Task - 1

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv("V:\\Datasets\\NLP\\ToyotaCorolla.csv")
```

In [3]:
```
df_1 = df.copy(deep=False)
```

In [4]:
```
df_1.head()
```

Out[4]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23	46986	Diesel	90	1	0	2000	3	1165
1	13750	23	72937	Diesel	90	1	0	2000	3	1165
2	13950	24	41711	Diesel	90	1	0	2000	3	1165
3	14950	26	48000	Diesel	90	0	0	2000	3	1165
4	13750	30	38500	Diesel	90	0	0	2000	3	1170

In [5]:
```
# Basic Information of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Price       1436 non-null   int64 
 1   Age         1436 non-null   int64 
 2   KM          1436 non-null   int64 
 3   FuelType    1436 non-null   object
 4   HP          1436 non-null   int64 
 5   MetColor    1436 non-null   int64 
 6   Automatic   1436 non-null   int64 
 7   CC          1436 non-null   int64 
 8   Doors       1436 non-null   int64 
 9   Weight      1436 non-null   int64 
dtypes: int64(9), object(1)
memory usage: 112.3+ KB
```

In [6]:
```
# Dimentions of the Dataset
df.shape
```

Out[6]:
```
(1436, 10)
```

In [7]:
```
# Determine the number of features available.
print("Number of features available in a Dataset: ",df.columns)
print("Number of Columns with respective to rows",df[df.columns].count())
```

```
Number of features available in a Dataset:  Index(['Price', 'Age', 'KM', 'FuelType',
'HP', 'MetColor', 'Automatic', 'CC',
       'Doors', 'Weight'],
      dtype='object')
Number of Columns with respective to rows Price        1436
Age          1436
KM           1436
FuelType     1436
HP           1436
MetColor     1436
Automatic    1436
CC           1436
Doors        1436
Weight       1436
dtype: int64
```

In [8]:

```
# Perform 5 number summary (min, lower quartile, median, upper quartile, max.
df.describe()
```

Out[8]:

	Price	Age	KM	HP	MetColor	Automatic	C
count	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.00000
mean	10730.824513	55.947075	68533.259749	101.502089	0.674791	0.055710	1566.82799
std	3626.964585	18.599988	37506.448872	14.981080	0.468616	0.229441	187.18243
min	4350.000000	1.000000	1.000000	69.000000	0.000000	0.000000	1300.00000
25%	8450.000000	44.000000	43000.000000	90.000000	0.000000	0.000000	1400.00000
50%	9900.000000	61.000000	63389.500000	110.000000	1.000000	0.000000	1600.00000
75%	11950.000000	70.000000	87020.750000	110.000000	1.000000	0.000000	1600.00000
max	32500.000000	80.000000	243000.000000	192.000000	1.000000	1.000000	2000.00000

In [9]:

```
# Access the top 10 rows from the dataset.
df.head(10)
```

Out[9]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23	46986	Diesel	90	1	0	2000	3	1165
1	13750	23	72937	Diesel	90	1	0	2000	3	1165
2	13950	24	41711	Diesel	90	1	0	2000	3	1165
3	14950	26	48000	Diesel	90	0	0	2000	3	1165
4	13750	30	38500	Diesel	90	0	0	2000	3	1170
5	12950	32	61000	Diesel	90	0	0	2000	3	1170
6	16900	27	94612	Diesel	90	1	0	2000	3	1245
7	18600	30	75889	Diesel	90	1	0	2000	3	1245
8	21500	27	19700	Petrol	192	0	0	1800	3	1185
9	12950	23	71138	Diesel	69	0	0	1900	3	1105

In [10]:
```
# Access last 2 rows from the dataset
df.tail(2)
```

Out[10]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
1434	7250	70	16916	Petrol	86	1	0	1300	3	1015
1435	6950	76	1	Petrol	110	0	0	1600	5	1114

Tast - 2

In [11]:
```
# Access a group of rows and columns by label(s).
# ['Price', 'Age', 'KM', 'FuelType']

res = df[["Price","Age","KM","FuelType"]]
print(res)
```

```
      Price  Age     KM FuelType
0     13500   23  46986   Diesel
1     13750   23  72937   Diesel
2     13950   24  41711   Diesel
3     14950   26  48000   Diesel
4     13750   30  38500   Diesel
...     ...  ...    ...      ...
1431   7500   69  20544   Petrol
1432  10845   72  19000   Petrol
1433   8500   71  17016   Petrol
1434   7250   70  16916   Petrol
1435   6950   76      1   Petrol

[1436 rows x 4 columns]
```

In [12]:
```
missing_value = df.isnull()
missing_value.head()
for i in missing_value.columns.values.tolist():
    print(missing_value[i].value_counts())
    print(" ")
```

```
False    1436
Name: Price, dtype: int64

False    1436
Name: Age, dtype: int64

False    1436
Name: KM, dtype: int64

False    1436
Name: FuelType, dtype: int64

False    1436
Name: HP, dtype: int64

False    1436
Name: MetColor, dtype: int64

False    1436
Name: Automatic, dtype: int64

False    1436
```

```
Name: CC, dtype: int64

False    1436
Name: Doors, dtype: int64

False    1436
Name: Weight, dtype: int64
```

In [13]:
```
# Display total number of missing values for each column.
df.isnull().sum()
```

Out[13]:
```
Price        0
Age          0
KM           0
FuelType     0
HP           0
MetColor     0
Automatic    0
CC           0
Doors        0
Weight       0
dtype: int64
```

In [14]:
```
# Replace missing values with mean for continuous variable and mod for categorical v
# Also display the result for total missing values after replacing the missing value
        # In my dataset there is no Missing Values
```

In [15]:
```
df.head()
```

Out[15]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23	46986	Diesel	90	1	0	2000	3	1165
1	13750	23	72937	Diesel	90	1	0	2000	3	1165
2	13950	24	41711	Diesel	90	1	0	2000	3	1165
3	14950	26	48000	Diesel	90	0	0	2000	3	1165
4	13750	30	38500	Diesel	90	0	0	2000	3	1170

In [16]:
```
# Remove the following features from the dataset
# ['CC', 'Doors', 'Weight']
df_1 = df.drop(columns = ["CC","Doors","Weight"],inplace = True)
```

In [17]:
```
df.columns
```

Out[17]:
```
Index(['Price', 'Age', 'KM', 'FuelType', 'HP', 'MetColor', 'Automatic'], dtype='obje
ct')
```

Task - 3

In [28]:
```
# Visualize the data using scatter plot for two features (x='Age', y='Price').Also i
# Provide title, and labels for both axis.
# Apply some marker and set different colors for bar and marker.
```

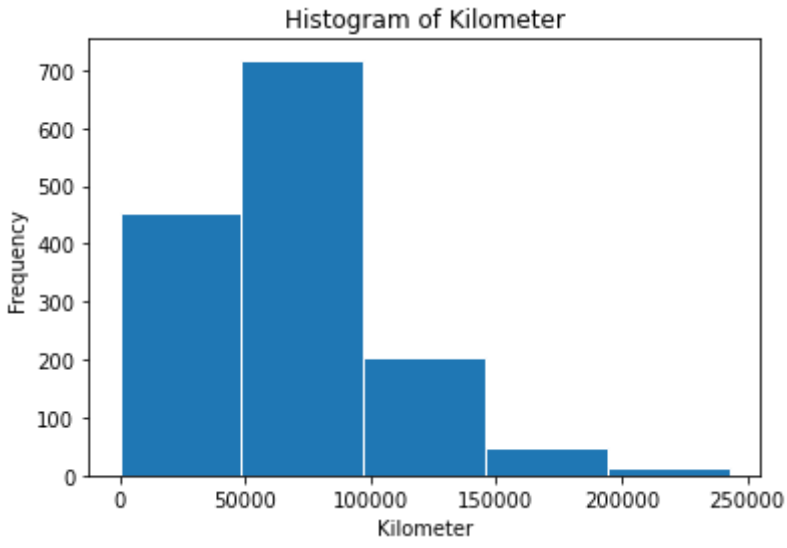
```
plt.scatter(x = df["Age"],y = df["Price"],color = "red", marker = "^")
plt.title("Age & Price")
plt.xlabel("Age")
plt.ylabel("Price")
plt.show()
```

In [36]:

```
# Create a histogram for the feature 'KM'.
    # Also set the following properties:
        # No of bins=5
        # Edge color=White
        # X label= Kilometer
        # Y label= Frequency
        # Title= Histogram of Kilometer

plt.hist(df["KM"],Edgecolor= "White", bins = 5)
plt.xlabel("Kilometer")
plt.ylabel("Frequency")
plt.title("Histogram of Kilometer")
plt.show()
```

```
C:\Users\svasu\AppData\Local\Temp/ipykernel_5548/79081367.py:9: MatplotlibDeprecatio
nWarning: Case-insensitive properties were deprecated in 3.3 and support will be rem
oved two minor releases later
  plt.hist(df["KM"],Edgecolor= "White", bins = 5)
```

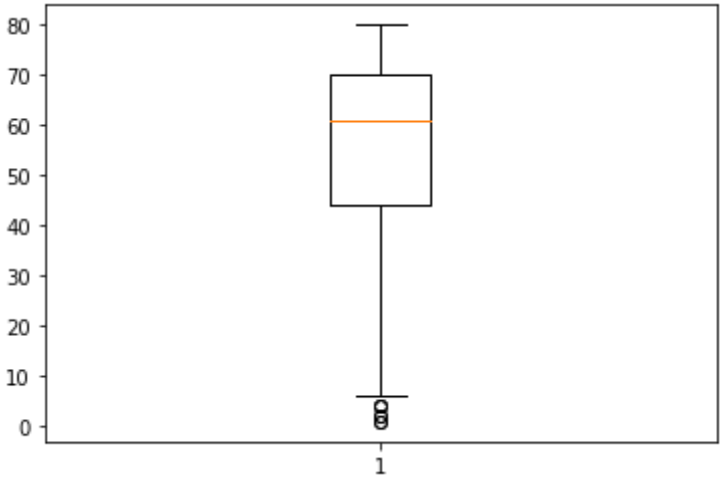


In [43]:

```
# Detect Outliers
        # Apply box and whisker plot to find the outliers in the dataset.
```

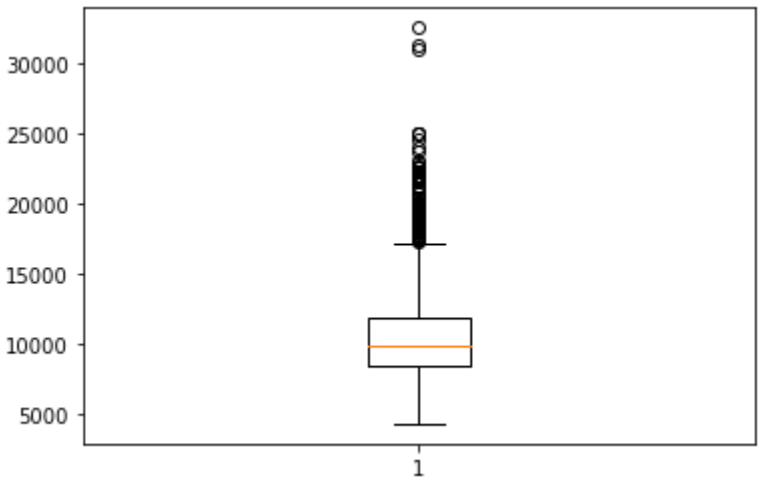
```
        # Also interpret the result.
plt.boxplot("Age",data = df)
```

Out[43]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1dfa279ff10>,
  <matplotlib.lines.Line2D at 0x1dfa279f580>],
 'caps': [<matplotlib.lines.Line2D at 0x1dfa279fbb0>,
  <matplotlib.lines.Line2D at 0x1dfa279f4c0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1dfa279f5b0>],
 'medians': [<matplotlib.lines.Line2D at 0x1dfa2618370>],
 'fliers': [<matplotlib.lines.Line2D at 0x1dfa2618850>],
 'means': []}
```



In [45]:
```
plt.boxplot("Price",data = df)
```

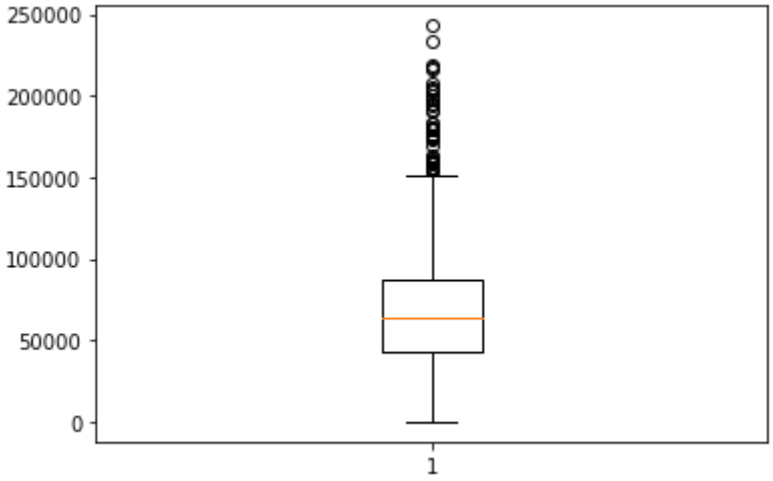
Out[45]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1dfa1a9dd60>,
  <matplotlib.lines.Line2D at 0x1dfa1a9d970>],
 'caps': [<matplotlib.lines.Line2D at 0x1dfa1a9d0d0>,
  <matplotlib.lines.Line2D at 0x1dfa1a2fe50>],
 'boxes': [<matplotlib.lines.Line2D at 0x1dfa1a57820>],
 'medians': [<matplotlib.lines.Line2D at 0x1dfa24ef430>],
 'fliers': [<matplotlib.lines.Line2D at 0x1dfa24efeb0>],
 'means': []}
```



In [46]:
```
plt.boxplot("KM",data = df)
```

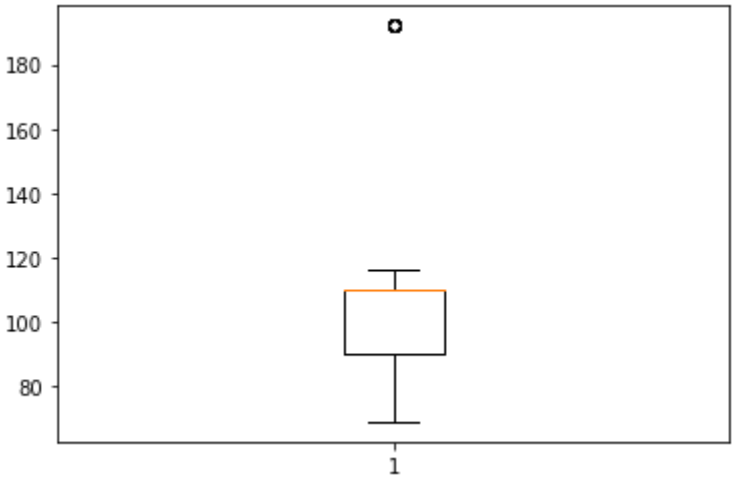
Out[46]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1dfa2550ca0>,
  <matplotlib.lines.Line2D at 0x1dfa2a4d070>],
 'caps': [<matplotlib.lines.Line2D at 0x1dfa2a4d430>,
  <matplotlib.lines.Line2D at 0x1dfa2a4d7c0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1dfa2550910>],
 'medians': [<matplotlib.lines.Line2D at 0x1dfa2a4db50>],
```

```
 'fliers': [<matplotlib.lines.Line2D at 0x1dfa2a4dee0>],
 'means': []}
```



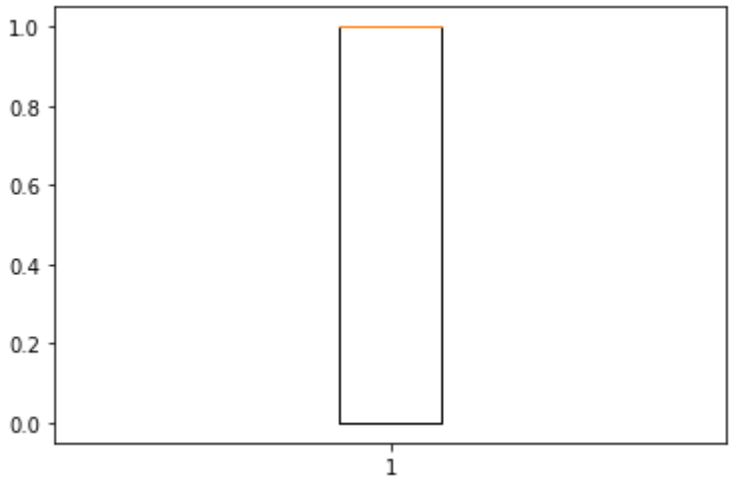
In [48]:
```
plt.boxplot("HP",data = df)
```

Out[48]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1dfa29969d0>,
  <matplotlib.lines.Line2D at 0x1dfa2996550>],
 'caps': [<matplotlib.lines.Line2D at 0x1dfa2996a30>,
  <matplotlib.lines.Line2D at 0x1dfa29b1280>],
 'boxes': [<matplotlib.lines.Line2D at 0x1dfa29965e0>],
 'medians': [<matplotlib.lines.Line2D at 0x1dfa29b1610>],
 'fliers': [<matplotlib.lines.Line2D at 0x1dfa29b19a0>],
 'means': []}
```



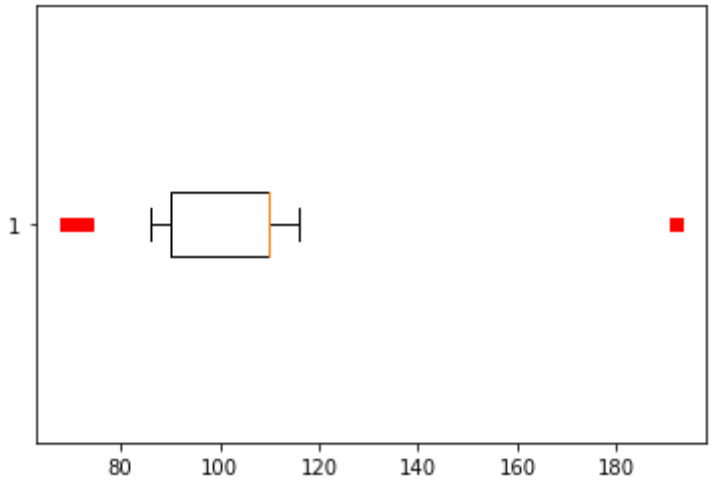
In [49]:
```
plt.boxplot("MetColor",data = df)
```

Out[49]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1dfa2881fa0>,
  <matplotlib.lines.Line2D at 0x1dfa2a1ac10>],
 'caps': [<matplotlib.lines.Line2D at 0x1dfa2a1a7c0>,
  <matplotlib.lines.Line2D at 0x1dfa2a1af10>],
 'boxes': [<matplotlib.lines.Line2D at 0x1dfa2881c70>],
 'medians': [<matplotlib.lines.Line2D at 0x1dfa2a1a550>],
 'fliers': [<matplotlib.lines.Line2D at 0x1dfa2a3a1f0>],
 'means': []}
```



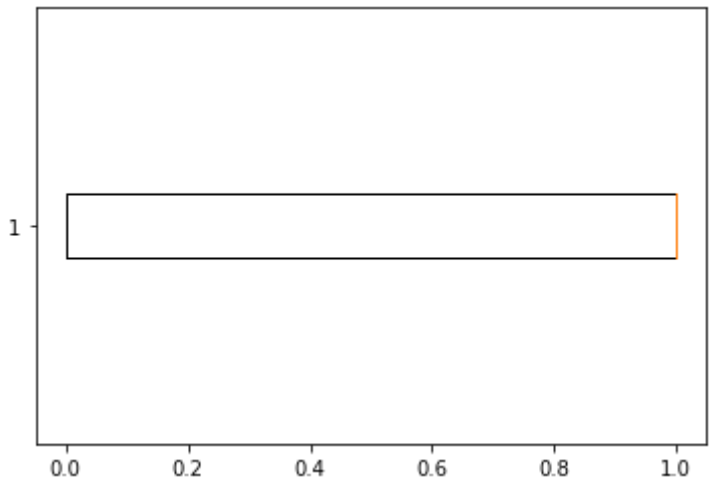
In [54]:

```
plt.figure()
plt.boxplot(df["HP"], 0, 'rs', 0, 0.75)
plt.show()
```



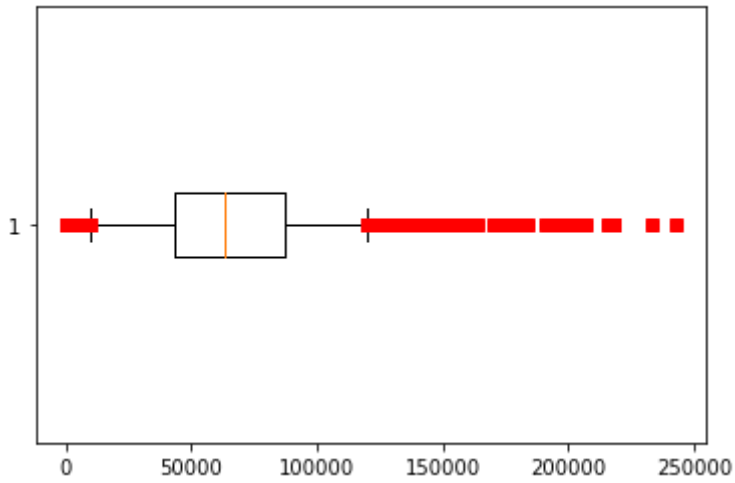
In [56]:

```
plt.figure()
plt.boxplot(df["MetColor"], 0, 'rs', 0, 0.75)
plt.show()
```



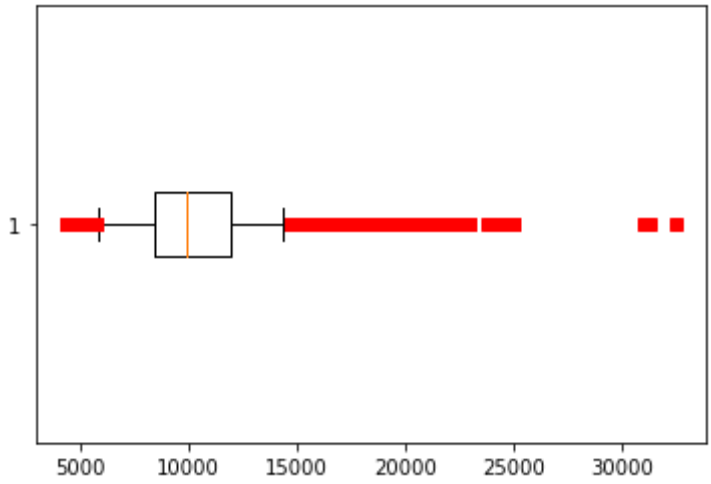
In [57]:

```
plt.figure()
plt.boxplot(df["KM"], 0, 'rs', 0, 0.75)
plt.show()
```



In [58]:

```
plt.figure()
plt.boxplot(df["Price"], 0, 'rs', 0, 0.75)
plt.show()
```



In [59]:

```
plt.figure()
plt.boxplot(df["Age"], 0, 'rs', 0, 0.75)
plt.show()
```