_excel("stock_data.xlsx" ,"Sheet1")     # onpen Sheet1 form this␣
↪exel file
df
```

[34]:

	tickers	eps	revenue	price	people
0	GOOGL	27.82	87	845	larry page
1	WMT	4.61	484	65	n.a.
2	MSFT	-1	85	64	bill gates
3	RIL	not available	50	1023	mukesh ambani
4	TATA	5.6	-1	n.a.	ratan tata

3.6 Using converters:

[35]:
```
def convert_people(cell):       # this is a demo function
    if cell == "n.a.":
        return "Unkonwn People"
    else:
        return cell

df = pd.read_excel("stock_data.xlsx" , "Sheet1" , converters={
    "people" : convert_people   # using a function for convert people row data
})
df
```

[35]:

	tickers	eps	revenue	price	people
0	GOOGL	27.82	87	845	larry page

```
1     WMT           4.61      484    65  Unkonwn People
2    MSFT             -1       85    64      bill gates
3     RIL   not available       50  1023   mukesh ambani
4    TATA            5.6       -1  n.a.      ratan tata
```

3.7 Write into exel file:

[36]:
```
df.to_excel("new.xlsx" , sheet_name="stock")     # create a new.xlsx file
df.to_excel("new.xlsx" , sheet_name="stock" , startrow=4 , startcol=5)
                                                 # start with 4 rows and 5 column
```

3.8 Dealing with 2 data freame

[37]:
```
df_stocks = pd.DataFrame({
'tickers': ['GOOGL', 'WMT', 'MSFT'],
'price': [845, 65, 64 ],
'pe':[30.37, 14.26, 30.97],
'eps': [27.82, 4.61, 2.12]
})

df_weather = pd.DataFrame({
'day': ['1/1/2017','1/2/2017' ,'1/3/2017'],
'temperature': [32, 35, 28],
'event': ['Rain', 'Sunny', 'Snow']
})
df
```

[37]:
```
  tickers            eps  revenue price          people
0   GOOGL          27.82       87   845      larry page
1     WMT           4.61      484    65  Unkonwn People
2    MSFT             -1       85    64      bill gates
3     RIL   not available       50  1023   mukesh ambani
4    TATA            5.6       -1  n.a.      ratan tata
```

[38]:
```
with pd.ExcelWriter('stocks_weather.xlsx') as writer:
    df_stocks.to_excel(writer, sheet_name="stocks")
    df_weather.to_excel(writer, sheet_name="weather" )
df
```

[38]:
```
  tickers            eps  revenue price          people
0   GOOGL          27.82       87   845      larry page
1     WMT           4.61      484    65  Unkonwn People
2    MSFT             -1       85    64      bill gates
3     RIL   not available       50  1023   mukesh ambani
4    TATA            5.6       -1  n.a.      ratan tata
```

4 Missing data handaling

4.1 fillna():

[39]:
```
df = pd.read_csv("data.csv" , parse_dates=["day"])      # replace day column to␣
 ↪date
new_df = df.fillna(0)       # all NaN value will replace with 0
new_df
```

[39]:

	day	temperature	windspeed	event
0	2017-01-01	32.0	6.0	Rain
1	2017-01-04	0.0	9.0	Sunny
2	2017-01-05	28.0	0.0	Snow
3	2017-01-06	0.0	7.0	0
4	2017-01-07	32.0	0.0	Rain
5	2017-01-08	0.0	0.0	Sunny
6	2017-01-09	0.0	0.0	0
7	2017-01-10	34.0	8.0	Cloudy
8	2017-01-11	40.0	12.0	Sunny

[40]:
```
new_df = df.fillna({
    "temperature" : 0,      # replace NaN vlaues in individual column with␣
 ↪individual values
    "windspeed" : 0 ,
    "event" : "no event"
})
new_df
```

[40]:

	day	temperature	windspeed	event
0	2017-01-01	32.0	6.0	Rain
1	2017-01-04	0.0	9.0	Sunny
2	2017-01-05	28.0	0.0	Snow
3	2017-01-06	0.0	7.0	no event
4	2017-01-07	32.0	0.0	Rain
5	2017-01-08	0.0	0.0	Sunny
6	2017-01-09	0.0	0.0	no event
7	2017-01-10	34.0	8.0	Cloudy
8	2017-01-11	40.0	12.0	Sunny

[41]:
```
new_df = df.fillna(method="ffill")      # ffill --> forword fill it will␣
 ↪replace NaN values with previous row values
new_df
```

[41]:

	day	temperature	windspeed	event
0	2017-01-01	32.0	6.0	Rain
1	2017-01-04	32.0	9.0	Sunny
2	2017-01-05	28.0	9.0	Snow

```
3 2017-01-06         28.0         7.0    Snow
4 2017-01-07         32.0         7.0    Rain
5 2017-01-08         32.0         7.0   Sunny
6 2017-01-09         32.0         7.0   Sunny
7 2017-01-10         34.0         8.0  Cloudy
8 2017-01-11         40.0        12.0   Sunny
```

[42]:
```
new_df = df.fillna(method="bfill")      # bfill --> Back fill it will replace␣
↪NaN values with Next row values
new_df
```

[42]:
```
         day  temperature  windspeed   event
0 2017-01-01         32.0        6.0    Rain
1 2017-01-04         28.0        9.0   Sunny
2 2017-01-05         28.0        7.0    Snow
3 2017-01-06         32.0        7.0    Rain
4 2017-01-07         32.0        8.0    Rain
5 2017-01-08         34.0        8.0   Sunny
6 2017-01-09         34.0        8.0  Cloudy
7 2017-01-10         34.0        8.0  Cloudy
8 2017-01-11         40.0       12.0   Sunny
```

[43]:
```
new_df = df.fillna(method="bfill" , axis="columns")      # bfill --> Back fill␣
↪it will replace NaN values with Next column values
new_df
```

[43]:
```
         day temperature windspeed   event
0 2017-01-01        32.0       6.0    Rain
1 2017-01-04         9.0       9.0   Sunny
2 2017-01-05        28.0      Snow    Snow
3 2017-01-06         7.0       7.0     NaN
4 2017-01-07        32.0      Rain    Rain
5 2017-01-08       Sunny     Sunny   Sunny
6 2017-01-09         NaT       NaT     NaT
7 2017-01-10        34.0       8.0  Cloudy
8 2017-01-11        40.0      12.0   Sunny
```

[44]:
```
new_df = df.fillna(method="ffill" , limit=1)      # forword fill 1 times for␣
↪every values
new_df
```

[44]:
```
         day  temperature  windspeed   event
0 2017-01-01         32.0        6.0    Rain
1 2017-01-04         32.0        9.0   Sunny
2 2017-01-05         28.0        9.0    Snow
3 2017-01-06         28.0        7.0    Snow
4 2017-01-07         32.0        7.0    Rain
```

```
5 2017-01-08         32.0        NaN   Sunny
6 2017-01-09          NaN        NaN   Sunny
7 2017-01-10         34.0        8.0  Cloudy
8 2017-01-11         40.0       12.0   Sunny
```

4.2 interpolate():

[45]:
```
df = pd.read_csv("data.csv")
new_df = df.interpolate() # Replace NULL values with the number of middle␣
 ↪values of between the previous and next row
new_df
# method("time")        --> will consider the time while insert NaN values
```

[45]:

	day	temperature	windspeed	event
0	1/1/2017	32.000000	6.00	Rain
1	1/4/2017	30.000000	9.00	Sunny
2	1/5/2017	28.000000	8.00	Snow
3	1/6/2017	30.000000	7.00	NaN
4	1/7/2017	32.000000	7.25	Rain
5	1/8/2017	32.666667	7.50	Sunny
6	1/9/2017	33.333333	7.75	NaN
7	1/10/2017	34.000000	8.00	Cloudy
8	1/11/2017	40.000000	12.00	Sunny

4.3 dropna():

[46]:
```
new_df = df.dropna()                # it will drop all NaN value row
new_df
```

[46]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

[47]:
```
new_df = df.dropna(how="all")       # when all values are NaN only this row␣
 ↪will drop
new_df
```

[47]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN

```
7  1/10/2017         34.0        8.0  Cloudy
8  1/11/2017         40.0       12.0   Sunny
```

[48]:
```
new_df = df.dropna(thresh=2)        # If i have at list 1 valid values keep the␣
  ↪row
new_df
```

[48]:
```
          day  temperature  windspeed   event
0   1/1/2017         32.0        6.0    Rain
1   1/4/2017          NaN        9.0   Sunny
2   1/5/2017         28.0        NaN    Snow
3   1/6/2017          NaN        7.0     NaN
4   1/7/2017         32.0        NaN    Rain
5   1/8/2017          NaN        NaN   Sunny
7  1/10/2017         34.0        8.0  Cloudy
8  1/11/2017         40.0       12.0   Sunny
```

4.4 Date_range():

[49]:
```
n_df = pd.read_csv("data.csv" , parse_dates=["day"])
n_df
```

[49]:
```
          day  temperature  windspeed   event
0 2017-01-01         32.0        6.0    Rain
1 2017-01-04          NaN        9.0   Sunny
2 2017-01-05         28.0        NaN    Snow
3 2017-01-06          NaN        7.0     NaN
4 2017-01-07         32.0        NaN    Rain
5 2017-01-08          NaN        NaN   Sunny
6 2017-01-09          NaN        NaN     NaN
7 2017-01-10         34.0        8.0  Cloudy
8 2017-01-11         40.0       12.0   Sunny
```

[50]:
```
dt = pd.date_range("01-01-2017" , "01-11-2017")
dt
```

[50]:
```
DatetimeIndex(['2017-01-01', '2017-01-02', '2017-01-03', '2017-01-04',
               '2017-01-05', '2017-01-06', '2017-01-07', '2017-01-08',
               '2017-01-09', '2017-01-10', '2017-01-11'],
              dtype='datetime64[ns]', freq='D')
```

[51]:
```
index = pd.DatetimeIndex(dt)
n_df = n_df.reindex(index)
n_df
```

[51]:

	day	temperature	windspeed	event
2017-01-01	NaT	NaN	NaN	NaN
2017-01-02	NaT	NaN	NaN	NaN
2017-01-03	NaT	NaN	NaN	NaN
2017-01-04	NaT	NaN	NaN	NaN
2017-01-05	NaT	NaN	NaN	NaN
2017-01-06	NaT	NaN	NaN	NaN
2017-01-07	NaT	NaN	NaN	NaN
2017-01-08	NaT	NaN	NaN	NaN
2017-01-09	NaT	NaN	NaN	NaN
2017-01-10	NaT	NaN	NaN	NaN
2017-01-11	NaT	NaN	NaN	NaN

5 Handaling Missing data

[52]:
```
import numpy as np
df = pd.read_csv("weather.csv")
df
```

[52]:

	day	temperature	windspeed	event
0	1/1/2017	32 C	6 mah	Rain
1	1/2/2017	-99999	7	Sunny
2	1/3/2017	28	-99999	Snow
3	1/4/2017	-99999	7	0
4	1/5/2017	32	-99999	Rain
5	1/6/2017	31	2	Sunny
6	1/6/2017	34	5	0

5.1 Replace data:

[53]:
```
df = pd.read_csv("weather.csv")
new_df = df.replace(["-99999"], np.nan ,regex=True)    # replace -99999 with␣
↪NaN
new_df
```

[53]:

	day	temperature	windspeed	event
0	1/1/2017	32 C	6 mah	Rain
1	1/2/2017	NaN	7	Sunny
2	1/3/2017	28	NaN	Snow
3	1/4/2017	NaN	7	0
4	1/5/2017	32	NaN	Rain
5	1/6/2017	31	2	Sunny
6	1/6/2017	34	5	0

5.2 Replace bashed on column:

[54]:
```
new_df = df.replace({
    "temperature" : "-99999",      # column wise replace
    "windspeed" : "-99999",        # -99999 string format in file
    "event" : "0"
} , np.nan)
new_df
```

[54]:

	day	temperature	windspeed	event
0	1/1/2017	32 C	6 mah	Rain
1	1/2/2017	NaN	7	Sunny
2	1/3/2017	28	NaN	Snow
3	1/4/2017	NaN	7	NaN
4	1/5/2017	32	NaN	Rain
5	1/6/2017	31	2	Sunny
6	1/6/2017	34	5	NaN

5.3 Replace beshed on value:

[55]:
```
new_df = df.replace({           # value wise replace
    "-99999" : np.nan,
    "0" : "Zero"
})
new_df
```

[55]:

	day	temperature	windspeed	event
0	1/1/2017	32 C	6 mah	Rain
1	1/2/2017	NaN	7	Sunny
2	1/3/2017	28	NaN	Snow
3	1/4/2017	NaN	7	Zero
4	1/5/2017	32	NaN	Rain
5	1/6/2017	31	2	Sunny
6	1/6/2017	34	5	Zero

5.4 Replace with regex:

[56]:
```
new_df = df.replace({
    "temperature" : "[A-Za-z]",
    "windspeed" : "[A-Za-z]"
},"" , regex=True)    # remoce c , mah etc
new_df
```

[56]:

	day	temperature	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	-99999	7	Sunny
2	1/3/2017	28	-99999	Snow

```
3  1/4/2017      -99999         7      0
4  1/5/2017          32    -99999   Rain
5  1/6/2017          31         2  Sunny
6  1/6/2017          34         5      0
```

5.5 Replace the list of values to another list of values:

[57]:
```
df = pd.DataFrame({
'score': ['exceptional','average', 'good', 'poor', 'average', 'exceptional'],
'student': ['rob', 'maya', 'parthiv', 'tom', 'julian', 'erica']
})
df
```

[57]:
```
          score  student
0  exceptional      rob
1      average     maya
2         good  parthiv
3         poor      tom
4      average   julian
5  exceptional    erica
```

[58]:
```
new_df = df.replace(["poor" , "average" , "good" , "exceptional"] , [1, 2,␣
↪3,4]) # replace 1 list to another
new_df
```

[58]:
```
   score  student
0      4      rob
1      2     maya
2      3  parthiv
3      1      tom
4      2   julian
5      4    erica
```

6 Pandas group by data

6.1 Data Frame groupby:

[59]:
```
df = pd.read_csv("weather_by_cities.csv")
df
```

[59]:
```
          day      city  temperature  windspeed   event
0   1/1/2017  new york           32          6    Rain
1   1/2/2017  new york           36          7   Sunny
2   1/3/2017  new york           28         12    Snow
3   1/4/2017  new york           33          7   Sunny
4   1/1/2017    mumbai           90          5   Sunny
```

```
5   1/2/2017    mumbai          85          12     Fog
6   1/3/2017    mumbai          87          15     Fog
7   1/4/2017    mumbai          92           5    Rain
8   1/1/2017     paris          45          20   Sunny
9   1/2/2017     paris          50          13  Cloudy
10  1/3/2017     paris          54           8  Cloudy
11  1/4/2017     paris          42          10  Cloudy
```

[60]:
```
g = df.groupby("city")          # dataframe groupby object
g
```

[60]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fe1d7858700>
```

[61]:
```
for i , j in g:
    print(i)        # i is city name
    print(j)        # j is data frame
```

```
mumbai
        day    city  temperature  windspeed  event
4  1/1/2017  mumbai           90          5  Sunny
5  1/2/2017  mumbai           85         12    Fog
6  1/3/2017  mumbai           87         15    Fog
7  1/4/2017  mumbai           92          5   Rain
new york
        day      city  temperature  windspeed  event
0  1/1/2017  new york           32          6   Rain
1  1/2/2017  new york           36          7  Sunny
2  1/3/2017  new york           28         12   Snow
3  1/4/2017  new york           33          7  Sunny
paris
         day   city  temperature  windspeed   event
8   1/1/2017  paris           45         20   Sunny
9   1/2/2017  paris           50         13  Cloudy
10  1/3/2017  paris           54          8  Cloudy
11  1/4/2017  paris           42         10  Cloudy
```

[62]:
```
mumbai = g.get_group("mumbai")  # mumbai data frame is hare
mumbai
```

[62]:
```
        day    city  temperature  windspeed  event
4  1/1/2017  mumbai           90          5  Sunny
5  1/2/2017  mumbai           85         12    Fog
6  1/3/2017  mumbai           87         15    Fog
7  1/4/2017  mumbai           92          5   Rain
```

[63]:
```
g.max()                         # new datafeame with max value
```

[63]:
```
              day  temperature  windspeed  event
city
mumbai    1/4/2017           92         15  Sunny
new york  1/4/2017           36         12  Sunny
paris     1/4/2017           54         20  Sunny
```

[64]:
```
g.mean()                        # mean of 3 data set will come hare a new␣
↪dataset
```

[64]:
```
          temperature  windspeed
city
mumbai          88.50       9.25
new york        32.25       8.00
paris           47.75      12.75
```

[65]:
```
g.describe()                    # describe many things of 3 different dataframe
```

[65]:
```
         temperature                                                    \
               count   mean       std   min    25%   50%    75%   max
city
mumbai           4.0  88.50  3.109126  85.0  86.50  88.5  90.50  92.0
new york         4.0  32.25  3.304038  28.0  31.00  32.5  33.75  36.0
paris            4.0  47.75  5.315073  42.0  44.25  47.5  51.00  54.0

         windspeed
             count   mean       std  min   25%   50%    75%   max
city
mumbai         4.0   9.25  5.057997  5.0  5.00   8.5  12.75  15.0
new york       4.0   8.00  2.708013  6.0  6.75   7.0   8.25  12.0
paris          4.0  12.75  5.251984  8.0  9.50  11.5  14.75  20.0
```

7 Pandas concat

7.1 Concatinate:

[66]:
```
india_weather = pd.DataFrame({
    "city" : ["mombai" , "dehli" , "banglore"] ,
    "temp" : [32 , 45 , 30],
    "humidity" : [80 , 60 , 78]
})
usa_weather = pd.DataFrame({
    "city" : ["new york" , "chicago" , "orlando"] ,
    "temp" : [21 , 14 , 35],
    "humidity" : [68 , 65 , 75]
})
df = pd.concat([india_weather , usa_weather])   # concat these two data frame
df
```

[66]:
```
        city  temp  humidity
0     mombai    32        80
1      dehli    45        60
2   banglore    30        78
0   new york    21        68
1    chicago    14        65
2    orlando    35        75
```

[67]:
```
df = pd.concat([india_weather , usa_weather] , ignore_index=True)    # will␣
 ↪ignore index , default is False
df
```

[67]:
```
        city  temp  humidity
0     mombai    32        80
1      dehli    45        60
2   banglore    30        78
3   new york    21        68
4    chicago    14        65
5    orlando    35        75
```

[68]:
```
df = pd.concat([india_weather , usa_weather] , keys=["India" , "USA"])    # will␣
 ↪create adition index ["India" , "USA"]
df
```

[68]:
```
              city  temp  humidity
India 0     mombai    32        80
      1      dehli    45        60
      2   banglore    30        78
USA   0   new york    21        68
      1    chicago    14        65
      2    orlando    35        75
```

[69]:
```
india = df.loc["India"]      # access India indexed data frame
india
```

[69]:
```
        city  temp  humidity
0     mombai    32        80
1      dehli    45        60
2   banglore    30        78
```

7.2 Concar another axis:

[70]:
```
temp_df = pd.DataFrame({
    "city" : ["mombai" , "dehli" , "banglore"] ,
    "temp" : [34 , 53 , 43]
})
wind_df = pd.DataFrame({
```

```
    "city" : ["new york" , "chicago" , "orlando"] ,
    "temp" : [31 , 24 , 25]
})
df = pd.concat([temp_df , wind_df] , axis=1)     # default axis is 0
df
```

[70]:

	city	temp	city	temp
0	mombai	34	new york	31
1	dehli	53	chicago	24
2	banglore	43	orlando	25

7.3 Indexing wiht concat:

[71]:
```
temp_df = pd.DataFrame({
    "city" : ["mombai" , "dehli" , "banglore"] ,
    "temp" : [34 , 53 , 43]
},index=[0, 1, 2])              # defining index

wind_df = pd.DataFrame({
    "city" : ["delhi" , "mombai"] ,
    "temp" : [31 , 24 ]
},index=[1 , 0])                # defining index

df = pd.concat([temp_df , wind_df] , axis=1)
df
```

[71]:

	city	temp	city	temp
0	mombai	34	mombai	24.0
1	dehli	53	delhi	31.0
2	banglore	43	NaN	NaN

7.4 Join dataframe with series():

[72]:
```
s = pd.Series(["Humid" , "Dry" , "Rani"] , name="Event")
s
```

[72]:
```
0    Humid
1      Dry
2     Rani
Name: Event, dtype: object
```

[73]:
```
temp_df = pd.DataFrame({
    "city" : ["mombai" , "dehli" , "banglore"] ,
    "temp" : [34 , 53 , 43]
},index=[0, 1, 2])
```

```
new = pd.concat([temp_df , s] , axis=1)      # append new datafreame into temp_df
new
```

[73]:

	city	temp	Event
0	mombai	34	Humid
1	dehli	53	Dry
2	banglore	43	Rani

8 Marge Dataframe

8.1 Using marge():

[74]:
```
df1 = pd.DataFrame({
    "city" : ["new york" , "chicago" , "orlando"] ,
    "temp" : [34 , 53 , 43]
})
df2 = pd.DataFrame({
    "city" : ["new york" , "chicago" , "orlando"] ,
    "humidity" : [64 , 68 , 75]
})

# Using marge():
df3 = pd.merge(df1 , df2 , on="city")
df3
```

[74]:

	city	temp	humidity
0	new york	34	64
1	chicago	53	68
2	orlando	43	75

[75]:
```
df1 = pd.DataFrame({
    "city" : ["new york" , "chicago" , "dhaka"] ,
    "temp" : [34 , 53 , 43]
})
df2 = pd.DataFrame({
    "city" : ["new york" , "chicago" , "orlando"] ,
    "humidity" : [64 , 68 , 75]
})
df3 = pd.merge(df1 , df2 , on="city")   # different city will ignore . it is␣
 ↪semilar to set theory
df3
```

[75]:

	city	temp	humidity
0	new york	34	64
1	chicago	53	68

[76]:
```
df3 = pd.merge(df1 , df2 , on="city" , how="outer")      # like as union set and␣
 ↪inner jion is default
df3
```

[76]:
```
        city  temp  humidity
0  new york  34.0      64.0
1   chicago  53.0      68.0
2     dhaka  43.0       NaN
3   orlando   NaN      75.0
```

[77]:
```
df3 = pd.merge(df1 , df2 , on="city" , how="left")      #  take onley left␣
 ↪varible city
df3
```

[77]:
```
        city  temp  humidity
0  new york    34      64.0
1   chicago    53      68.0
2     dhaka    43       NaN
```

[78]:
```
df3 = pd.merge(df1 , df2 , on="city" , how="outer" , indicator=True)       # ␣
 ↪show marge indicator like left_only , both , right_only
df3
```

[78]:
```
        city  temp  humidity      _merge
0  new york  34.0      64.0        both
1   chicago  53.0      68.0        both
2     dhaka  43.0       NaN   left_only
3   orlando   NaN      75.0  right_only
```

8.2 suffixes:

[79]:
```
df1 = pd.DataFrame({
    "city": ["new york","chicago","orlando", "baltimore"],
    "temperature": [21,14,35,38],
    "humidity": [65,68,71, 75]
})

df2 = pd.DataFrame({
    "city": ["chicago","new york","san diego"],
    "temperature": [21,14,35],
    "humidity": [65,68,71]
})

df3 = pd.merge(df1 , df2 , on="city")       # it will append suffixes into␣
 ↪header
df3
```

[79]:
```
        city  temperature_x  humidity_x  temperature_y  humidity_y
0  new york             21          65             14          68
1   chicago             14          68             21          65
```

[80]:
```
df3 = pd.merge(df1 , df2 , on="city" , suffixes=("_left" , "_right"))       #␣
 ↪create own suffixes
df3
```

[80]:
```
        city  temperature_left  humidity_left  temperature_right  \
0  new york                21             65                 14
1   chicago                14             68                 21

   humidity_right
0              68
1              65
```

9 pivot and pivot tabel

[81]:
```
# pivot allaws you to reshape and tranform your data
df = pd.read_csv("pivot_weather.csv")
df
```

[81]:
```
        date      city  temperature  humidity
0  5/1/2017  new york           65        56
1  5/2/2017  new york           66        58
2  5/3/2017  new york           68        60
3  5/1/2017    mumbai           75        80
4  5/2/2017    mumbai           78        83
5  5/3/2017    mumbai           82        85
6  5/1/2017   beijing           80        26
7  5/2/2017   beijing           77        30
8  5/3/2017   beijing           79        35
```

[82]:
```
n_df = df.pivot(index="date" , columns="city")  # tranform as index date and␣
 ↪city column
n_df
```

[82]:
```
         temperature                 humidity
city         beijing mumbai new york  beijing mumbai new york
date
5/1/2017          80     75       65       26     80       56
5/2/2017          77     78       66       30     83       58
5/3/2017          79     82       68       35     85       60
```

[83]:
```
n_df = df.pivot(index="date" , columns="city" , values="humidity")  # print␣
 ↪only humidity
```

```
n_df
```

[83]:
```
city       beijing  mumbai  new york
date
5/1/2017        26      80        56
5/2/2017        30      83        58
5/3/2017        35      85        60
```

9.1 pivot tabel:

[84]:
```
# pivot table is used to sammarize ang aggregate data inside dataframe
new_df = df.pivot_table(index="city" , columns="date")  # create a new table
new_df
```

[84]:
```
          humidity                    temperature
date     5/1/2017 5/2/2017 5/3/2017    5/1/2017 5/2/2017 5/3/2017
city
beijing        26       30       35          80       77       79
mumbai         80       83       85          75       78       82
new york       56       58       60          65       66       68
```

[85]:
```
new_df = df.pivot_table(index="city" , columns="date" ,aggfunc=sum)  # is not␣
↪work until call (margins = True)
new_df
```

[85]:
```
          humidity                    temperature
date     5/1/2017 5/2/2017 5/3/2017    5/1/2017 5/2/2017 5/3/2017
city
beijing        26       30       35          80       77       79
mumbai         80       83       85          75       78       82
new york       56       58       60          65       66       68
```

[86]:
```
new_df = df.pivot_table(index="city" , columns="date" ,aggfunc=sum ,␣
↪margins=True)
new_df
```

[86]:
```
          humidity                          temperature
date     5/1/2017 5/2/2017 5/3/2017  All    5/1/2017 5/2/2017 5/3/2017  All
city
beijing        26       30       35   91          80       77       79  236
mumbai         80       83       85  248          75       78       82  235
new york       56       58       60  174          65       66       68  199
All           162      171      180  513         220      221      229  670
```

9.2 Use grouper function to aggregate:

[87]:
```
df["date"] = pd.to_datetime(df["date"])
x = df.pivot_table(index=pd.Grouper(freq="M" , key="date") , columns="city")
  ↪# default is average
x
```

[87]:

	humidity			temperature		
city	beijing	mumbai	new york	beijing	mumbai	new york
date						
2017-05-31	30.333333	82.666667	58.0	78.666667	78.333333	66.333333