In [10]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib as mp
import scipy
from scipy import stats
from sklearn.preprocessing import OneHotEncoder
import matplotlib. pyplot as plt
```

In [4]:
```python
ds=pd.read_csv("ablone 3.csv")
```

In [11]:
```python
ds.head()
```

Out[11]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In [11]:
```python
ds.tail()
```

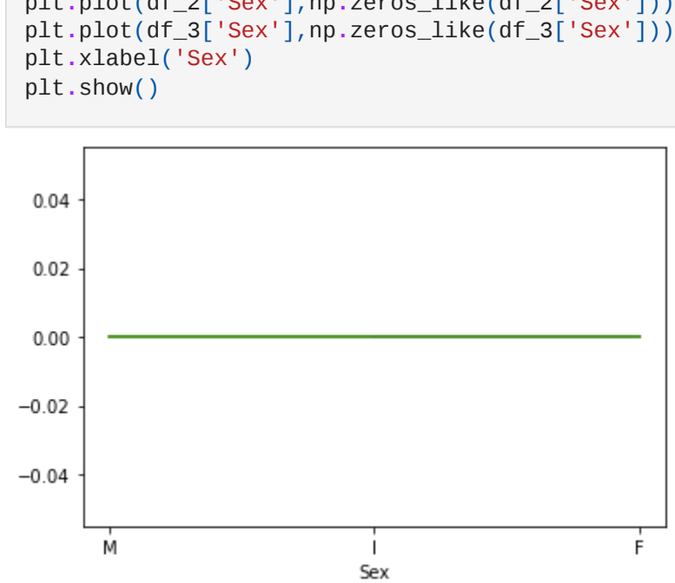
Out[11]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

Univariate Analysis

In [7]:
```python
df_1=ds.loc[ds['Rings']==7]
df_2=ds.loc[ds['Rings']==9]
df_3=ds.loc[ds['Rings']==10]
```

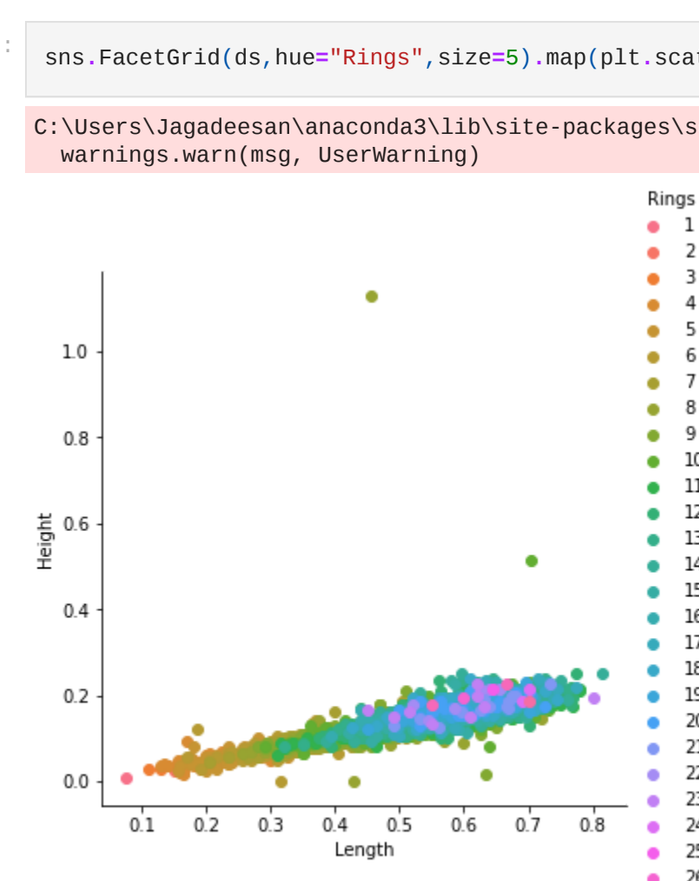
In [11]:
```python
plt.plot(df_1['Sex'],np.zeros_like(df_1['Sex']))
plt.plot(df_2['Sex'],np.zeros_like(df_2['Sex']))
plt.plot(df_3['Sex'],np.zeros_like(df_3['Sex']))
plt.xlabel('Sex')
plt.show()
```



Bi-Variate Analysis

In [13]:
```python
sns.FacetGrid(ds,hue="Rings",size=5).map(plt.scatter,"Length","Height").add_legend();
```

```
C:\Users\Jagadeesan\anaconda3\lib\site-packages\seaborn\axisgrid.py:337: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

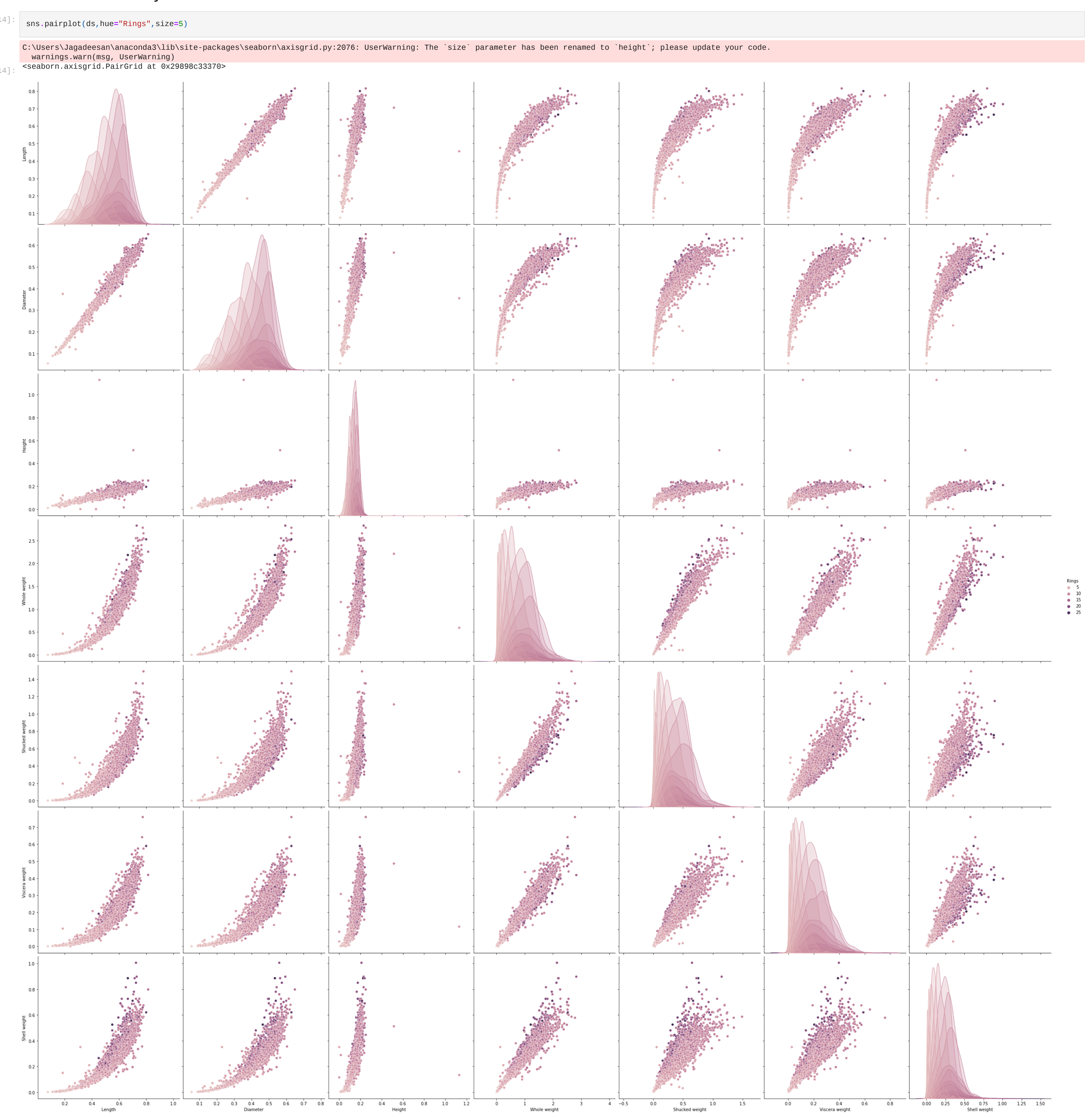


Multi-Variate Analysis

In [14]:
```python
sns.pairplot(ds,hue="Rings",size=5)
```

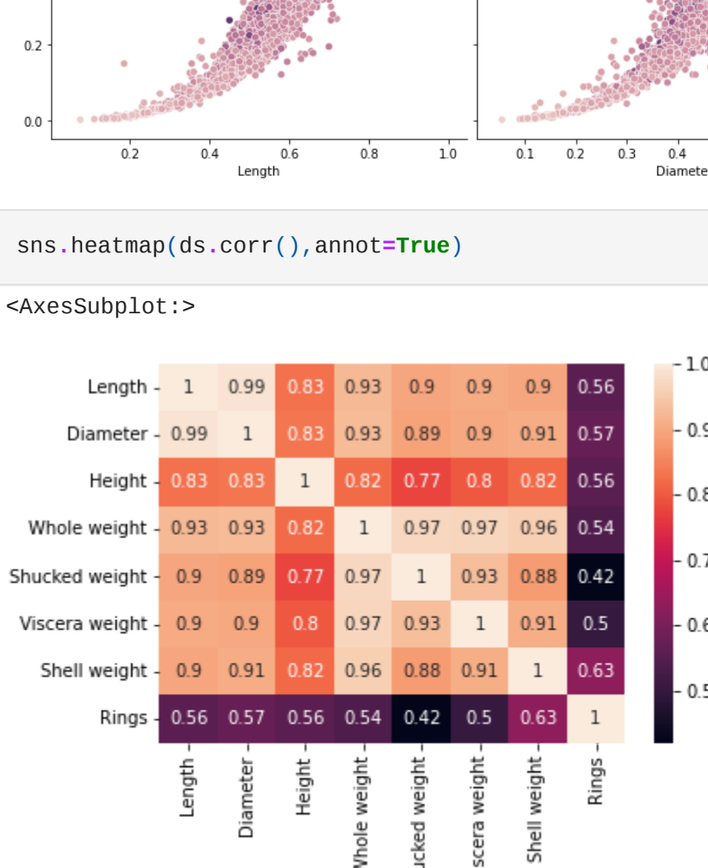
```
C:\Users\Jagadeesan\anaconda3\lib\site-packages\seaborn\axisgrid.py:2076: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

Out[14]: <seaborn.axisgrid.PairGrid at 0x29898c33370>



In [25]:
```python
sns.heatmap(ds.corr(),annot=True)
```

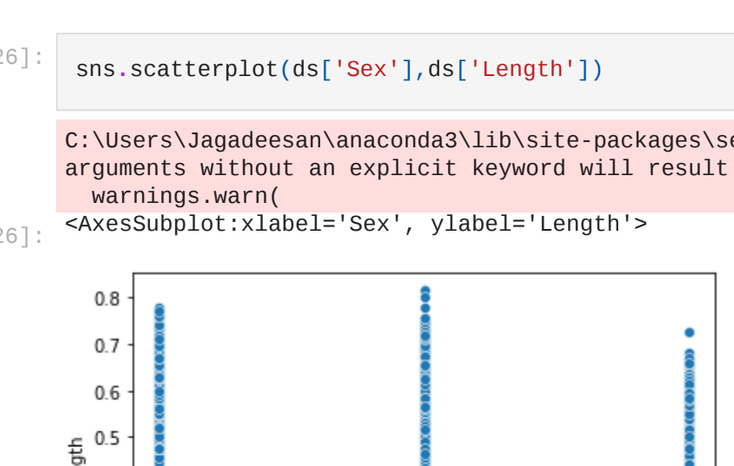
Out[25]: <AxesSubplot:>



In [26]:
```python
sns.scatterplot(ds['Sex'],ds['Length'])
```

```
C:\Users\Jagadeesan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

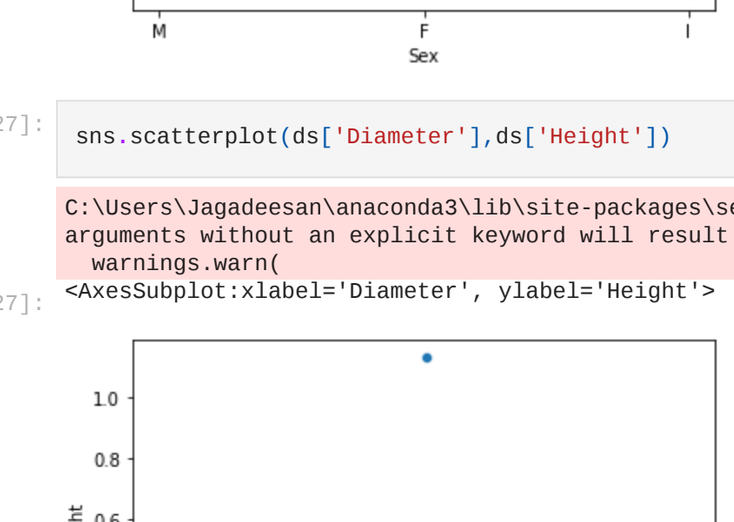
Out[26]: <AxesSubplot:xlabel='Sex', ylabel='Length'>



In [27]:
```python
sns.scatterplot(ds['Diameter'],ds['Height'])
```

```
C:\Users\Jagadeesan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[27]: <AxesSubplot:xlabel='Diameter', ylabel='Height'>



In [ ]:
```python
# Descriptive statistics
```

In [16]:
```python
ds.describe()
```

Out[16]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

In [17]:
```python
ds.dtypes
```

Out[17]:
```
Sex                object
Length            float64
Diameter          float64
Height            float64
Whole weight      float64
Shucked weight    float64
Viscera weight    float64
Shell weight      float64
Rings               int64
dtype: object
```

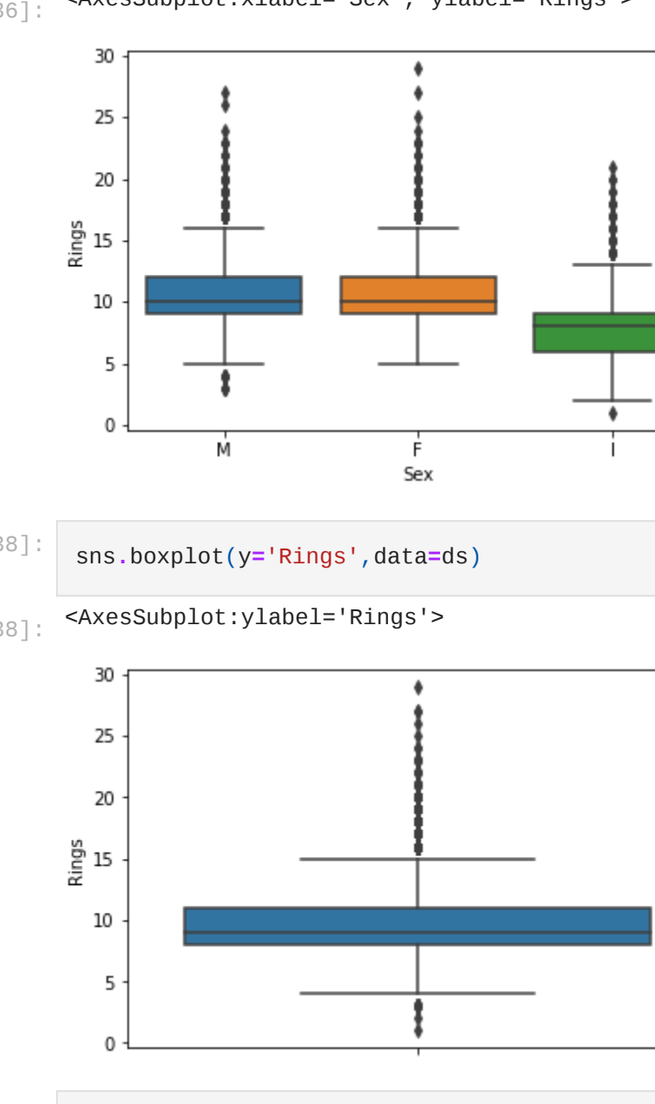
In [20]:
```python
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

In [ ]:
```python
#outliers
```

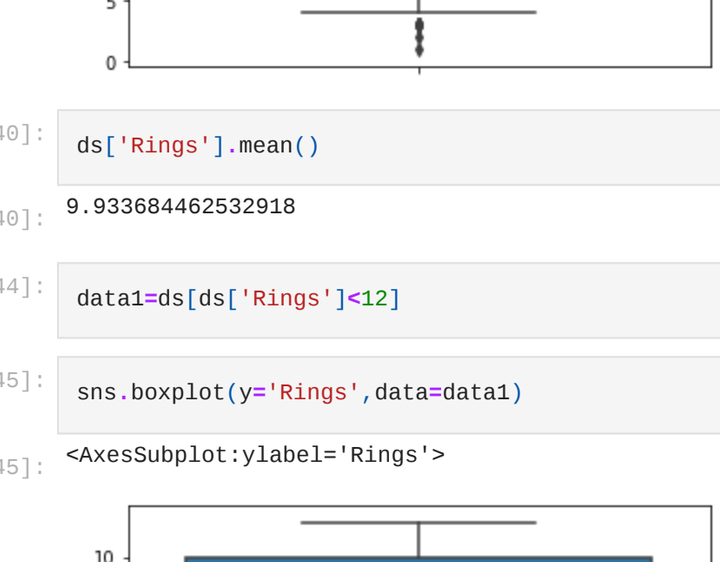
In [34]:
```python
sns.displot(ds['Sex'])
```

Out[34]: <seaborn.axisgrid.FacetGrid at 0x2989e885700>



In [36]:
```python
sns.boxplot(x='Sex',y='Rings',data=ds)
```

Out[36]: <AxesSubplot:xlabel='Sex', ylabel='Rings'>



In [38]:
```python
sns.boxplot(y='Rings',data=ds)
```

Out[38]: <AxesSubplot:ylabel='Rings'>

In [40]:
```python
ds['Rings'].mean()
```

Out[40]: 9.933684462532918

In [44]:
```python
data1=ds[ds['Rings']<12]
```

In [45]:
```python
sns.boxplot(y='Rings',data=data1)
```

Out[45]: <AxesSubplot:ylabel='Rings'>

missing values

In [48]:
```python
ds.duplicated()
```

Out[48]:
```
0       False
1       False
2       False
3       False
4       False
        ...
4172    False
4173    False
4174    False
4175    False
4176    False
Length: 4177, dtype: bool
```

In [49]:
```python
ds.isna().any().sum()
```

Out[49]: 0

In [50]:
```python
ds['Sex'].unique()
```

Out[50]: array(['M', 'F', 'I'], dtype=object)

In [51]:
```python
from sklearn.preprocessing import LabelEncoder
```

In [52]:
```python
le=LabelEncoder
```

In [53]:
```python
ds["Sex"]=ds["Sex"].replace(["M","F","I"],[0,1,2])
```

In [54]:
```python
ds.head()
```

Out[54]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	1	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In [55]:
```python
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   int64
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(2)
memory usage: 293.8 KB
```

In [56]:
```python
ds.describe()
```

Out[56]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.955470	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.827815	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.000000	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.000000	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	1.000000	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	2.000000	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	2.000000	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

In [57]:
```python
ds.duplicated().sum()
```

Out[57]: 0

In [58]:
```python
ds.corr()
```

Out[58]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
Sex	1.000000	-0.448765	-0.458245	-0.417928	-0.461238	-0.440927	-0.454658	-0.445549	-0.351822
Length	-0.448765	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
Diameter	-0.458245	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
Height	-0.417928	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
Whole weight	-0.461238	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
Shucked weight	-0.440927	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
Viscera weight	-0.454658	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
Shell weight	-0.445549	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
Rings	-0.351822	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

split of dependent and independent value

In [59]:
```python
x=ds.drop('Rings',axis=1)
y=ds['Rings']
```

In [62]:
```python
x
```

Out[62]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500
1	0	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700
2	1	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100
3	0	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550
...	...	...	...	...	...	...	...	...
4172	1	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490
4173	0	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605
4174	0	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080
4175	1	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960
4176	0	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950

4177 rows × 8 columns

In [63]:
```python
y
```

Out[63]:
```
0       15
1        7
2        9
3       10
4        7
        ..
4172    11
4173    10
4174     9
4175    10
4176    12
Name: Rings, Length: 4177, dtype: int64
```

In [64]:
```python
data_tips=pd.get_dummies(ds)
data_tips
```

Out[64]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	0	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	1	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	0	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	1	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	0	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	0	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	1	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	0	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

4177 rows × 9 columns

catgorical encoding

In [66]:
```python
one_encode=OneHotEncoder(sparse=False)
encoded_arr=one_encode.fit_transform(ds[['Length','Height','Sex','Diameter']])
encoded_arr
```

Out[66]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

split into train and test

In [68]:
```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Builiding an model

In [69]:
```python
from sklearn.naive_bayes import GaussianNB
```

In [70]:
```python
nb=GaussianNB()
```

In [71]:
```python
nb.fit(x_train,y_train)
```

Out[71]: GaussianNB()

Train the Model Test the Model

In [73]:
```python
p=nb.predict(x_test)
p
```

Out[73]:
```
array([10,  8,  8,  8, 11, 11,  6, 10,  4, 11,  7,  7,  8,  9, 11,  9, 11,
        7,  9, 11, 11, 11, 11,  8, 11,  9, 11,  7,  9,  8, 11,  9, 10,  8,
        8,  3,  9,  5,  5,  9,  6, 10, 10, 10,  5,  9,  6, 11,  9,  8,  4,
        5,  6,  6,  9,  9,  8,  8, 11, 11,  6, 10,  8,  5, 27,  7,  7,  9,
       11, 11, 11,  6, 11,  8,  8,  6,  5, 11, 11,  9, 10,  9,  5,  9,  7,
        6, 10,  7,  4,  9, 11,  6,  7,  9, 11,  6,  8,  9, 11, 11, 27, 11,
        9, 10,  6,  9, 11,  5,  8,  7,  5,  8, 11, 11, 11,  8,  8,  7,  9,
        8,  5, 11,  7, 10,  6,  7,  7,  9,  5, 10,  7,  8, 10, 10,  9, 11,
        8,  8,  7,  6,  8,  6, 11, 11, 11, 11,  5,  7, 11, 11, 11, 11,  7,
        8, 11,  8,  9,  8,  3,  9,  9,  9,  9, 11, 11, 11,  8,  7,  9,  9,
        8,  9,  5,  8,  7,  7, 11,  9,  7, 11,  8,  5, 11,  7,  8, 11,  7,
        9,  6,  8,  9,  9, 10, 11,  9,  5,  5,  7,  9,  7, 10,  7,  6,  8,
       11, 27,  7,  7,  6, 11,  9, 11,  9,  3,  9,  9,  8, 11,  8, 11,  7,
        6,  9,  9,  8,  9, 11,  6,  6,  8,  5,  5,  7,  7,  7,  5,  9,  7,
        9,  9,  7, 10, 11,  6, 10,  7, 10,  7,  8, 20,  7, 10, 10,  9,  8,
        8,  6,  9,  6, 10,  9,  9,  4, 11,  7,  9,  4, 10,  5, 11, 11, 11,
        9,  8,  5,  8,  5,  7,  9,  3, 11, 11, 11,  8,  7,  7,  9,  7,  7,
        8,  7,  7, 16,  8,  7,  8, 11, 11,  9,  6,  9,  9, 11,  7,  9,  6,
       11,  6, 11,  5,  9,  8,  5,  9,  5, 11,  8,  9, 11, 10, 11,  9, 10,
       11, 10,  9,  6,  8, 11,  6, 11,  8,  9,  6,  9,  4,  6,  8, 11, 11,
       11,  9, 11, 11, 10,  9,  7,  5,  9,  6,  7,  9,  5,  9,  8,  9, 11,
       11,  9, 11,  7,  6, 11,  9,  9, 11,  9, 11, 10, 11,  4,  9,  8, 11,
        7, 11,  8,  9,  8,  7,  9,  6,  9, 11, 11, 11, 11,  9,  4, 11,  8,
       11, 11, 10,  8,  7,  9,  7,  8, 10,  9, 11,  9, 11,  8,  5, 11,  8,
        6,  8,  8, 10,  9,  5,  6,  9,  6, 11,  4,  8,  7, 11,  8,  6,  6,
        9, 11, 11, 10,  5, 11,  5, 11,  6,  7,  7,  9,  4,  9, 11,  8,  8,
        6,  4,  8, 11,  7,  8, 11, 11, 11,  7, 27,  5,  3,  9,  5,  7, 11,
        9, 11, 10,  7,  8,  7,  6,  6,  8,  9,  5,  5, 11,  9,  5, 11,  5,
        6, 17, 11,  6,  6,  9,  8, 11,  8,  8,  8,  8, 27,  7, 11,  9,  6,
        9,  9,  4,  7,  9,  9, 10, 11, 10,  7, 10,  7,  9,  7,  9, 11,  8,
       10,  6,  5,  8,  9,  7,  9,  7,  8,  8, 11,  9,  9,  6, 10, 11, 10,
       10, 11, 10, 11,  8,  9, 11,  9,  8,  8,  6,  3, 10,  9, 11, 16, 11,
        4,  9,  3,  8,  9,  7,  6, 10,  6,  9,  9,  9,  8,  9,  7,  7,  7,
        8, 10,  9, 10,  9, 11,  9,  7,  3,  4,  8,  9, 11,  8, 11,  6,  9,
       11, 11,  7, 10,  9,  6,  9,  8,  7, 10,  7,  9,  9,  6,  6, 27,  9,
        9,  7,  8,  9,  9, 10,  9,  6,  9,  9,  6,  9, 11, 11, 11,  7,  9,
       11, 10,  9,  9,  9, 10, 11, 10,  9, 11,  6,  5,  9,  8,  5, 11,  7,
        9,  8,  8,  7,  7,  9,  8,  5,  5, 11,  8,  7, 11,  6, 11,  7, 10,
       11, 11,  9, 11,  9,  9, 20,  7,  9, 11, 11, 11,  4,  9,  8,  6,  9,
        8,  8,  8,  8, 10, 11,  9,  8, 11, 11, 11,  7,  7,  9,  8, 10, 11,
       10, 10, 11, 10, 11, 11, 11,  8,  5, 11,  6,  8,  8,  8,  7, 11,  9,
        4,  9,  7,  7,  9,  9,  9,  7, 11, 27,  9,  7, 10, 11,  8,  9, 10,
        8, 20,  6,  8,  8,  7, 10,  9, 11,  7,  8,  5,  5, 11, 11, 11,  7,
        7,  8,  9,  6,  3,  6, 11, 10,  9,  9,  8,  7,  9, 11,  9,  9,  7,
       11, 11,  3, 11,  6,  6,  6, 11, 11,  6,  9, 17,  6, 11,  9,  6,  6,
       10, 10,  8,  9,  9,  7,  9,  7,  7,  5,  9,  7, 10, 10,  6, 11,  8,
        9,  7,  7,  9,  9,  6,  9, 11, 11,  7, 10, 11,  8,  9,  6, 10, 11,
       10,  9, 11,  9,  8,  6,  9, 11,  5,  9,  9,  7,  9, 11,  8,  7,  9,
        5,  9, 11,  7,  8, 11,  7,  7, 11,  9,  9, 11, 11,  7,  9,  9,  6,
       11, 11,  7], dtype=int64)
```

Metrics

In [74]:
```python
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [88]:
```python
acc=accuracy_score(y_test,p)
acc
```

Out[88]: 0.23923444976076555

In [76]:
```python
from sklearn.neighbors import KNeighborsClassifier
```

In [77]:
```python
knn=KNeighborsClassifier()
```

In [78]:
```python
knn.fit(x_train,y_train)
```

Out[78]: KNeighborsClassifier()

In [79]:
```python
pn=knn.predict(x_test)
```

In [80]:
```python
accuracy_score(y_test,pn)
```

Out[80]: 0.22966507177033493

In [81]:
```python
confusion_matrix(y_test,pn)
```

Out[81]:
```
array([[ 0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  4,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  5,  4,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  3,  8,  8,  4,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  1,  5, 22, 15,  3,  1,  0,  1,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  1, 27, 29, 21, 10,  2,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  3,  8, 26, 31, 33,  9,  3,  2,  0,  1,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  2, 16, 31, 38, 26, 14,  1,  1,  1,  1,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  2,  6, 17, 35, 39, 18,  2,  5,  1,  1,  1,  1,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  4, 11, 24, 20, 16,  4,  4,  0,  1,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  2,  2,  7, 16, 10,  4,  3,  0,  0,  0,  2,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  1,  5,  8,  6, 12,  6,  0,  2,  0,  1,  1,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  1,  1,  7, 11,  5,  3,  2,  0,  0,  0,  0,
         0,  1,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  1,  1,  7,  4,  3,  3,  1,  2,  0,  0,  0,
         1,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  1,  1,  1,  4,  0,  2,  1,  1,  0,  0,  1,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  1,  3,  1,  0,  2,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  1,  2,  7,  1,  2,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  3,  2,  0,  1,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  2,  1,  1,  0,  1,  0,  0,  0,  2,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  1,  0,  1,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  1,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  1,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0]], dtype=int64)
```