Basic operations using Numpy and Pandas

In [1]:
```
import numpy as np
import pandas as pd
```

Importing the dataset

In [2]:
```
data = pd.read_csv(r"C:\Users\user\Desktop\fiat500_VehicleSelection_Dataset (1
```

Selecting first 10 rows from the dataset

In [3]:
```
data.head(10)
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pric
0	1	lounge	51	882	25000	1	44.907242	8.611560	890
1	2	pop	51	1186	32500	1	45.666359	12.241890	880
2	3	sport	74	4658	142228	1	45.503300	11.417840	420
3	4	lounge	51	2739	160000	1	40.633171	17.634609	600
4	5	pop	73	3074	106880	1	41.903221	12.495650	570
5	6	pop	74	3623	70225	1	45.000702	7.682270	790
6	7	lounge	51	731	11600	1	44.907242	8.611560	1075
7	8	lounge	51	1521	49076	1	41.903221	12.495650	919
8	9	sport	73	4049	76000	1	45.548000	11.549470	560
9	10	sport	51	3653	89000	1	45.438301	10.991700	600

Selecting last 10 rows from the dataset

In [4]:
```
data.tail(10)
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
1528	1529	lounge	51	2861	126000	1	43.841980	10.51531
1529	1530	lounge	51	731	22551	1	38.122070	13.36112
1530	1531	lounge	51	670	29000	1	45.764648	8.99450
1531	1532	sport	73	4505	127000	1	45.528511	9.59323
1532	1533	pop	51	1917	52008	1	45.548000	11.54947
1533	1534	sport	51	3712	115280	1	45.069679	7.70492
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687
1535	1536	pop	51	2223	60457	1	45.481541	9.41348
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227
1537	1538	pop	51	1766	54276	1	40.323410	17.56827

To show the statistical data of the table

In [5]:
```
data.describe()
```

Out[5]:

	ID	engine_power	age_in_days	km	previous_owners	lat	
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	

To show the row and column

In [6]:
```
data.shape
```

Out[6]:
```
(1538, 9)
```

To show size of the table

In [7]:
```
data.size
```

Out[7]:
```
13842
```

To count the missing values

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
...	...	...	...	...	...	...	...	...	...
1533	False	False	False	False	False	False	False	False	False
1534	False	False	False	False	False	False	False	False	False
1535	False	False	False	False	False	False	False	False	False
1536	False	False	False	False	False	False	False	False	False
1537	False	False	False	False	False	False	False	False	False

1538 rows × 9 columns

Fill the empty values

In [9]:
```
data.fillna("5")
```

Out[9]:

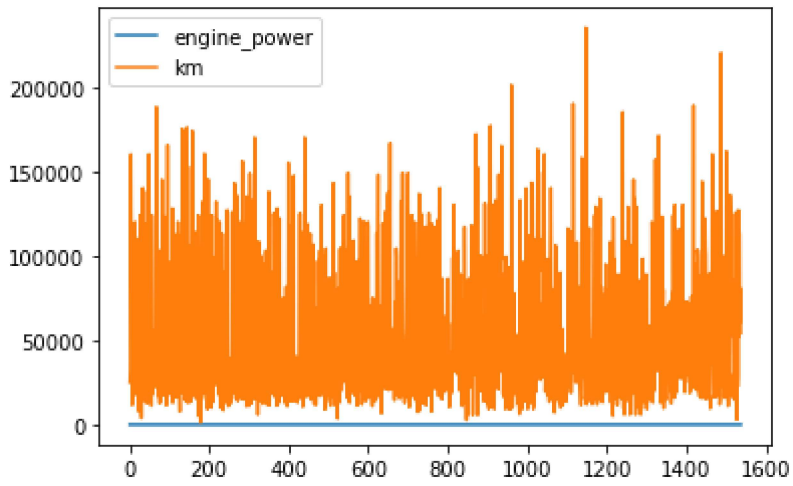
	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1	lounge	51	882	25000	1	44.907242	8.611560
1	2	pop	51	1186	32500	1	45.666359	12.241890
2	3	sport	74	4658	142228	1	45.503300	11.417840
3	4	lounge	51	2739	160000	1	40.633171	17.634609
4	5	pop	73	3074	106880	1	41.903221	12.495650
...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870
1535	1536	pop	51	2223	60457	1	45.481541	9.413480
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270
1537	1538	pop	51	1766	54276	1	40.323410	17.568270

1538 rows × 9 columns

Visualization

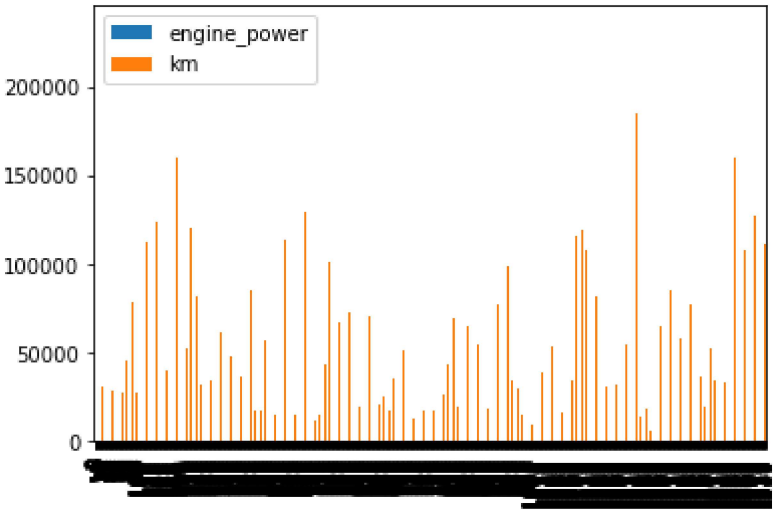
In [10]:
```
df=data[['engine_power','km']]
df.plot.line()
```

Out[10]: <AxesSubplot:>



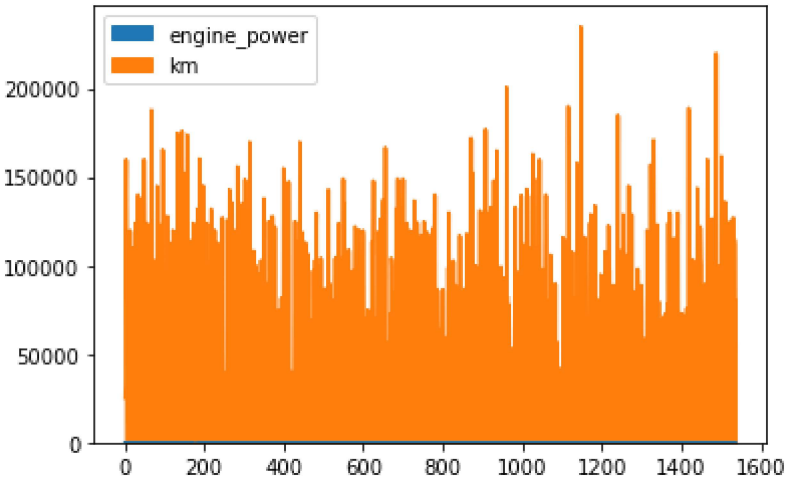
In [11]:
```
df.plot.bar()
```

Out[11]: <AxesSubplot:>



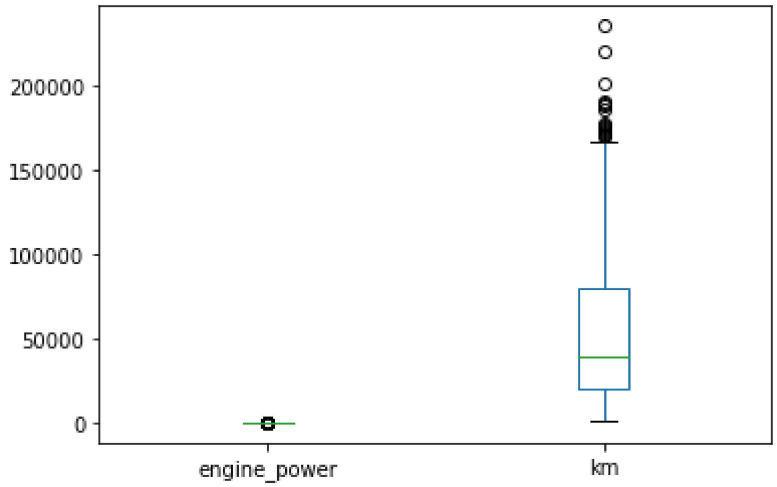
In [12]:
```
df.plot.area()
```

Out[12]: <AxesSubplot:>



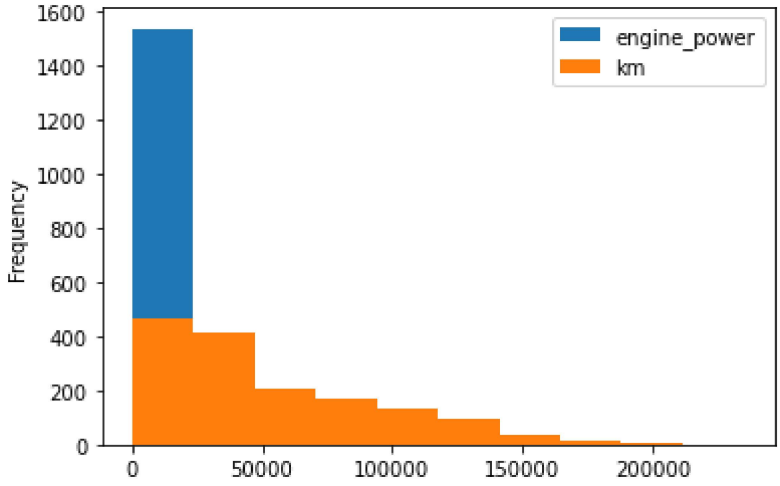
In [13]:
```
df.plot.box()
```

Out[13]: <AxesSubplot:>



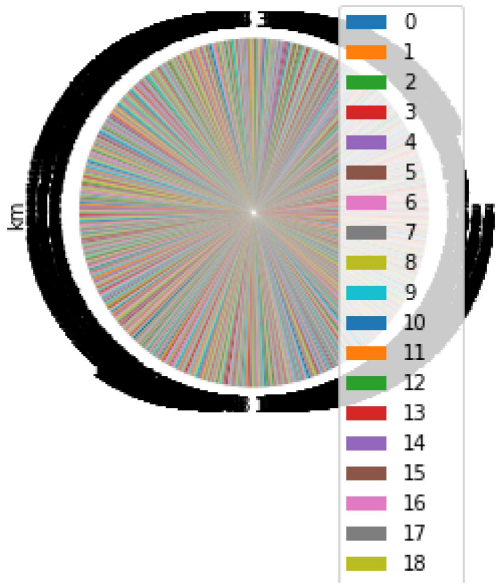
In [14]:
```
df.plot.hist()
```

Out[14]: <AxesSubplot:ylabel='Frequency'>



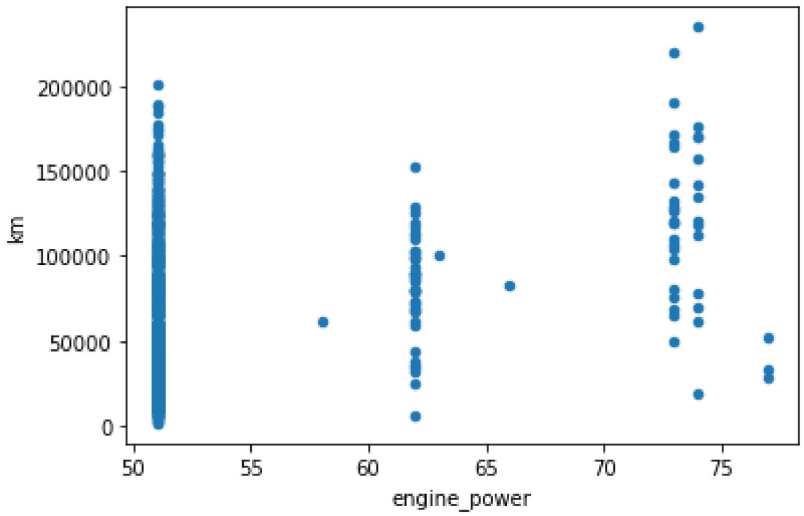
In [15]:
```
df.plot.pie(y='km')
```

Out[15]: <AxesSubplot:ylabel='km'>



In [16]:
```
df.plot.scatter(x='engine_power',y='km')
```

Out[16]: <AxesSubplot:xlabel='engine_power', ylabel='km'>



Data Set 2[2015]

Importing the dataset

In [17]:
```
data1 = pd.read_csv(r"C:\Users\user\Desktop\2015 dataset.csv")
```

Selecting first 10 rows from the dataset

In [18]:
```
data1.head(10)
```

Out[18]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freec
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63
5	Finland	Western Europe	6	7.406	0.03140	1.29025	1.31826	0.88911	0.64
6	Netherlands	Western Europe	7	7.378	0.02799	1.32944	1.28017	0.89284	0.61
7	Sweden	Western Europe	8	7.364	0.03157	1.33171	1.28907	0.91087	0.65
8	New Zealand	Australia and New Zealand	9	7.286	0.03371	1.25018	1.31967	0.90837	0.63
9	Australia	Australia and New Zealand	10	7.284	0.04083	1.33358	1.30923	0.93156	0.65

Selecting last 10 rows from the dataset

In [19]:
```
data1.tail(10)
```

Out[19]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Fre
148	Chad	Sub-Saharan Africa	149	3.667	0.03830	0.34193	0.76062	0.15010	0
149	Guinea	Sub-Saharan Africa	150	3.656	0.03590	0.17417	0.46475	0.24009	0
150	Ivory Coast	Sub-Saharan Africa	151	3.655	0.05141	0.46534	0.77115	0.15185	0
151	Burkina Faso	Sub-Saharan Africa	152	3.587	0.04324	0.25812	0.85188	0.27125	0
152	Afghanistan	Southern Asia	153	3.575	0.03084	0.31982	0.30285	0.30335	0
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0

To show the statistical data of the table

In [20]:
```
data1.describe()
```

Out[20]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(G C
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	1
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	

To show the row and column

In [21]:
```
data1.shape
```

Out[21]: (158, 12)

To show size of the table

In [22]:
```
data1.size
```

Out[22]: 1896

To count the missing values

In [23]:
```
data1.isna()
```

Out[23]:

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

158 rows × 12 columns

To remove rows that has empty values

In [24]:
```
data1.dropna()
```

Out[24]:

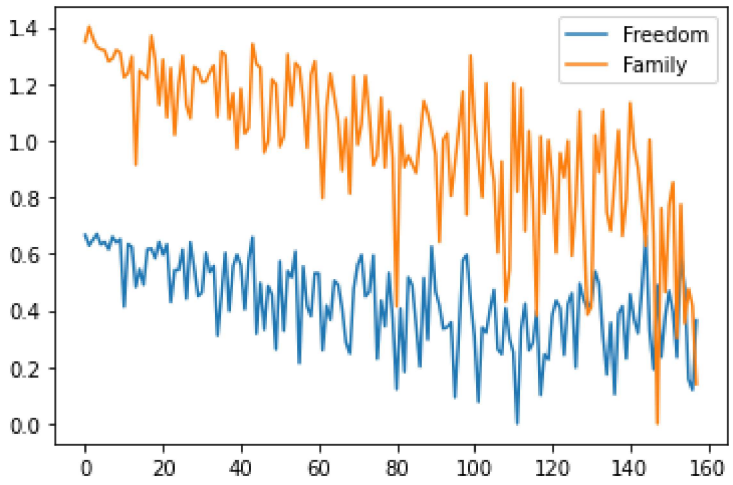
	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Fre
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.
2	Denmark	Western Europe	3	7.527	0.03328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

158 rows × 12 columns

Visualization

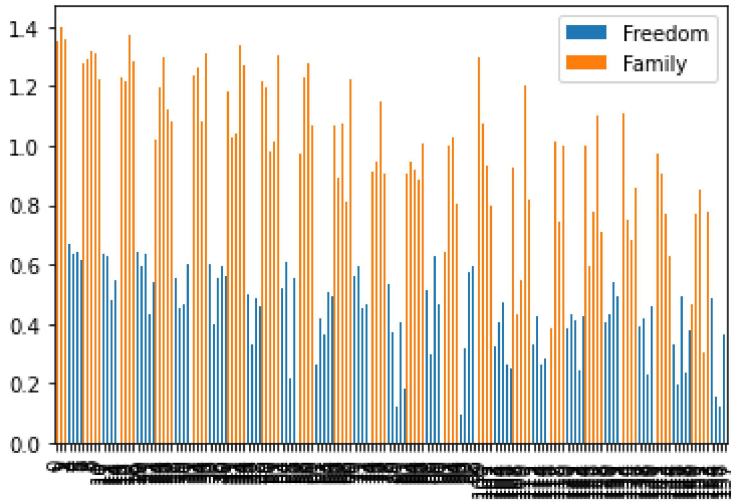
In [26]:
```
df=data1[['Freedom','Family']]
df.plot.line()
```

Out[26]: <AxesSubplot:>



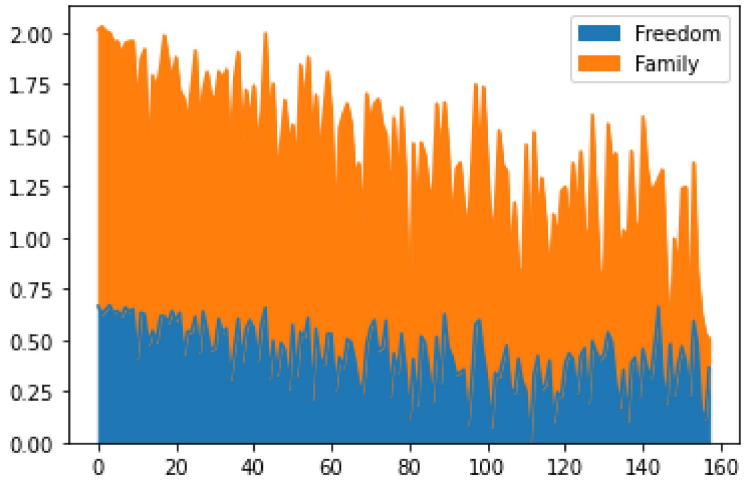
In [27]:
```
df.plot.bar()
```

Out[27]: <AxesSubplot:>



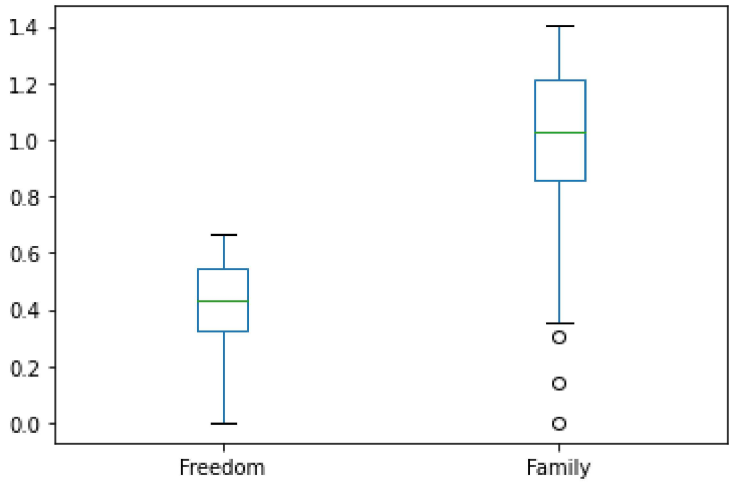
In [28]:
```
df.plot.area()
```

Out[28]: <AxesSubplot:>



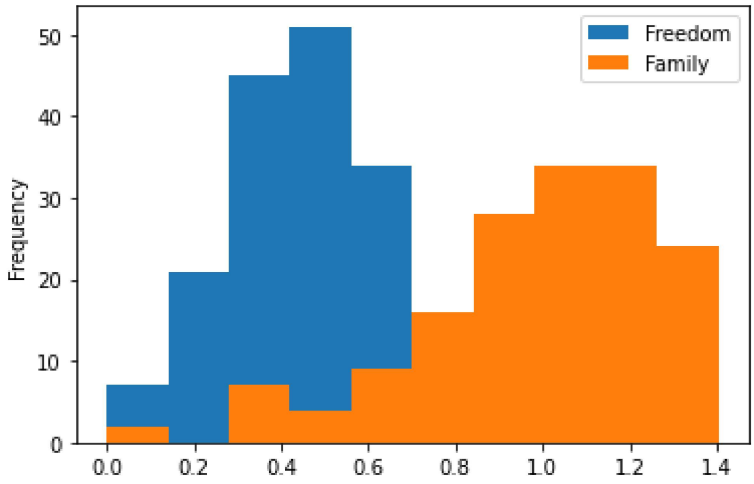
In [29]:
```
df.plot.box()
```

Out[29]: <AxesSubplot:>



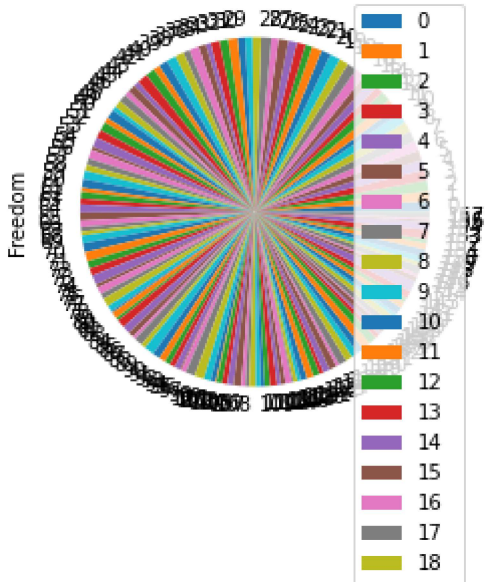
In [30]:
```
df.plot.hist()
```

Out[30]: <AxesSubplot:ylabel='Frequency'>



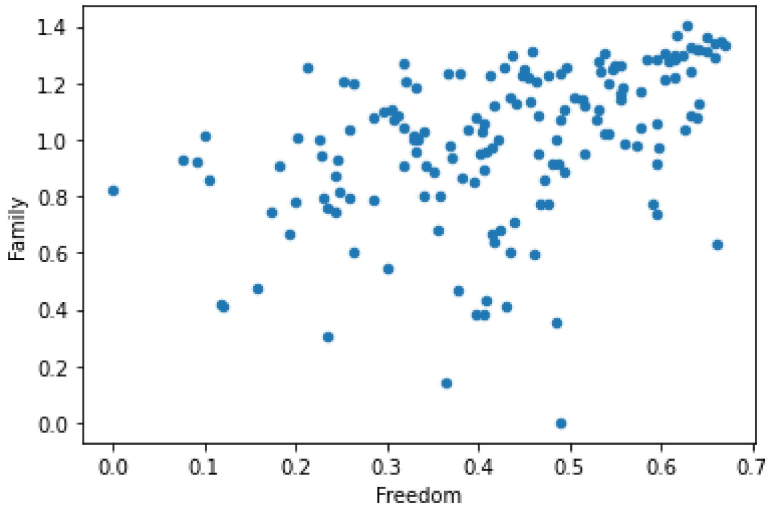
In [34]:
```
df.plot.pie(y='Freedom')
```

Out[34]: <AxesSubplot:ylabel='Freedom'>



In [35]:
```
df.plot.scatter(x='Freedom',y='Family')
```

Out[35]:
```
<AxesSubplot:xlabel='Freedom', ylabel='Family'>
```



In [ ]: