In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:

```
import tkinter as TK
from tkinter.filedialog import askopenfilename
```

In [7]:

```
mpg = pd.read_csv("C:/Users/HP/OneDrive/Desktop/DataSciece/auto-mpg.csv")
```

In [8]:

```
mpg.head()
```

Out[8]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

In [13]:

```
#part(a)
#Identify the Dimentions of data
mpg.shape
```

Out[13]:

```
(398, 9)
```

In [14]:

```
#part(a)
#Identify the structure of data
mpg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [15]:

```
#part(a)
#Identify the summary of data
mpg.describe()
```

Out[15]:

	mpg	cylinders	displacement	weight	acceleration	model year	ori
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.572
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.802
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000
25%	17.500000	4.000000	104.250000	2223.750000	13.825000	73.000000	1.000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000
75%	29.000000	8.000000	262.000000	3608.000000	17.175000	79.000000	2.000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000

In [16]:

```
#part(b)
mpg.isnull().sum()
```

Out[16]:

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

In [17]:

```
mpg.dropna()
```

Out[17]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	ca nam
0	18.0	8	307.0	130	3504	12.0	70	1	chevrole chevell malib
1	15.0	8	350.0	165	3693	11.5	70	1	buic skylar 32
2	18.0	8	318.0	150	3436	11.0	70	1	plymoutl satellit
3	16.0	8	304.0	150	3433	12.0	70	1	am rebel ss
4	17.0	8	302.0	140	3449	10.5	70	1	for torin
...	...	...	...	...	...	...	...	...	.
393	27.0	4	140.0	86	2790	15.6	82	1	for mustan g
394	44.0	4	97.0	52	2130	24.6	82	2	vw picku
395	32.0	4	135.0	84	2295	11.6	82	1	dodg rampag
396	28.0	4	120.0	79	2625	18.6	82	1	for range
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s 1

398 rows × 9 columns

In [31]:

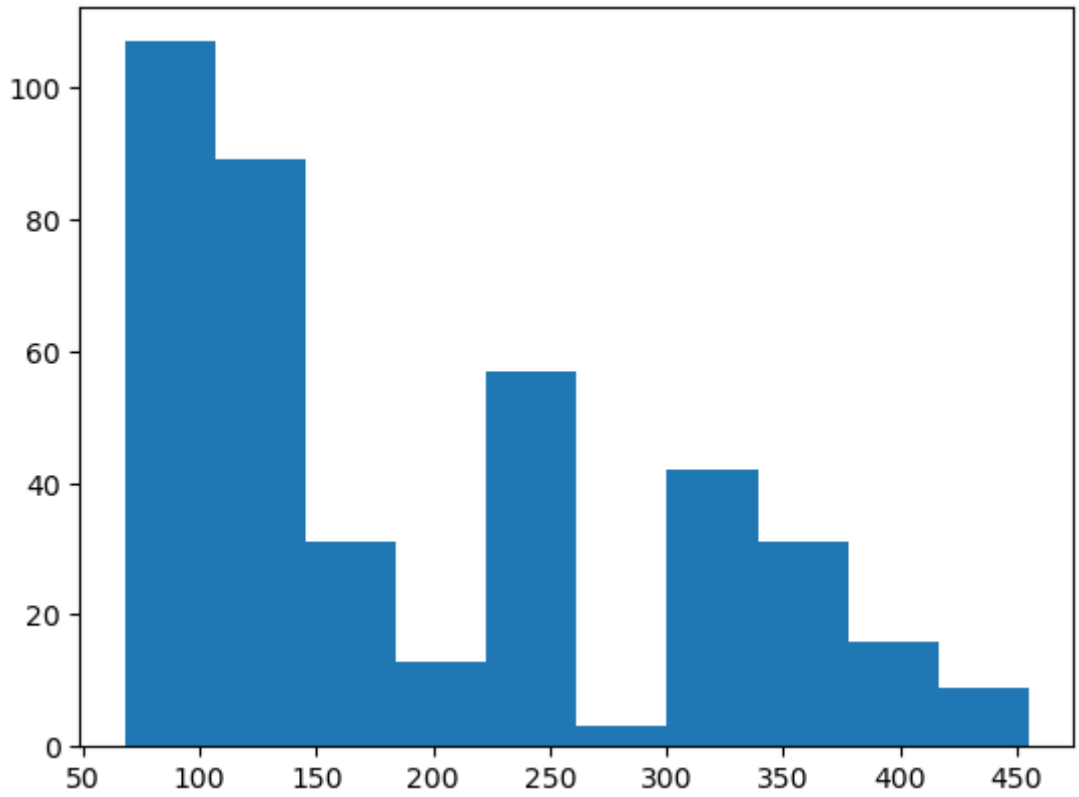
```
#part(c)
#to convert certain numerical into categorical variable for visualisation of data
#Some advantages of using categorical data include:

#Memory efficiency: Categorical data can be more memory-efficient than storing the same
#This is because categorical data is internally represented as integers, and the mapping
#is stored separately.

mpg['acceleration']=pd.Categorical(mpg['acceleration'])
```

In [30]:

```
#plot histogram for continous variable
plt.hist(mpg['displacement'])
```

Out[30]:

```
(array([107.,  89.,  31.,  13.,  57.,   3.,  42.,  31.,  16.,   9.]),
 array([ 68. , 106.7, 145.4, 184.1, 222.8, 261.5, 300.2, 338.9, 377.6,
        416.3, 455. ]),
 <BarContainer object of 10 artists>)
```

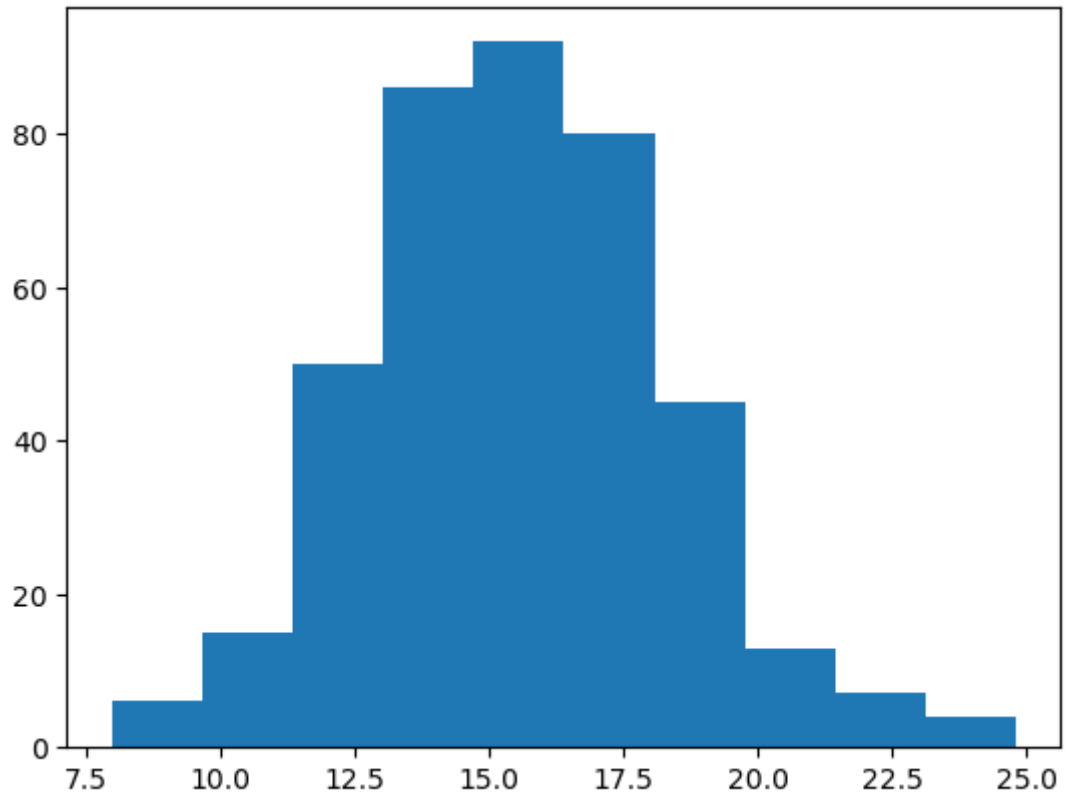


In [21]:

```
plt.hist(mpg['acceleration'])
```

Out[21]:

```
(array([ 6., 15., 50., 86., 92., 80., 45., 13.,  7.,  4.]),
 array([ 8.  ,  9.68, 11.36, 13.04, 14.72, 16.4 , 18.08, 19.76, 21.44,
        23.12, 24.8 ]),
 <BarContainer object of 10 artists>)
```

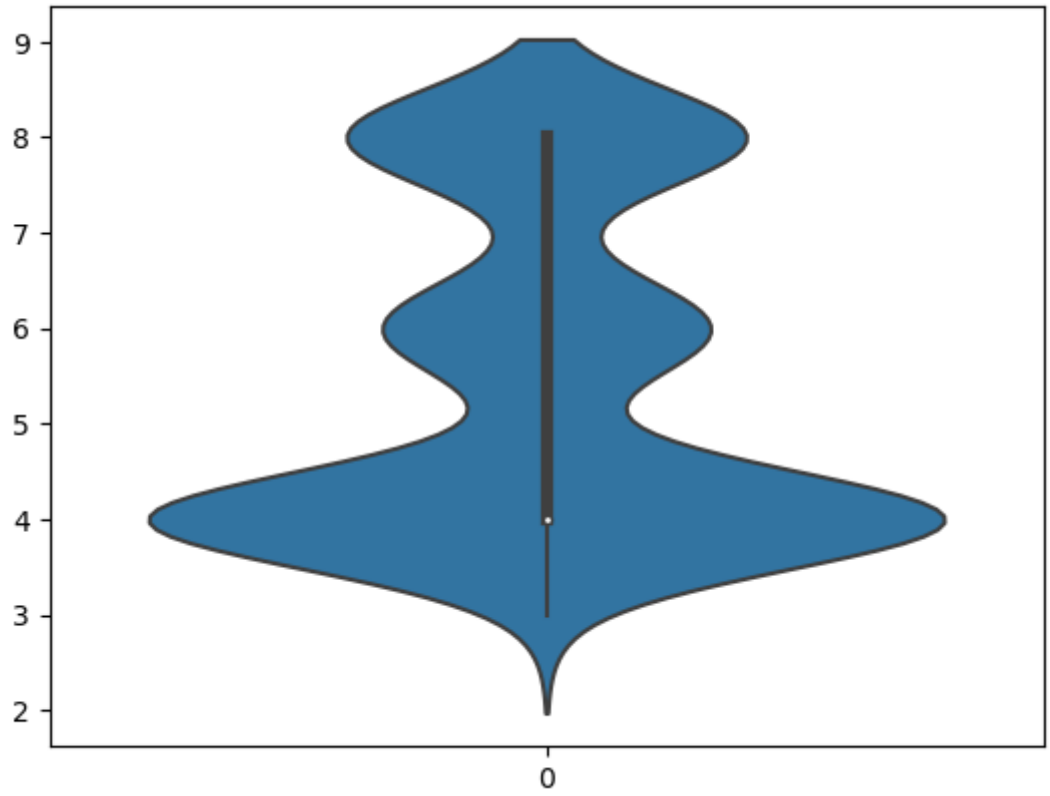


In [24]:

```
#part(d)     plot violin plot for numerical variable
sns.violinplot(mpg['cylinders'])
```

Out[24]:

<Axes: >

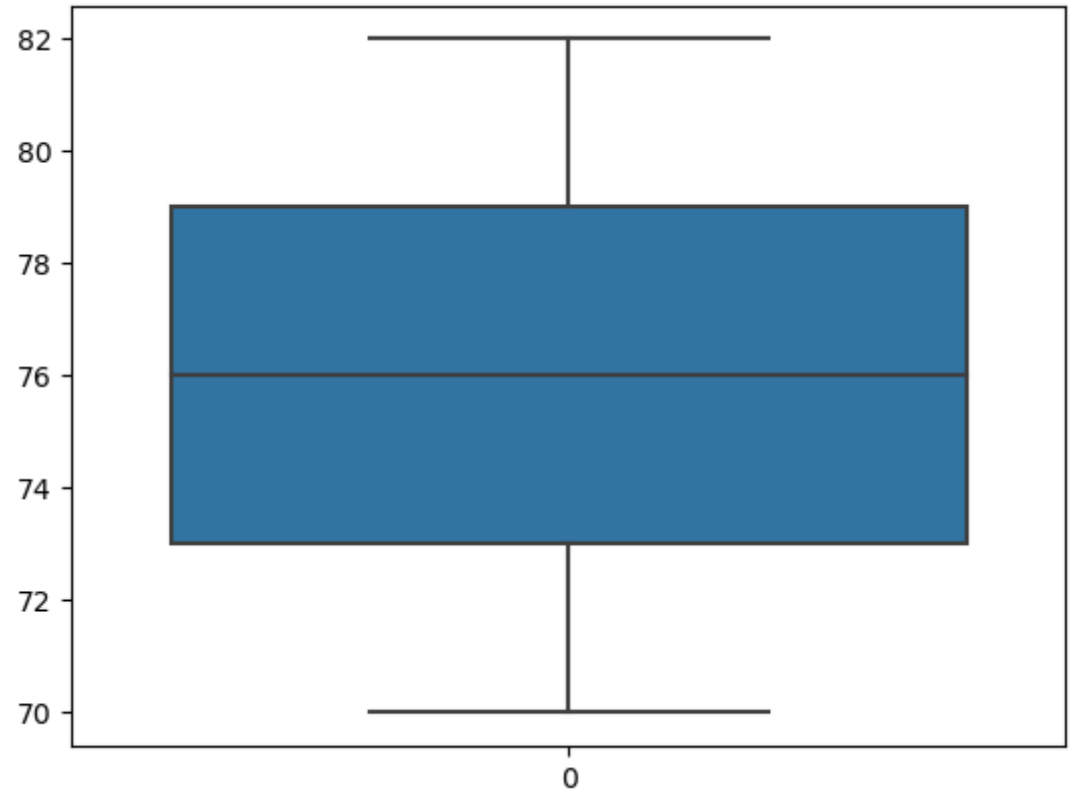


In [50]:

```
#part(e)
sns.boxplot(mpg['model year'])
```

Out[50]:

```
<Axes: >
```



In [ ]:

```
#part(f)
```

In [ ]:

```

```

In [51]:

```
#part(g)
#standardize the dataset for numerical attributes
nums = list(mpg.select_dtypes(exclude=['object']).columns)
nums   # to show output
```

Out[51]:

```
['mpg',
 'cylinders',
 'displacement',
 'weight',
 'acceleration',
 'model year',
 'origin']
```

In [53]:

```
from sklearn import preprocessing
min_max_scaler=preprocessing.MinMaxScaler()
mpg[['mpg',
 'cylinders',
 'displacement',
 'weight',
 'acceleration',
 'model year',
 'origin']]
mpg.head()
```

Out[53]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

In [ ]: