PANDAS

Introduction

- Pandas is an open- source python library for data analysis and manipulation. It provides data structures like Series and DataFrame for handling structured data .

Features

Installation and Import

!pip install pandas

import pandas as pd

Series

- A one dimensional labeled array capable of holding any data type (integer,float,string etc..)

In [2]:
```
import pandas  as pd
import numpy as np
```

In [12]:
```
s = pd.Series ([1,2,3,4,"mahesh",23.90])
```

In [13]:
```
s
```

Out[13]:
```
0         1
1         2
2         3
3         4
4    mahesh
5      23.9
dtype: object
```

In [14]:
```
names = ["mahesh","charam","maha","charan","mahi"]
names = pd.Series(names,index = [101,102,103,104,105])
names
```

Out[14]:
```
101    mahesh
102    charam
103      maha
104    charan
105      mahi
dtype: object
```

DataFrame

- A two dimensional labeled data structure with columns of potentially different types
  .

In [15]:
```
## cretae data frame  ##column wise
## from dictionary
ipl = {"player" :["virat","mahi","pant","siraj"],
      "team":["rcb","mi","mi","mi"],
      "age":[34,54,32,43]}
```

In [16]:
```
pd.DataFrame(ipl)
```

Out[16]:

	player	team	age
0	virat	rcb	34
1	mahi	mi	54
2	pant	mi	32
3	siraj	mi	43

In [17]:
```
##rowwise dataframe
fruits = [["apple","red",100],
      ["mango","yellow",30],
      ["grapes","black",90],
      ["dragon","pink",80]]
pd.DataFrame(fruits,columns = ["name","colors","price"])
```

Out[17]:

	name	colors	price
0	apple	red	100
1	mango	yellow	30
2	grapes	black	90
3	dragon	pink	80

DataInput / DataOutput

CSV file

- Reading :
- df =pd.read_csv(r"filename.csv")

Excel

- Reading
- df = pd.read_excel(r"filename.xlsx")

Json

- Reading
- df = pd.read_json(r"filename.json")

Example of opening csv file

In [18]:
```
df=pd.read_csv(r"C:\Users\Meharaj\loandata.csv")
```

In [19]:
```
df
```

Out[19]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInco
0	LP001722	Male	Yes	0	Graduate	No	
1	LP002502	Female	Yes	2	Not Graduate	NaN	
2	LP002949	Female	No	3+	Graduate	NaN	
3	LP002603	Female	No	0	Graduate	No	
4	LP001644	NaN	Yes	0	Graduate	Yes	
...	...	...	...	...	...	...	
609	LP001640	Male	Yes	0	Graduate	Yes	39
610	LP001536	Male	Yes	3+	Graduate	No	39
611	LP001585	NaN	Yes	3+	Graduate	No	51
612	LP002101	Male	Yes	0	Graduate	NaN	63
613	LP002317	Male	Yes	3+	Graduate	No	81

614 rows × 13 columns

Data exploration

- view first five rows.

In [20]:
```
df.head() #no of rows default = 5
```

Out[20]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001722	Male	Yes	0	Graduate	No	150
1	LP002502	Female	Yes	2	Not Graduate	NaN	210
2	LP002949	Female	No	3+	Graduate	NaN	416
3	LP002603	Female	No	0	Graduate	No	645
4	LP001644	NaN	Yes	0	Graduate	Yes	674

- view last (bottom) fiverows

In [21]:
```
df.tail()
```

Out[21]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInco
609	LP001640	Male	Yes	0	Graduate	Yes	39
610	LP001536	Male	Yes	3+	Graduate	No	39
611	LP001585	NaN	Yes	3+	Graduate	No	51
612	LP002101	Male	Yes	0	Graduate	NaN	63
613	LP002317	Male	Yes	3+	Graduate	No	81

Shapem of the dataset

- It can show how many number of columns and rows are present in the given dataset

In [22]:
```
df.shape
```

Out[22]:
```
(614, 13)
```

Columns Names

In [23]:
```
df.columns
```

Out[23]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

Dataset information

In [24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

Summary statistics

In [25]:
```
df.describe()
```

Out[25]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_H
count	614.000000	614.000000	592.000000	600.00000	564.(
mean	5403.459283	1621.245798	146.412162	342.00000	0.8
std	6109.041673	2926.248369	85.587325	65.12041	0.3
min	150.000000	0.000000	9.000000	12.00000	0.(
25%	2877.500000	0.000000	100.000000	360.00000	1.(
50%	3812.500000	1188.500000	128.000000	360.00000	1.(
75%	5795.000000	2297.250000	168.000000	360.00000	1.(
max	81000.000000	41667.000000	700.000000	480.00000	1.(

In [26]:
```
df.describe(include = 'all',percentiles = [0,0.5])
# It describes the both categorical and numerical data of the dataset.
# It gives the statistical measures of the data.
```

Out[26]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantI
count	614	601	611	599	614	582	614.(
unique	614	2	2	4	2	2	
top	LP002317	Male	Yes	0	Graduate	No	
freq	1	489	398	345	480	500	
mean	NaN	NaN	NaN	NaN	NaN	NaN	5403.4
std	NaN	NaN	NaN	NaN	NaN	NaN	6109.(
min	NaN	NaN	NaN	NaN	NaN	NaN	150.(
0%	NaN	NaN	NaN	NaN	NaN	NaN	150.(
50%	NaN	NaN	NaN	NaN	NaN	NaN	3812.5
max	NaN	NaN	NaN	NaN	NaN	NaN	81000.(

Datatype

In [27]:
```
df.dtypes
```

Out[27]:
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

Accessing columns

In [28]:
```
df.Gender
```

Out[28]:
```
0        Male
1      Female
2      Female
3      Female
4         NaN
       ...
609      Male
610      Male
611       NaN
612      Male
613      Male
Name: Gender, Length: 614, dtype: object
```

In [29]:
```
df.Dependents
```

Out[29]:
```
0       0
1       2
2      3+
3       0
4       0
       ..
609     0
610    3+
611    3+
612     0
613    3+
Name: Dependents, Length: 614, dtype: object
```

In [30]:
```
type(df.CoapplicantIncome)
```

Out[30]:
```
pandas.core.series.Series
```

In [31]:
```
df["ApplicantIncome"]    ##AS series
```

Out[31]:
```
0        150
1        210
2        416
3        645
4        674
       ...
609    39147
610    39999
611    51763
612    63337
613    81000
Name: ApplicantIncome, Length: 614, dtype: int64
```

In [32]:
```
df[["ApplicantIncome"]]   ##as dataframe
```

Out[32]:

	ApplicantIncome
0	150
1	210
2	416
3	645
4	674
...	...
609	39147
610	39999
611	51763
612	63337
613	81000

614 rows × 1 columns

In [33]:
```
df[["ApplicantIncome","LoanAmount","Gender"]]  ###we can access multiple columns
```

Out[33]:

	ApplicantIncome	LoanAmount	Gender
0	150	135.0	Male
1	210	98.0	Female
2	416	350.0	Female
3	645	113.0	Female
4	674	168.0	NaN
...	...	...	...
609	39147	120.0	Male
610	39999	600.0	Male
611	51763	700.0	NaN
612	63337	490.0	Male
613	81000	360.0	Male

614 rows × 3 columns

In [34]:
```
df.at[10,"Gender"]   ## access a single value for a row /column label pair
```

Out[34]:
```
'Male'
```

In [35]:
```
df.iat[2,3]    ##index base
```

Out[35]:
```
'3+'
```

In [36]:
```
## we can assign (update)the data also
df.iat[2,3] = 4
```

In [37]:
```
df.iat[2,3]
```

Out[37]:
```
4
```

Check for missing values

In [38]:
```
df.isna().sum()
```

Out[38]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

select rows by index

In [39]:
```
df.iloc[4,2]   ### indexing wise
```

Out[39]:
```
'Yes'
```

In [40]:
```
df.head()
```

Out[40]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001722	Male	Yes	0	Graduate	No	150
1	LP002502	Female	Yes	2	Not Graduate	NaN	210
2	LP002949	Female	No	4	Graduate	NaN	416
3	LP002603	Female	No	0	Graduate	No	645
4	LP001644	NaN	Yes	0	Graduate	Yes	674

In [41]:
```
df.loc[3,"Gender"]    ## name the labeling to slice
```

Out[41]:
```
'Female'
```

unique values

In [42]:
```
df["Gender"].unique()
```

Out[42]:
```
array(['Male', 'Female', nan], dtype=object)
```

Statisticalmeasures

In [43]:
```
df["ApplicantIncome"].mean()
```

Out[43]:
```
np.float64(5403.459283387622)
```

In [44]:
```
df["ApplicantIncome"].median()
```

Out[44]:
```
np.float64(3812.5)
```

In [45]:
```
df["Gender"].mode()  ## most  repeated value
```

Out[45]:
```
0    Male
Name: Gender, dtype: object
```

In [46]:
```
df["ApplicantIncome"].mean()
```

Out[46]:
```
np.float64(5403.459283387622)
```

In [47]:
```
df["ApplicantIncome"].skew()
```

Out[47]:
```
np.float64(6.539513113994621)
```

In [48]:
```
df["ApplicantIncome"].max()
```

Out[48]: np.int64(81000)

In [49]:
```
df["ApplicantIncome"].min()
```

Out[49]: np.int64(150)

In [50]:
```
df["ApplicantIncome"].quantile(0.25)
```

Out[50]: np.float64(2877.5)

In [51]:
```
df["ApplicantIncome"].quantile(0.5)
```

Out[51]: np.float64(3812.5)

In [52]:
```
df["ApplicantIncome"].quantile(0.75)
```

Out[52]: np.float64(5795.0)

In [53]:
```
df["ApplicantIncome"].var()
```

Out[53]: np.float64(37320390.16718121)

In [54]:
```
df["ApplicantIncome"].std()
```

Out[54]: np.float64(6109.041673387178)

In [55]:
```
df.describe()
```

Out[55]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_H
count	614.000000	614.000000	592.000000	600.00000	564.(
mean	5403.459283	1621.245798	146.412162	342.00000	0.8
std	6109.041673	2926.248369	85.587325	65.12041	0.3
min	150.000000	0.000000	9.000000	12.00000	0.(
25%	2877.500000	0.000000	100.000000	360.00000	1.(
50%	3812.500000	1188.500000	128.000000	360.00000	1.(
75%	5795.000000	2297.250000	168.000000	360.00000	1.(
max	81000.000000	41667.000000	700.000000	480.00000	1.(

Data Manipulation

Add a new column

In [56]:
```
df["Genderwise _education"] = df["Gender"] + df["Education"]
```

In [57]:
```
df.head()
```

Out[57]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001722	Male	Yes	0	Graduate	No	150
1	LP002502	Female	Yes	2	Not Graduate	NaN	210
2	LP002949	Female	No	4	Graduate	NaN	416
3	LP002603	Female	No	0	Graduate	No	645
4	LP001644	NaN	Yes	0	Graduate	Yes	674

Drop column

In [58]:
```
df = df.drop(["Genderwise _education"],axis = 1)
```

In [59]:
```
df.head()
```

Out[59]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001722	Male	Yes	0	Graduate	No	150
1	LP002502	Female	Yes	2	Not Graduate	NaN	210
2	LP002949	Female	No	4	Graduate	NaN	416
3	LP002603	Female	No	0	Graduate	No	645
4	LP001644	NaN	Yes	0	Graduate	Yes	674

Rename columns

In [60]:
```
df=df.rename(columns ={"Married" : "Married_status"})
```

In [61]:
```
df.head()
```

Out[61]:

	Loan_ID	Gender	Married_status	Dependents	Education	Self_Employed	Applicant
0	LP001722	Male	Yes	0	Graduate	No	
1	LP002502	Female	Yes	2	Not Graduate	NaN	
2	LP002949	Female	No	4	Graduate	NaN	
3	LP002603	Female	No	0	Graduate	No	
4	LP001644	NaN	Yes	0	Graduate	Yes	

In [62]:
```
df["Married_status"].value_counts()
```

Out[62]:
```
Married_status
Yes    398
No     213
Name: count, dtype: int64
```

In [63]:
```
sum((df["Gender"] == "Female")      & (df["Married_status"] == "Yes"))
```

Out[63]: 31

In [64]:
```
sum((df["Gender"] == "Male")      &  (df["Property_Area"] == "Rural"))
```

Out[64]: 151

In [65]:
```
df.drop(index = 0 , axis = 0 ,inplace = True)  ## inplace its delete the orginal
```

In [66]:
```
df.head()
```

Out[66]:

	Loan_ID	Gender	Married_status	Dependents	Education	Self_Employed	Applicant
1	LP002502	Female	Yes	2	Not Graduate	NaN	
2	LP002949	Female	No	4	Graduate	NaN	
3	LP002603	Female	No	0	Graduate	No	
4	LP001644	NaN	Yes	0	Graduate	Yes	
5	LP001259	Male	Yes	1	Graduate	Yes	

In [67]:
```
df.reset_index(drop = True,inplace = True)  ##get original data
```

In [68]:
```
df.head()
```

Out[68]:

	Loan_ID	Gender	Married_status	Dependents	Education	Self_Employed	Applicant
0	LP002502	Female	Yes	2	Not Graduate	NaN	
1	LP002949	Female	No	4	Graduate	NaN	
2	LP002603	Female	No	0	Graduate	No	
3	LP001644	NaN	Yes	0	Graduate	Yes	
4	LP001259	Male	Yes	1	Graduate	Yes	

pd.Concat

In [69]:
```
df = pd.DataFrame({"names" : ["mahesh","prabha","darling"],
                   "exp":[2,3,4,],
                   "sal": [10000,20000,30000]})
```

In [70]:
```
df1 = pd.DataFrame({"GEnder" : ["male","female","male"],
                    "mar":["un","ma","m"]})
```

In [71]:
```
df2 = pd.concat((df,df1),axis = 1)
```

In [72]:
```
df2
```

Out[72]:

	names	exp	sal	GEnder	mar
0	mahesh	2	10000	male	un
1	prabha	3	20000	female	ma
2	darling	4	30000	male	m

Merging the Data

In [73]:
```
df1  = pd.DataFrame({"Name":['Ramesh','Suresh','Prakash','Mahesh'],
             'Company':["IBM",'Google','IBM','Infosys'],
              'Country':['India','India','USA','UK'] })
df1
```

Out[73]:

	Name	Company	Country
0	Ramesh	IBM	India
1	Suresh	Google	India
2	Prakash	IBM	USA
3	Mahesh	Infosys	UK

In [74]:
```
df2 = pd.DataFrame({'Name':['Arjun','Akash','Arthi'],
                    'Company':['Accenture','IBM','Google'],
                    'Country':['India','India','India'],
                    'Salary' : [45000,50000,60000]})

df2
```

Out[74]:

	Name	Company	Country	Salary
0	Arjun	Accenture	India	45000
1	Akash	IBM	India	50000
2	Arthi	Google	India	60000

In [75]:
```
pd.merge(df1,df2, on = "Company" ,how="inner")
```

Out[75]:

	Name_x	Company	Country_x	Name_y	Country_y	Salary
0	Ramesh	IBM	India	Akash	India	50000
1	Suresh	Google	India	Arthi	India	60000
2	Prakash	IBM	USA	Akash	India	50000

In [76]:
```
pd.merge(df1,df2, on = "Country" ,how="inner")
```

Out[76]:

	Name_x	Company_x	Country	Name_y	Company_y	Salary
0	Ramesh	IBM	India	Arjun	Accenture	45000
1	Ramesh	IBM	India	Akash	IBM	50000
2	Ramesh	IBM	India	Arthi	Google	60000
3	Suresh	Google	India	Arjun	Accenture	45000
4	Suresh	Google	India	Akash	IBM	50000
5	Suresh	Google	India	Arthi	Google	60000

In [77]:
```
pd.merge(df1,df2, on = "Country" ,how="outer")
```

Out[77]:

	Name_x	Company_x	Country	Name_y	Company_y	Salary
0	Ramesh	IBM	India	Arjun	Accenture	45000.0
1	Ramesh	IBM	India	Akash	IBM	50000.0
2	Ramesh	IBM	India	Arthi	Google	60000.0
3	Suresh	Google	India	Arjun	Accenture	45000.0
4	Suresh	Google	India	Akash	IBM	50000.0
5	Suresh	Google	India	Arthi	Google	60000.0
6	Mahesh	Infosys	UK	NaN	NaN	NaN
7	Prakash	IBM	USA	NaN	NaN	NaN

In [78]:
```
pd.merge(df1,df2, on = "Country" ,how="right")
```

Out[78]:

	Name_x	Company_x	Country	Name_y	Company_y	Salary
0	Ramesh	IBM	India	Arjun	Accenture	45000
1	Suresh	Google	India	Arjun	Accenture	45000
2	Ramesh	IBM	India	Akash	IBM	50000
3	Suresh	Google	India	Akash	IBM	50000
4	Ramesh	IBM	India	Arthi	Google	60000
5	Suresh	Google	India	Arthi	Google	60000

In [79]:
```
pd.merge(df1,df2, on = "Country" ,how="left")
```

Out[79]:

	Name_x	Company_x	Country	Name_y	Company_y	Salary
0	Ramesh	IBM	India	Arjun	Accenture	45000.0
1	Ramesh	IBM	India	Akash	IBM	50000.0
2	Ramesh	IBM	India	Arthi	Google	60000.0
3	Suresh	Google	India	Arjun	Accenture	45000.0
4	Suresh	Google	India	Akash	IBM	50000.0
5	Suresh	Google	India	Arthi	Google	60000.0
6	Prakash	IBM	USA	NaN	NaN	NaN
7	Mahesh	Infosys	UK	NaN	NaN	NaN

In [80]:
```
df2.sort_values(by = "Salary",ascending = False,ignore_index = True,inplace = T
```

In [81]:
```
df2
```

Out[81]:

	Name	Company	Country	Salary
0	Arthi	Google	India	60000
1	Akash	IBM	India	50000
2	Arjun	Accenture	India	45000

Time Series

In [128…
```
df = pd.read_csv(r"C:\Users\Meharaj\time_series.csv")
```

In [129…
```
df
```

Out[129…

	ID	Datetime	Count
0	0	25-08-2012 00:00	8
1	1	25-08-2012 01:00	2
2	2	25-08-2012 02:00	6
3	3	25-08-2012 03:00	2
4	4	25-08-2012 04:00	2
...	...	...	...
18283	18283	25-09-2014 19:00	868
18284	18284	25-09-2014 20:00	732
18285	18285	25-09-2014 21:00	702
18286	18286	25-09-2014 22:00	580
18287	18287	25-09-2014 23:00	534

18288 rows × 3 columns

In [130…
```
df["Datetime"]
```

Out[130…
```
0        25-08-2012 00:00
1        25-08-2012 01:00
2        25-08-2012 02:00
3        25-08-2012 03:00
4        25-08-2012 04:00
               ...
18283    25-09-2014 19:00
18284    25-09-2014 20:00
18285    25-09-2014 21:00
18286    25-09-2014 22:00
18287    25-09-2014 23:00
Name: Datetime, Length: 18288, dtype: object
```

In [92]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [131…
```
df2
```

Out[131…

	Name	Company	Country	Salary
0	Arthi	Google	India	60000
1	Akash	IBM	India	50000
2	Arjun	Accenture	India	45000

In [132…
```
df2["Salary"].nsmallest(1)
```

Out[132…
```
2    45000
Name: Salary, dtype: int64
```

In [133…
```
df2["Salary"].nlargest(1)
```

Out[133…
```
0    60000
Name: Salary, dtype: int64
```

Group by

In [87]:
```
df=pd.read_csv(r"C:\Users\Meharaj\loandata.csv")
```

In [88]:
```
df.head()
```

Out[88]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001722	Male	Yes	0	Graduate	No	150
1	LP002502	Female	Yes	2	Not Graduate	NaN	210
2	LP002949	Female	No	3+	Graduate	NaN	416
3	LP002603	Female	No	0	Graduate	No	645
4	LP001644	NaN	Yes	0	Graduate	Yes	674

In [93]:
```
df.groupby(by = "Gender")["ApplicantIncome"].agg([np.mean,np.max])
```

Out[93]:

	mean	max
Gender		
Female	4643.473214	19484
Male	5446.460123	81000

In [94]:
```
df.groupby(by = "Gender")[["ApplicantIncome","LoanAmount"]].agg([np.mean,np.max]
```

Out[94]:

	ApplicantIncome		LoanAmount	
	mean	max	mean	max
Gender				
Female	4643.473214	19484	126.697248	600.0
Male	5446.460123	81000	149.265957	650.0

In [95]:
```
## getting  the average income of loanaapplicant from different areas
df.groupby(by = "Property_Area")["ApplicantIncome"].agg(np.mean)
```

Out[95]:
```
Property_Area
Rural        5554.083799
Semiurban    5292.261803
Urban        5398.247525
Name: ApplicantIncome, dtype: float64
```

Pivot Table

In [96]:
```
pd.pivot_table(data = df,values = "ApplicantIncome",index = "Property_Area",aggf
```

Out[96]:

	mean	std
	ApplicantIncome	ApplicantIncome
Property_Area		
Rural	5554.083799	6782.658637
Semiurban	5292.261803	5279.629359
Urban	5398.247525	6392.928779

In [97]:
```
pd.pivot_table(data = df,values = ["ApplicantIncome","LoanAmount"],index = ["Gen
```

Out[97]:

		mean		std	
		ApplicantIncome	LoanAmount	ApplicantIncome	LoanAmount
Gender	Married				
Female	No	4503.787500	116.115385	3333.160790	57.781042
	Yes	4829.645161	153.322581	4152.831039	114.183883
Male	No	5236.146154	136.088000	4379.170994	81.771516
	Yes	5529.540616	154.011662	6743.209021	83.025254

MultiIndexing

In [98]:
```
outer = ["A","A","A","B","B","B "]
inner = [1,2,3,1,2,3]
list(zip(outer,inner))
```

Out[98]:
```
[('A', 1), ('A', 2), ('A', 3), ('B', 1), ('B', 2), ('B ', 3)]
```

In [99]:
```
multi= list(zip(outer,inner))
```

In [100…
```
index = pd.MultiIndex.from_tuples(multi)
```

In [106…
```
a=np.random.randint(1,100,6)
```

In [107…
```
df = pd.DataFrame(a,index = index)
```

In [103…
```
df
```

Out[103…

		0
A	**1**	17
	2	93
	3	53
B	**1**	26
	2	52
B	**3**	84

In [104…
```
df.loc["A"]
```

Out[104…

	0
1	17
2	93
3	53

In [105…
```
df.loc["B"].loc[2]
```

Out[105…
```
0    52
Name: 2, dtype: int32
```

Data Cleaning

Identify Missing values

In [3]:
```
df = pd.read_csv(r"C:\Users\Meharaj\missing values\SampleWeatherData.csv")
```

In [116…
```
df
```

Out[116…

	Time	Temperature	Humidity
0	2023-04-01	22.0	55.0
1	2023-04-02	NaN	NaN
2	2023-04-03	NaN	60.0
3	2023-04-04	25.0	NaN
4	2023-04-05	NaN	65.0
5	2023-04-06	28.0	NaN
6	2023-04-07	29.0	70.0
7	2023-04-08	NaN	NaN
8	2023-04-09	NaN	75.0
9	2023-04-10	30.0	NaN

In [117…
```
df.isna().sum()
```

Out[117…
```
Time           0
Temperature    5
Humidity       5
dtype: int64
```

In [118…
```
df.dropna()
```

Out[118…

	Time	Temperature	Humidity
0	2023-04-01	22.0	55.0
6	2023-04-07	29.0	70.0

In [112…
```
df.dropna(axis= 1)
```

Out[112…

	Loan_ID	Education	ApplicantIncome	CoapplicantIncome	Property_Area	Loan_St
0	LP001722	Graduate	150	1800.0	Rural	
1	LP002502	Not Graduate	210	2917.0	Semiurban	
2	LP002949	Graduate	416	41667.0	Urban	
3	LP002603	Graduate	645	3683.0	Rural	
4	LP001644	Graduate	674	5296.0	Rural	
…	…	…	…	…	…	
609	LP001640	Graduate	39147	4750.0	Semiurban	
610	LP001536	Graduate	39999	0.0	Semiurban	
611	LP001585	Graduate	51763	0.0	Urban	
612	LP002101	Graduate	63337	0.0	Urban	
613	LP002317	Graduate	81000	0.0	Rural	

614 rows × 6 columns

In [119…
```
df
```

Out[119…

	Time	Temperature	Humidity
0	2023-04-01	22.0	55.0
1	2023-04-02	NaN	NaN
2	2023-04-03	NaN	60.0
3	2023-04-04	25.0	NaN
4	2023-04-05	NaN	65.0
5	2023-04-06	28.0	NaN
6	2023-04-07	29.0	70.0
7	2023-04-08	NaN	NaN
8	2023-04-09	NaN	75.0
9	2023-04-10	30.0	NaN

In [120…
```
df["Temperature"].mean()
```

Out[120…
```
np.float64(26.8)
```

In [121…
```
df["Temperature"].fillna(df["Temperature"].mean(),inplace  = True)    ##using fil
```

In [122…
```
df
```

Out[122…

	Time	Temperature	Humidity
0	2023-04-01	22.0	55.0
1	2023-04-02	26.8	NaN
2	2023-04-03	26.8	60.0
3	2023-04-04	25.0	NaN
4	2023-04-05	26.8	65.0
5	2023-04-06	28.0	NaN
6	2023-04-07	29.0	70.0
7	2023-04-08	26.8	NaN
8	2023-04-09	26.8	75.0
9	2023-04-10	30.0	NaN

In [123…
```
df["Humidity"].median()
```

Out[123…
```
np.float64(65.0)
```

In [124…
```
df["Humidity"].fillna(df["Humidity"].median(),inplace = True)
```

In [125…
```
df
```

Out[125…

	Time	Temperature	Humidity
0	2023-04-01	22.0	55.0
1	2023-04-02	26.8	65.0
2	2023-04-03	26.8	60.0
3	2023-04-04	25.0	65.0
4	2023-04-05	26.8	65.0
5	2023-04-06	28.0	65.0
6	2023-04-07	29.0	70.0
7	2023-04-08	26.8	65.0
8	2023-04-09	26.8	75.0
9	2023-04-10	30.0	65.0

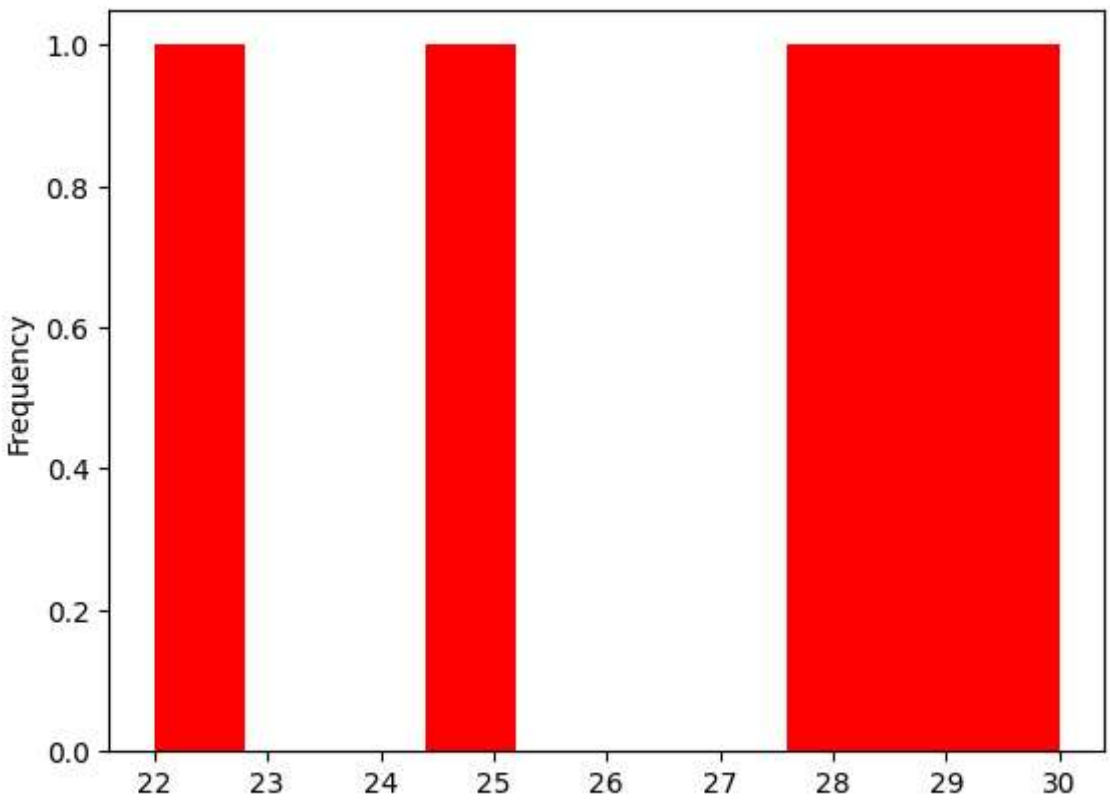
In [126…
```
df.isna().sum()
```

Out[126…
```
Time           0
Temperature    0
Humidity       0
dtype: int64
```

Data visualizations

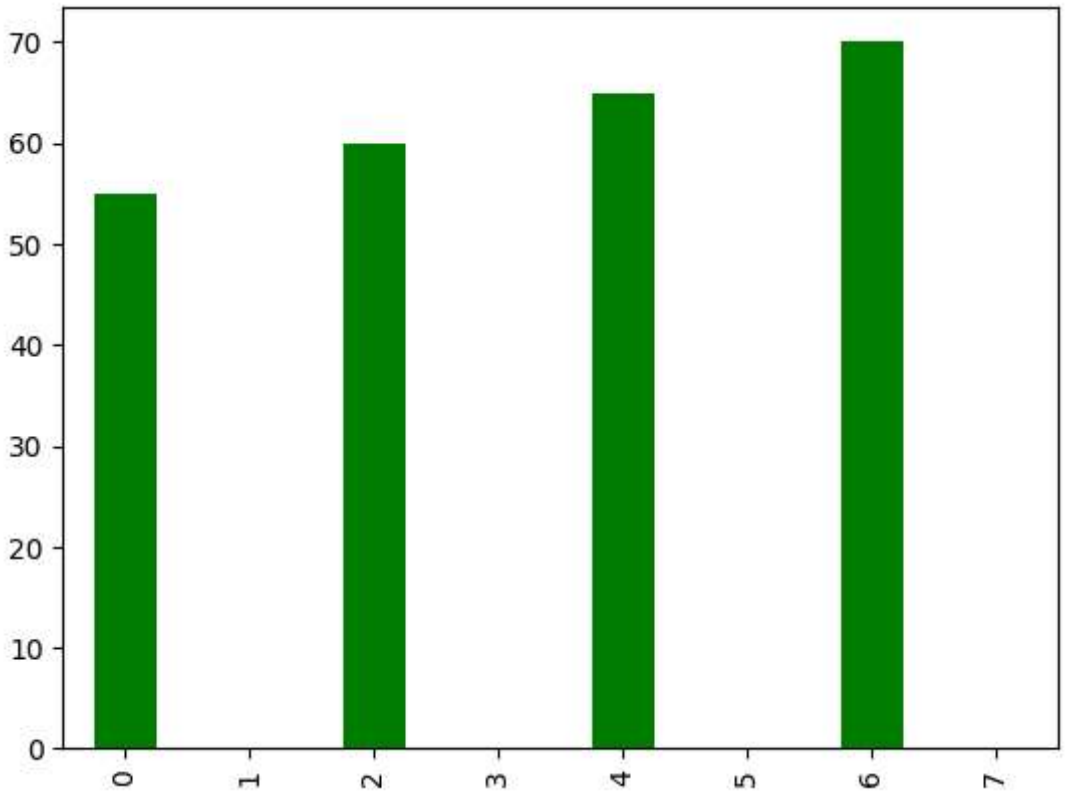
In [5]:
```
df["Temperature"].plot(kind = "hist",color = "red")
```

Out[5]:
```
<Axes: ylabel='Frequency'>
```



In [14]:
```
df["Humidity"].head(8).plot(kind = "bar",color  = "green")
```

Out[14]: <Axes: >



In [ ]: