Basic Operations using NumPy and Pandas

Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
```

Importing DataSet 1`

In [2]:

```
data=pd.read_csv(r"C:\Users\user\Downloads\fiat500_VehicleSelection_Dataset (1).csv")
data
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.6115
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.241
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.634
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.495
...	...	...	...	...	...	...	...	
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

1549 rows × 11 columns

Selecting first 7 rows using head()

In [3]:

```
data.head(7)
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lo
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.61155986
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.2418899
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.4178
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.6346092
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.4956502
5	6.0	pop	74.0	3623.0	70225.0	1.0	45.000702	7.6822700
6	7.0	lounge	51.0	731.0	11600.0	1.0	44.907242	8.61155986

Selecting Last 8 rows using tail()

In [4]:

```
data.tail(8)
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1541	NaN	NaN	NaN	NaN	NaN	NaN	NaN	countif	377
1542	NaN	NaN	NaN	NaN	NaN	NaN	NaN	sumif	4017825
1543	NaN	NaN	NaN	NaN	NaN	NaN	NaN	counta (not empty)	1538
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	length	5
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	concat	lonprice
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null values	NC
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	find	1
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	search	1

To get the Statistics data from the Table

In [5]:

```
data.describe()
```

Out[5]:

	ID	engine_power	age_in_days	km	previous_owners	la
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.00000(
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.54136
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.13351
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.85583
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.80299(
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.39409(
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.46796(
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.79561

Find Row and Column of the Table using shape

In [6]:

```
data.shape
```

Out[6]:

```
(1549, 11)
```

Find size of the table

In [7]:

```
data.size
```

Out[7]:

```
17039
```

Find Missing values using isna()

In [8]:

```
data.isna()
```

Out[8]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
1544	True	True	True	True	True	True	True	False	False
1545	True	True	True	True	True	True	True	False	False
1546	True	True	True	True	True	True	True	False	False
1547	True	True	True	True	True	True	True	False	False
1548	True	True	True	True	True	True	True	False	False

1549 rows × 11 columns

Fill the missing values using fillna(value=" ")

In [9]:

```
data.fillna(value=5)
```

Out[9]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.61155
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.2418
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.4
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.6346
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.4956
...	...	...	...	...	...	...	...	
1544	5.0	5	5.0	5.0	5.0	5.0	5.000000	l
1545	5.0	5	5.0	5.0	5.0	5.0	5.000000	c
1546	5.0	5	5.0	5.0	5.0	5.0	5.000000	Null v
1547	5.0	5	5.0	5.0	5.0	5.0	5.000000	
1548	5.0	5	5.0	5.0	5.0	5.0	5.000000	s

1549 rows × 11 columns

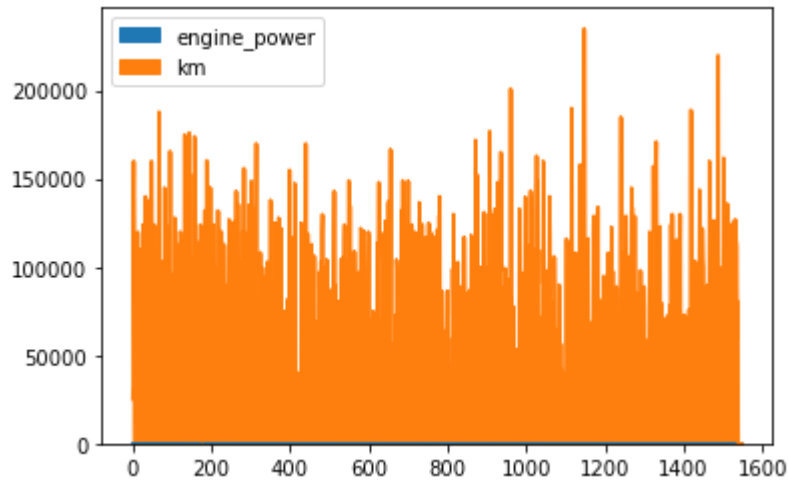
Visualization

In [20]:

```
df=data[['engine_power','km']]
df.plot.area()
```

Out[20]:

<AxesSubplot:>

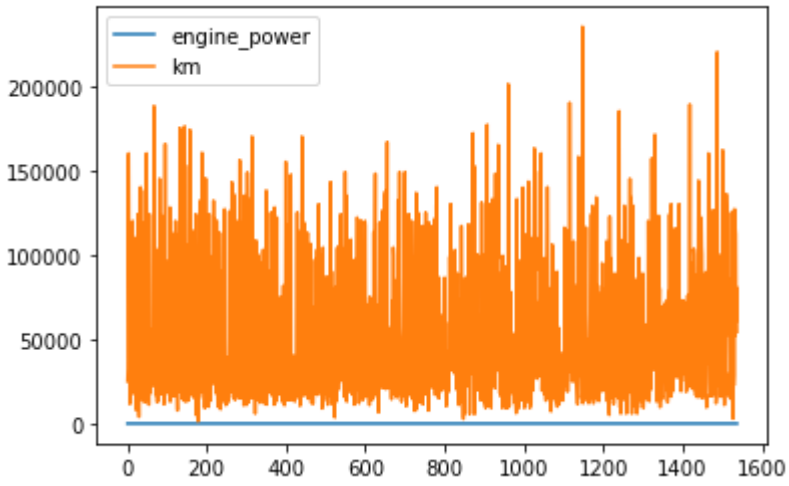


In [21]:

```
df.plot.line()
```

Out[21]:

```
<AxesSubplot:>
```

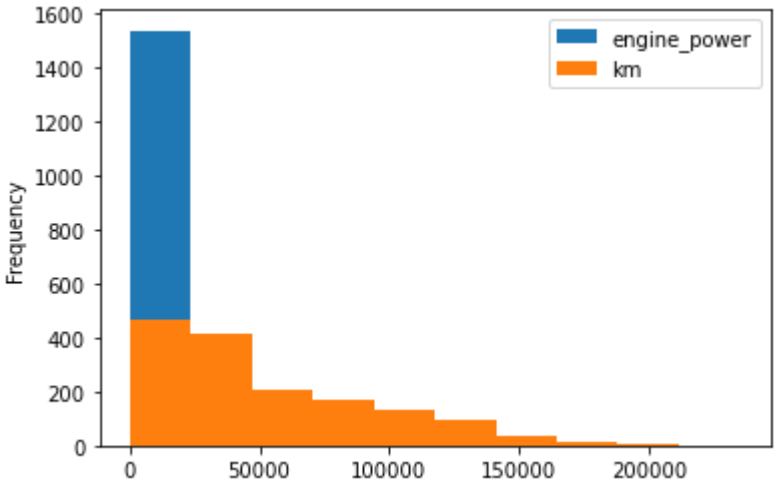


In [29]:

```
df.plot.hist()
```

Out[29]:

```
<AxesSubplot:ylabel='Frequency'>
```

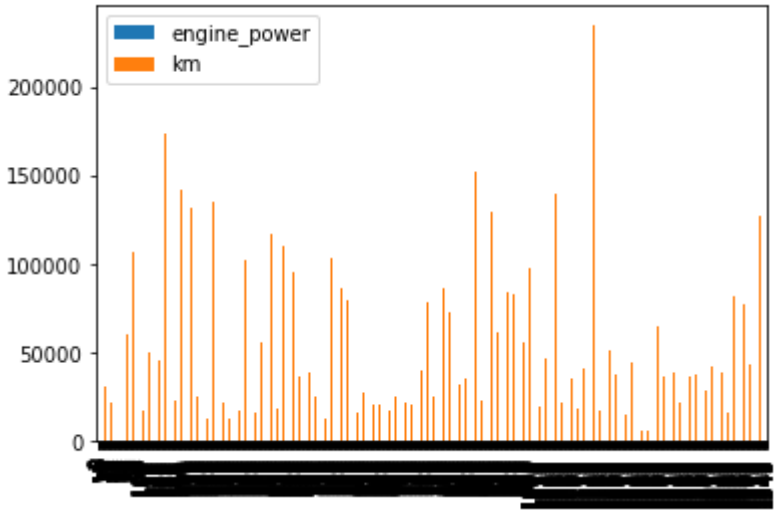


In [26]:

```
df.plot.bar()
```

Out[26]:

<AxesSubplot:>

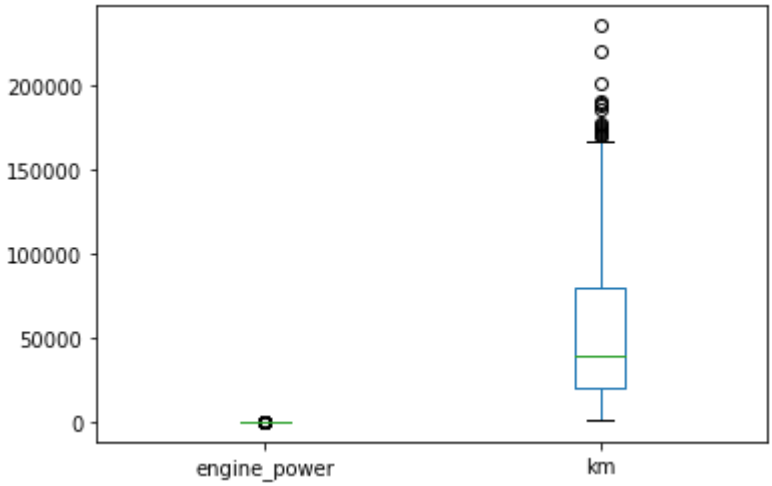


In [27]:

```
df.plot.box()
```

Out[27]:

<AxesSubplot:>



In [32]:

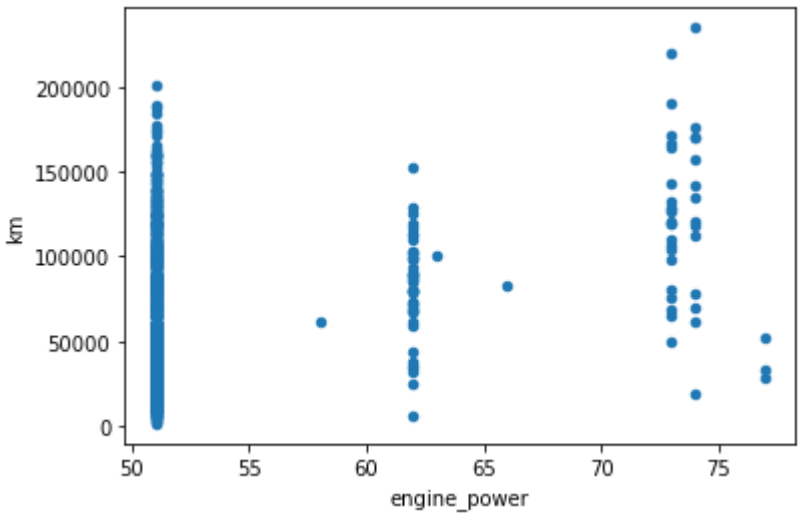
```
df.plot.pie(y="km")
```

In [33]:

```
df.plot.scatter(x="engine_power",y="km")
```

Out[33]:

```
<AxesSubplot:xlabel='engine_power', ylabel='km'>
```



Importing DataSet 2

In [11]:

```
data1=pd.read_csv(r"C:\Users\user\Downloads\2015.csv")
data1
```

Out[11]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563
...	...	...	...	...	...	...	...	...
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443

158 rows × 12 columns

Selecting first 4 rows using head()

In [12]:

```
data1.head(4)
```

Out[12]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Fr
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0

Selecting last 10 rows using tail()

In [13]:

```
data1.tail(10)
```

Out[13]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)
148	Chad	Sub-Saharan Africa	149	3.667	0.03830	0.34193	0.76062	0.15010
149	Guinea	Sub-Saharan Africa	150	3.656	0.03590	0.17417	0.46475	0.24009
150	Ivory Coast	Sub-Saharan Africa	151	3.655	0.05141	0.46534	0.77115	0.15185
151	Burkina Faso	Sub-Saharan Africa	152	3.587	0.04324	0.25812	0.85188	0.27125
152	Afghanistan	Southern Asia	153	3.575	0.03084	0.31982	0.30285	0.30335
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443

To get the Statistics data from the Table

In [14]:

```
data1.describe()
```

Out[14]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730

Find Row and Column of the Table using shape

In [15]:

```
data1.shape
```

Out[15]:

```
(158, 12)
```

Find size of the table

In [16]:

```
data1.size
```

Out[16]:

```
1896
```

Find Missing values using isna()

In [17]:

```
data1.isna()
```

Out[17]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Free
0	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	
153	False	False	False	False	False	False	False	False	F
154	False	False	False	False	False	False	False	False	F
155	False	False	False	False	False	False	False	False	F
156	False	False	False	False	False	False	False	False	F
157	False	False	False	False	False	False	False	False	F

158 rows × 12 columns

Drop the missing values using dropna()

In [18]:

```
data1.dropna(axis=1)#Drop Column
```

Out[18]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(Go C
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	
...	...	...	...	...	...	...	...	...	...	

In [19]:

```
data1.dropna()#Drop Row
```

Out[19]:

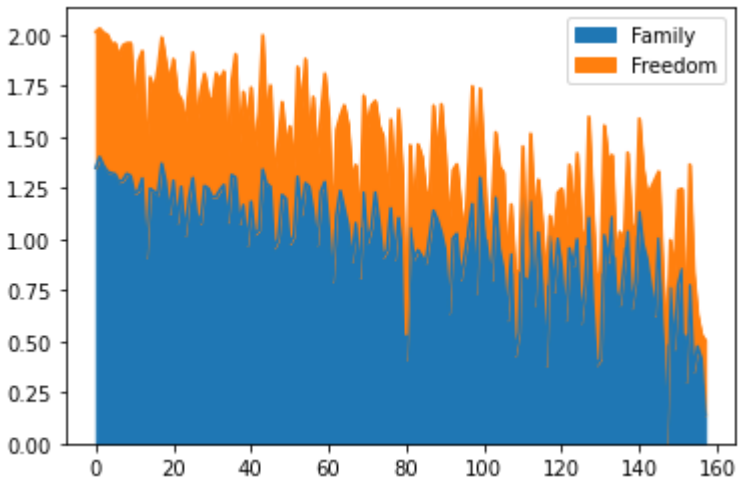
	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(Gove Corr
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	
5	Finland	Western Europe	6	7.406	0.03140	1.29025	1.31826	0.88911	0.64169	
...	...	...	...	...	...	...	...	...	...	

In [35]:

```
df1=data1[['Family','Freedom']]
df1.plot.area()
```

Out[35]:

```
<AxesSubplot:>
```

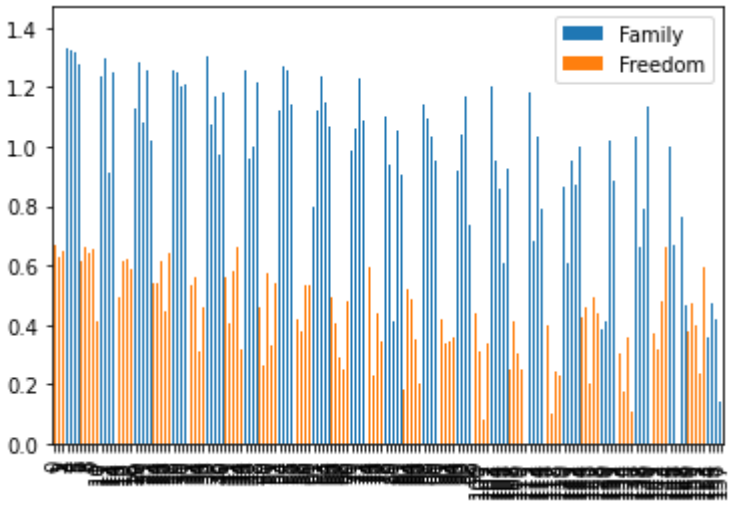


In [36]:

```
df1.plot.bar()
```

Out[36]:

<AxesSubplot:>

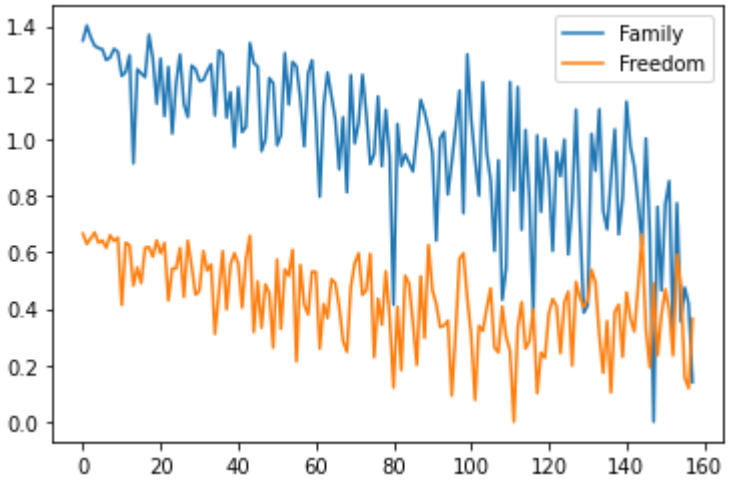


In [37]:

```
df1.plot.line()
```

Out[37]:

<AxesSubplot:>

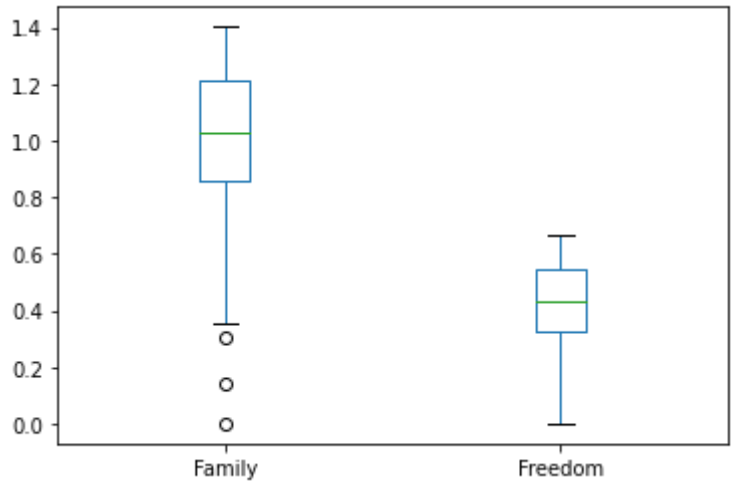


In [38]:

```
df1.plot.box()
```

Out[38]:

```
<AxesSubplot:>
```

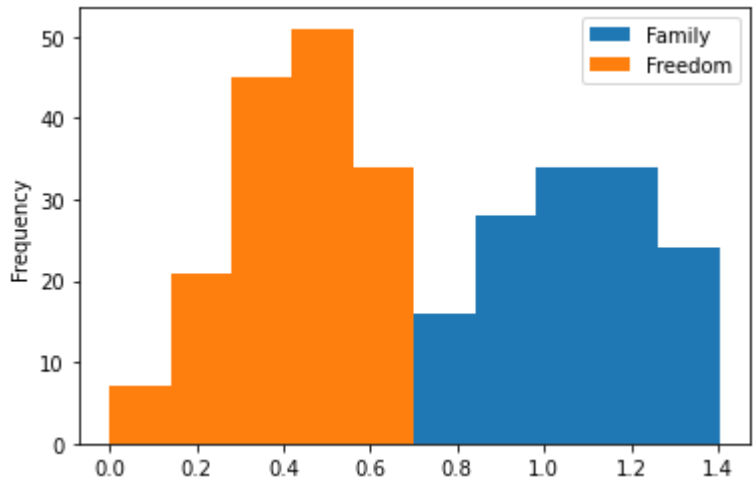


In [39]:

```
df1.plot.hist()
```

Out[39]:

```
<AxesSubplot:ylabel='Frequency'>
```

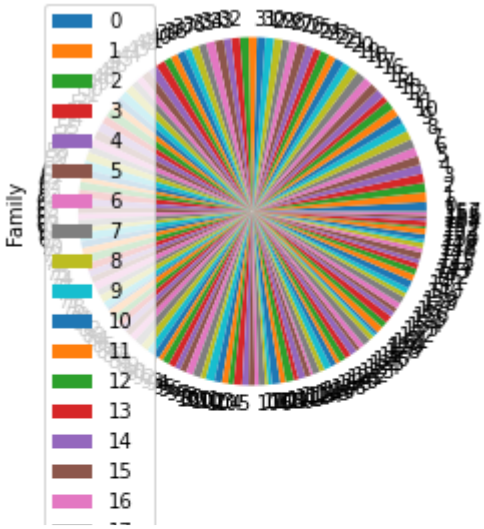


In [40]:

```
df1.plot.pie(y="Family")
```

Out[40]:

```
<AxesSubplot:ylabel='Family'>
```

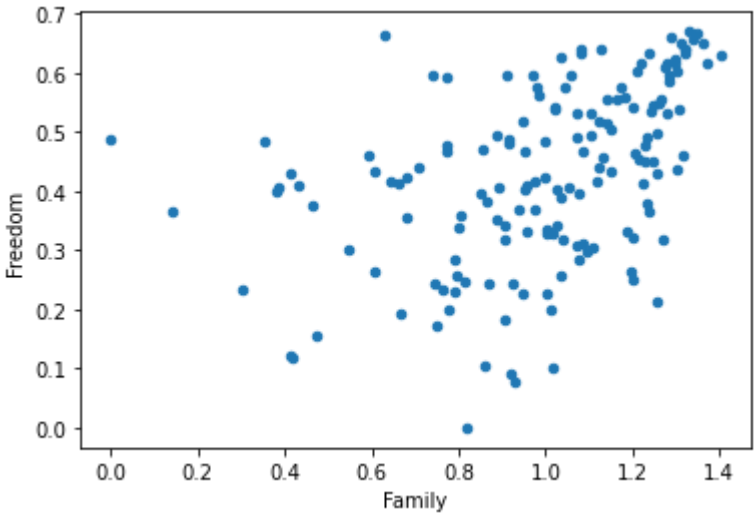


In [41]:

```
df1.plot.scatter(x="Family",y="Freedom")
```

Out[41]:

```
<AxesSubplot:xlabel='Family', ylabel='Freedom'>
```



In [ ]: