In [275]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [276]:
```
data=pd.read_csv("D:\CarPricePrediction.csv")
data
```

Out[276]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front
...	...	...	...	...	...	...	...	...	...
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front

205 rows × 26 columns

In [278]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

In [279]:
```
data.head(11)
```

Out[279]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front
5	6	2	audi fox	gas	std	two	sedan	fwd	front
6	7	1	audi 100ls	gas	std	four	sedan	fwd	front
7	8	1	audi 5000	gas	std	four	wagon	fwd	front
8	9	1	audi 4000	gas	turbo	four	sedan	fwd	front
9	10	0	audi 5000s (diesel)	gas	turbo	two	hatchback	4wd	front
10	11	2	bmw 320i	gas	std	two	sedan	rwd	front

11 rows × 26 columns

In [280]:
```
data.tail(5)
```

Out[280]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	w
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	

5 rows × 26 columns

In [281]:
```
data=data.drop(['car_ID'],axis=1)
```

In [282]:
```
data['CarName'] = data['CarName'].str.split(' ',expand=True)
```

In [283]:
```
data['CarName'] = data['CarName'].replace({'maxda': 'mazda', 'nissan': 'Nissan'
, 'porcshce': 'porsche', 'toyouta': 'toyota',
                                'vokswagen': 'volkswagen', 'vw': 'volkswagen'})
```

In [284]:
```
data['symboling']=data['symboling'].astype('str')
```

In [285]:
```
categorical_cols=data.select_dtypes(include=['object']).columns
```

In [286]:
```
data[categorical_cols].head(2)
```

Out[286]:

	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	enginetype
0	3	alfa-romero	gas	std	two	convertible	rwd	front	dohc
1	3	alfa-romero	gas	std	two	convertible	rwd	front	dohc

In [287]:
```
numerical_cols=data.select_dtypes(exclude=['object']).columns
```

In [288]:
```
data[numerical_cols].head(2)
```

Out[288]:

	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	h
0	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	9.0	
1	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	9.0	

In [289]:
```
data.describe()
```

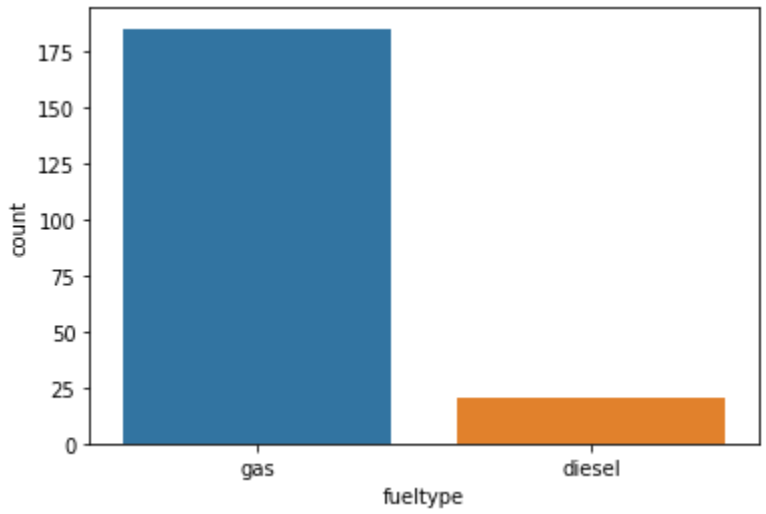
Out[289]:

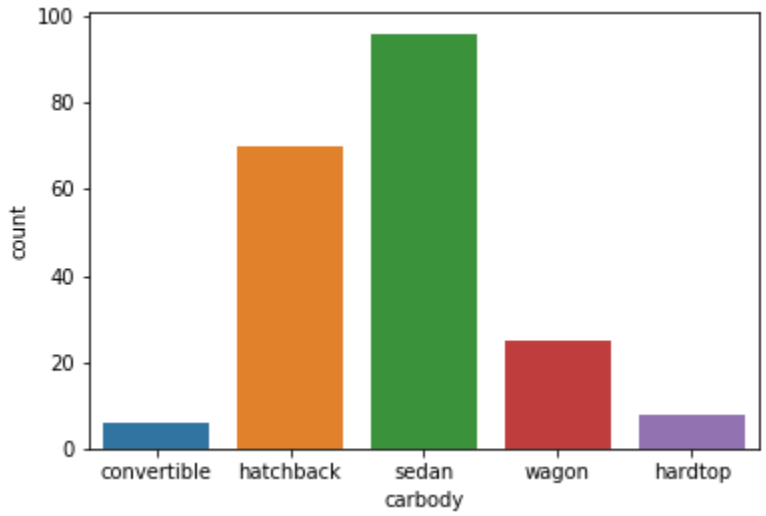
	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	co
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	
mean	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	
std	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597	
min	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	
25%	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	
50%	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	
75%	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	
max	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	

In [290]:
```
data.columns
```

Out[290]:
```
Index(['symboling', 'CarName', 'fueltype', 'aspiration', 'doornumber',
       'carbody', 'drivewheel', 'enginelocation', 'wheelbase', 'carlength',
       'carwidth', 'carheight', 'curbweight', 'enginetype', 'cylindernumber',
       'enginesize', 'fuelsystem', 'boreratio', 'stroke', 'compressionratio',
       'horsepower', 'peakrpm', 'citympg', 'highwaympg', 'price'],
      dtype='object')
```

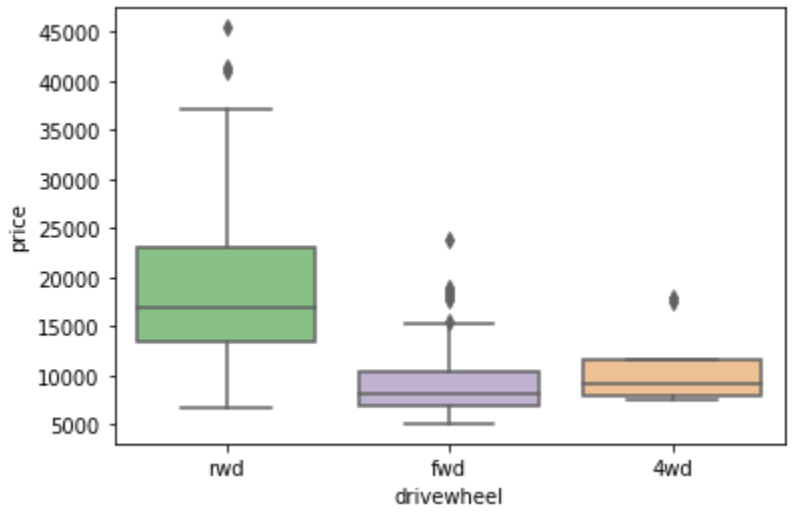
In [291]:
```
sns.countplot(x='fueltype',data=data)
plt.show()

sns.countplot(x='carbody',data=data)
plt.show()
```



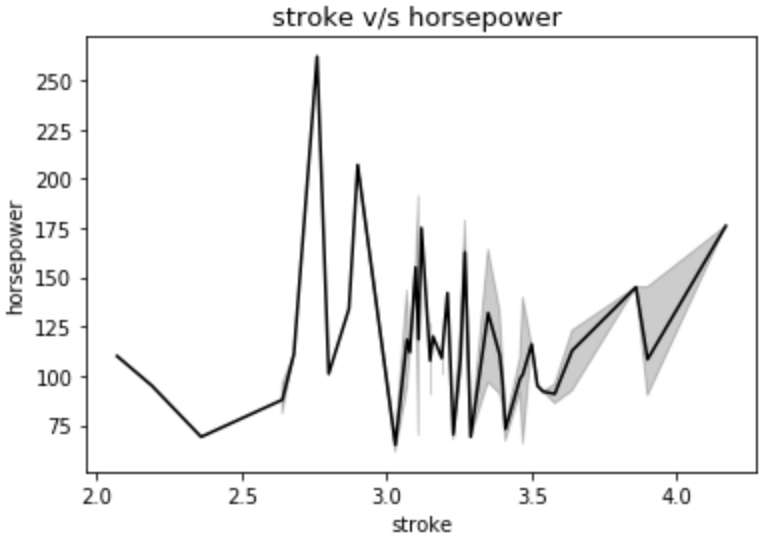


In [292]:
```
sns.boxplot(x = 'drivewheel', y = 'price', data = data,palette='Accent')
```

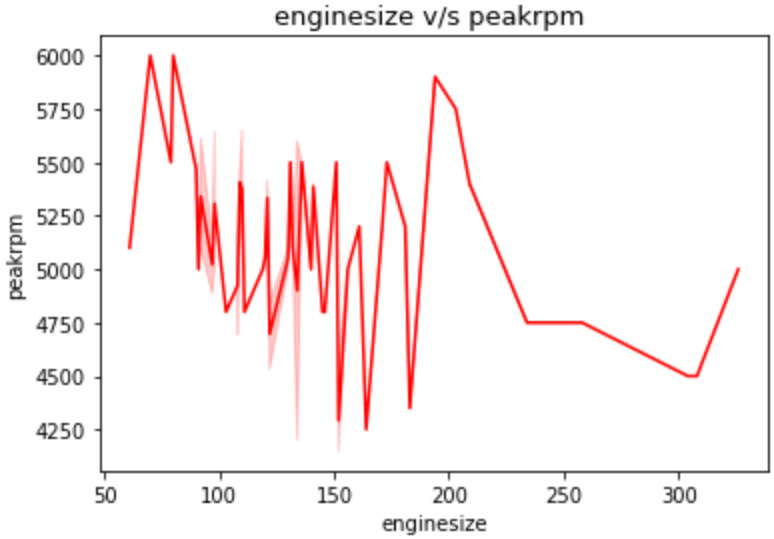
Out[292]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120076c3108>
```



In [293]:
```
sns.lineplot(x='stroke',y="horsepower",data=data,color='black')
plt.title("stroke v/s horsepower ",fontsize=13)
plt.show()
```

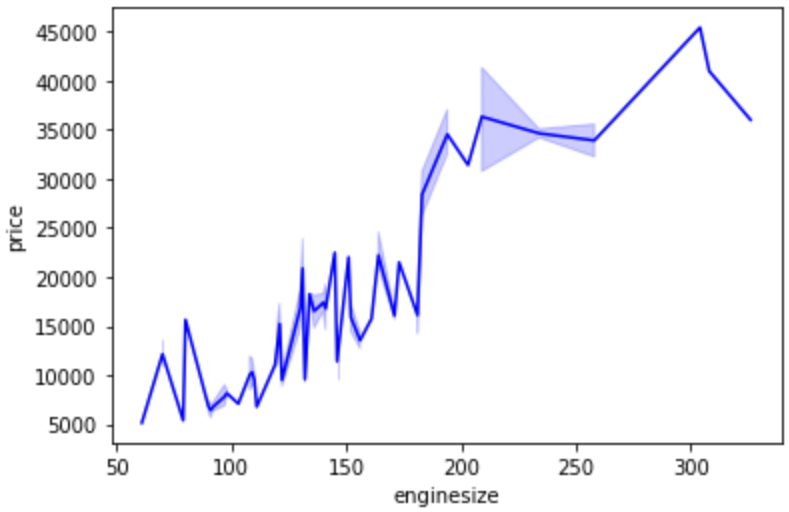


In [328]:
```
sns.lineplot(x='enginesize',y="peakrpm",data=data,color='r')
plt.title("enginesize v/s peakrpm ",fontsize=13)
plt.show()
```

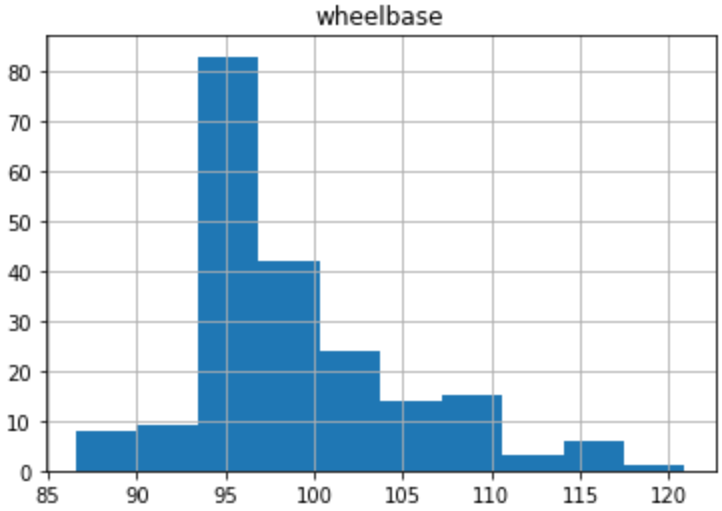


In [295]:
```
sns.lineplot(x='enginesize',y='price',data=data,color='b')
```

Out[295]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1200199d048>
```

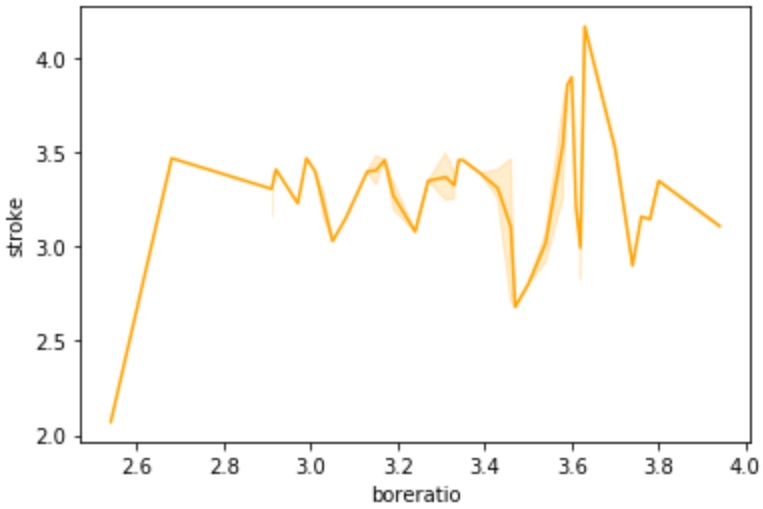


In [296]:
```
data.hist(column=['wheelbase'])
plt.show()
```

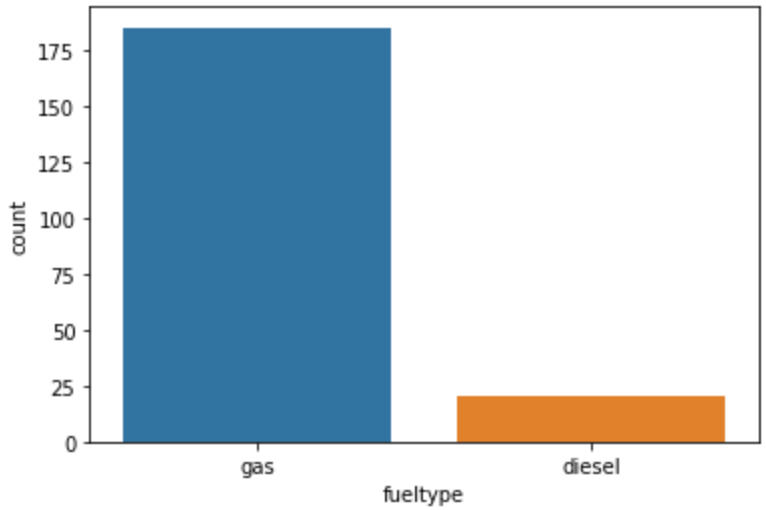


In [297]:
```
sns.lineplot(x='boreratio',y='stroke',data=data,color='orange')
```

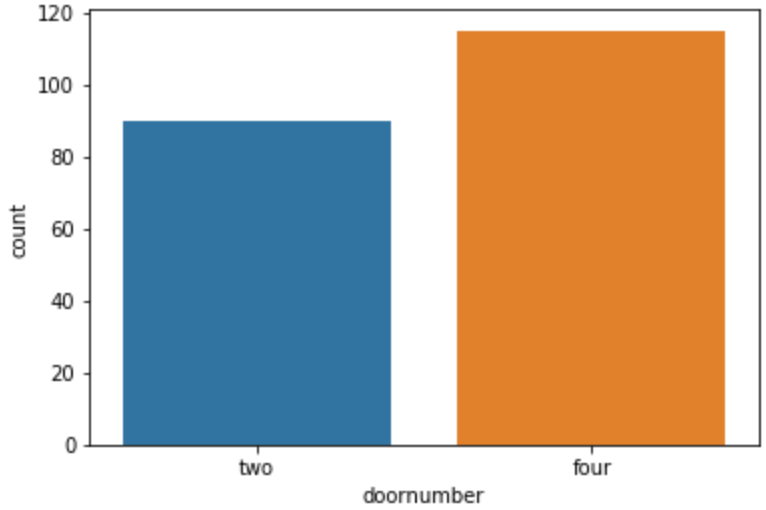
Out[297]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1207595a3c8>
```



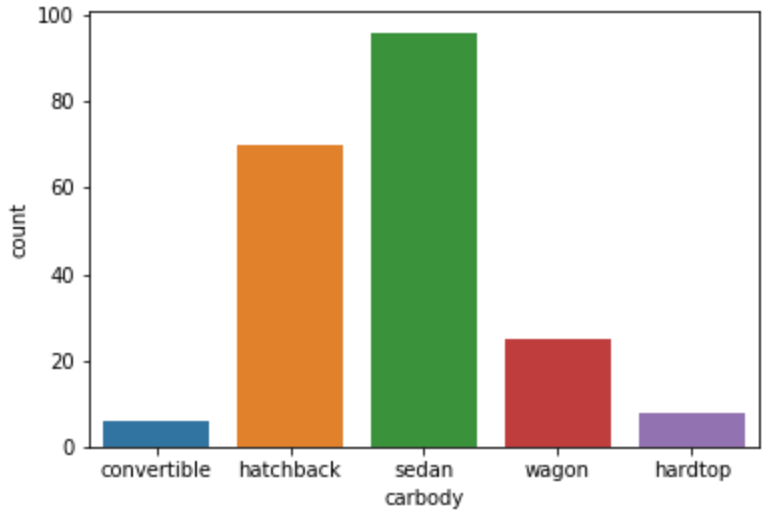
In [298]:
```
sns.countplot(x='fueltype',data=data)
plt.show()
```



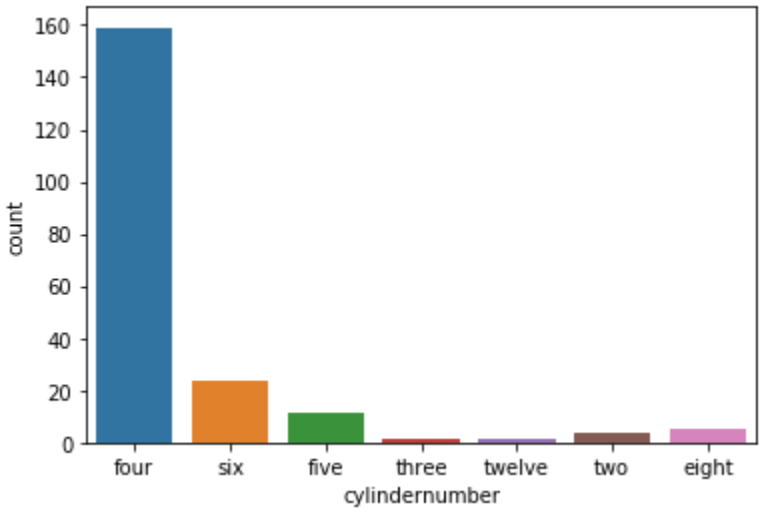
In [299]:
```
sns.countplot(x='doornumber',data=data)
plt.show()
```



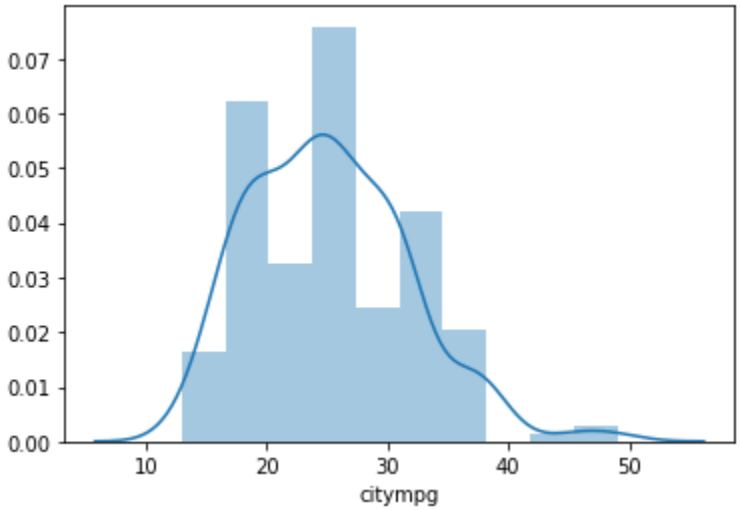
In [300]:
```
sns.countplot(x='carbody',data=data)
plt.show()
```



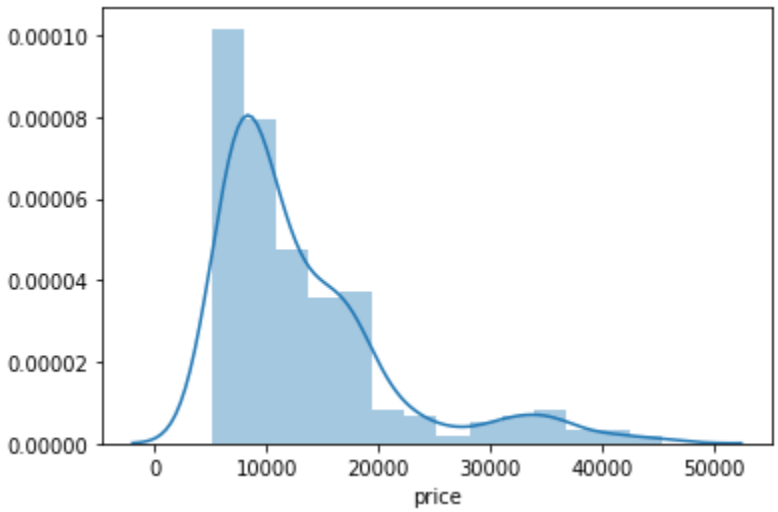
In [301]:
```
sns.countplot(x='cylindernumber',data=data)
plt.show()
```



In [302]:
```
sns.distplot(data['citympg'])
plt.show()
```

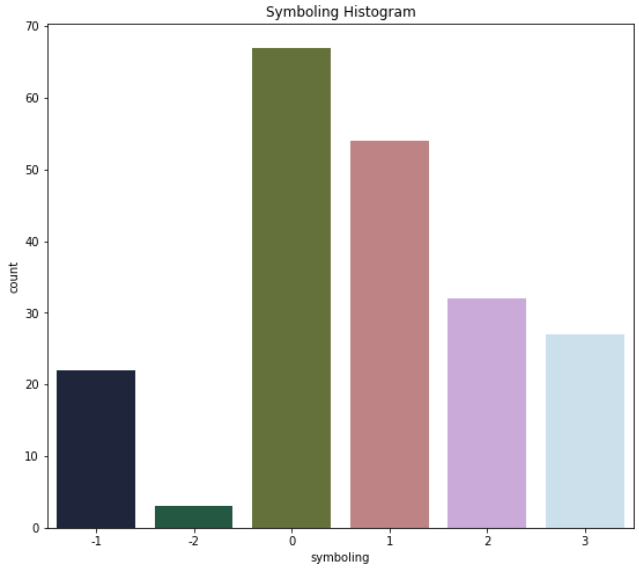


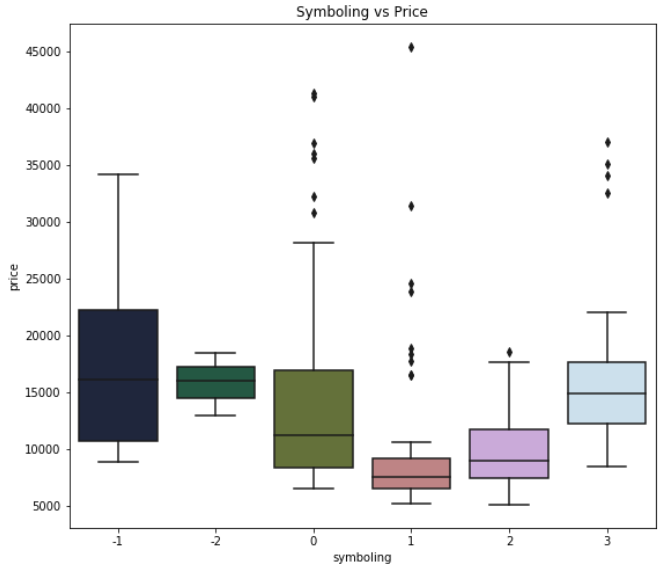
In [303]:
```
sns.distplot(data['price'])
plt.show()
```



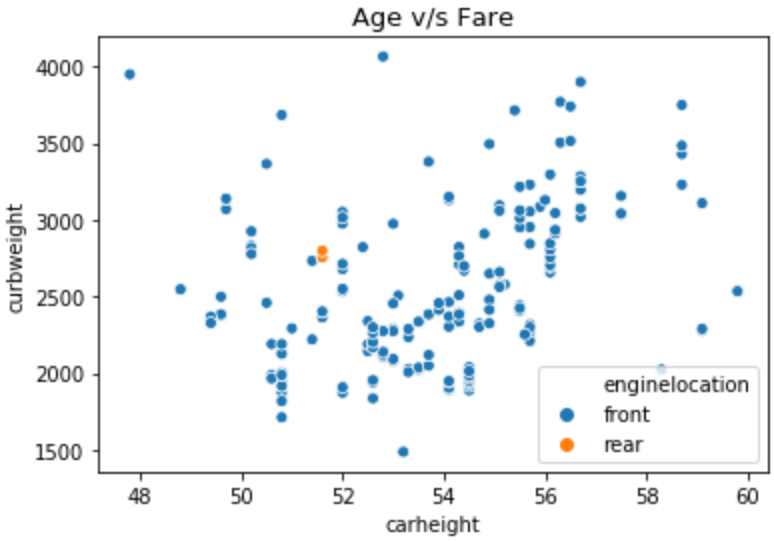
In [304]:

```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('Symboling Histogram')
sns.countplot(data.symboling, palette=("cubehelix"))

plt.subplot(1,2,2)
plt.title('Symboling vs Price')
sns.boxplot(x=data.symboling, y=data.price, palette=("cubehelix"))

plt.show()
```

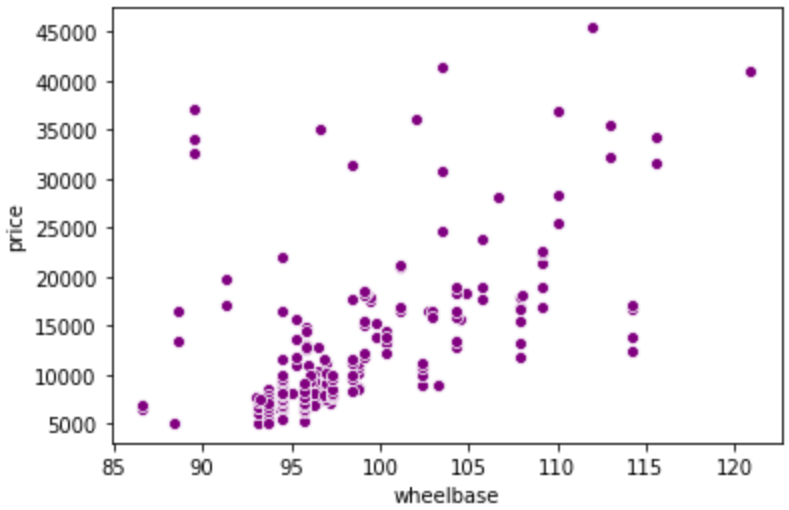




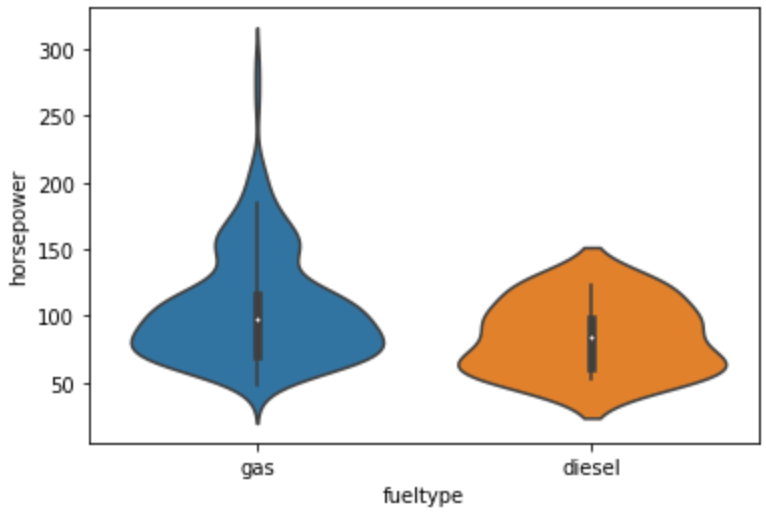
In [305]:
```
sns.scatterplot(x='carheight',y='curbweight',data=data,hue='enginelocation')
plt.title('Age v/s Fare',fontsize=13)
plt.show()

sns.scatterplot(x="wheelbase", y="price", data=data,color='purple')
```

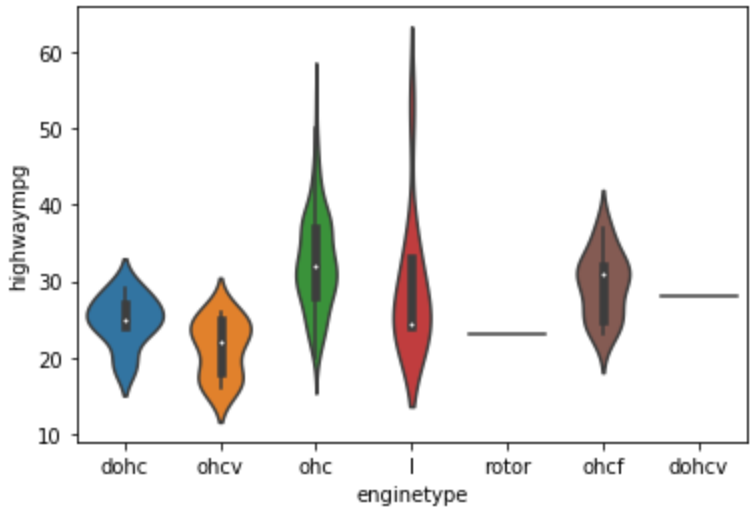


Out[305]: <matplotlib.axes._subplots.AxesSubplot at 0x12000dca108>



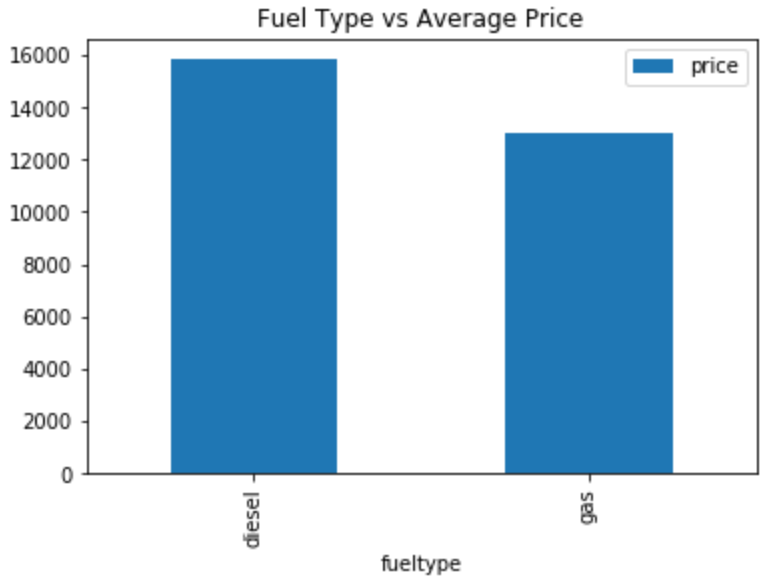
In [306]:
```
sns.violinplot(x='fueltype',y='horsepower',data=data,split=True)
plt.show()

sns.violinplot(x='enginetype',y='highwaympg',data=data,split=True)
plt.show()
```

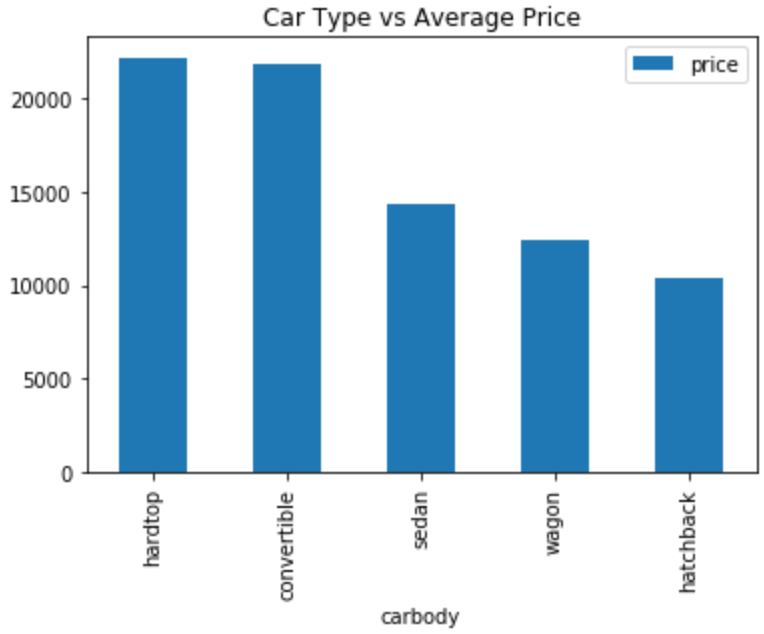




In [307]:
```
plt.figure(figsize=(25, 6))

df = pd.DataFrame(data.groupby(['fueltype'])['price'].mean().sort_values(ascend
ing = False))
df.plot.bar()
plt.title('Fuel Type vs Average Price')
plt.show()

df = pd.DataFrame(data.groupby(['carbody'])['price'].mean().sort_values(ascendi
ng = False))
df.plot.bar()
plt.title('Car Type vs Average Price')
plt.show()
```

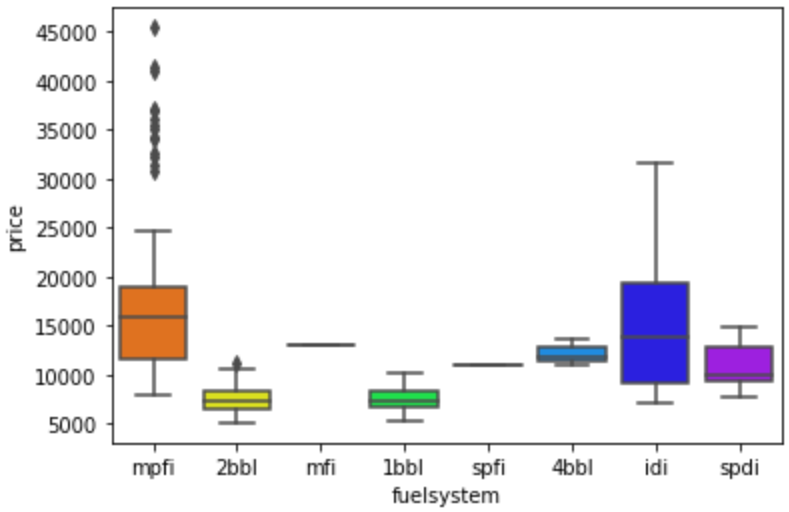
```
<Figure size 1800x432 with 0 Axes>
```



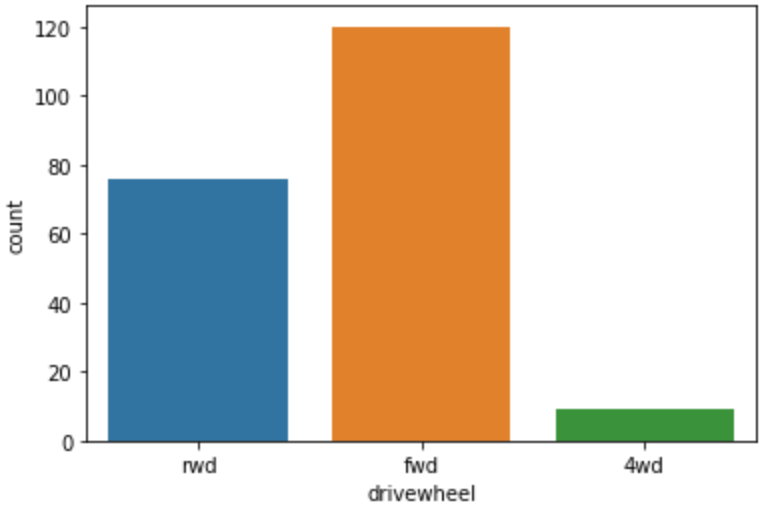


In [308]:
```
sns.boxplot(x = 'fuelsystem', y = 'price', data = data,palette='gist_rainbow')
```

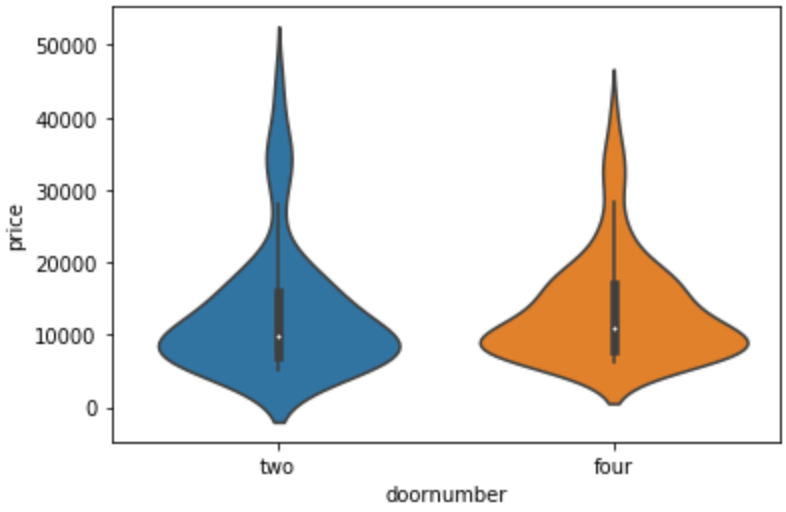
Out[308]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12007b85508>
```



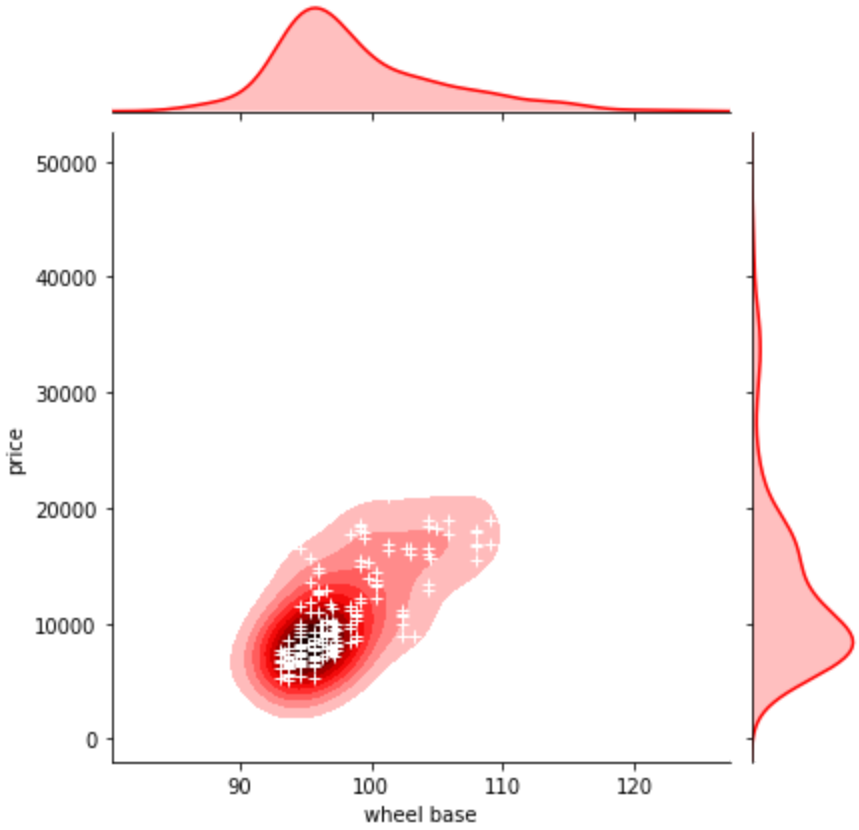
In [309]:
```
sns.countplot(x='drivewheel',data=data)
plt.show()
```



In [310]:
```
sns.violinplot(x='doornumber',y='price',data=data,split=True)
plt.show()
```



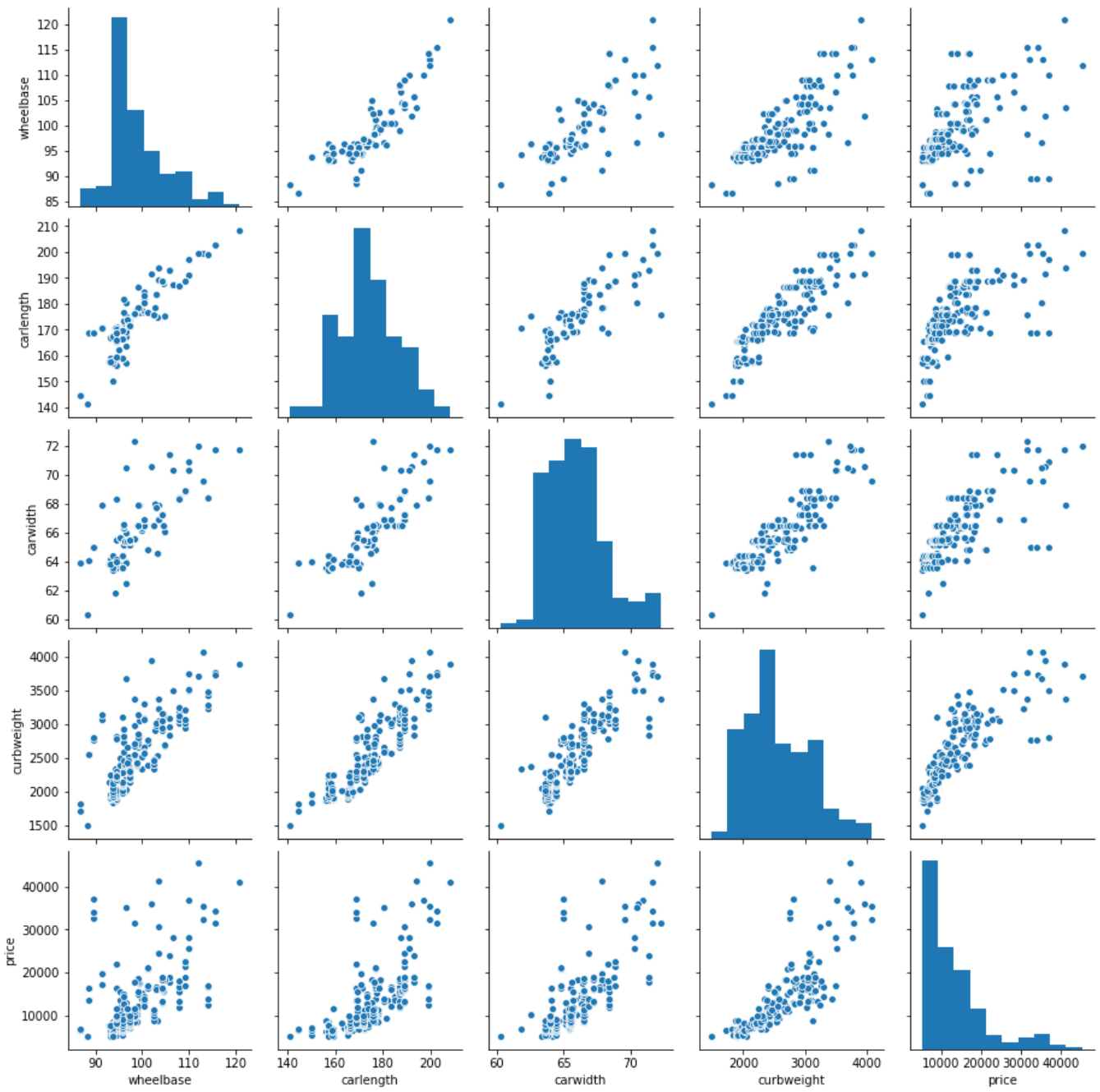
In [311]:
```
g = sns.jointplot(x="wheelbase", y="price", data=data, kind="kde", color="r")
g.plot_joint(plt.scatter, c="w", s=30, linewidth=1, marker="+")
g.ax_joint.collections[0].set_alpha(0)
g.set_axis_labels("wheel base", "price");
```



In [312]:
```
col=['wheelbase','carlength','carwidth','curbweight','price']
```

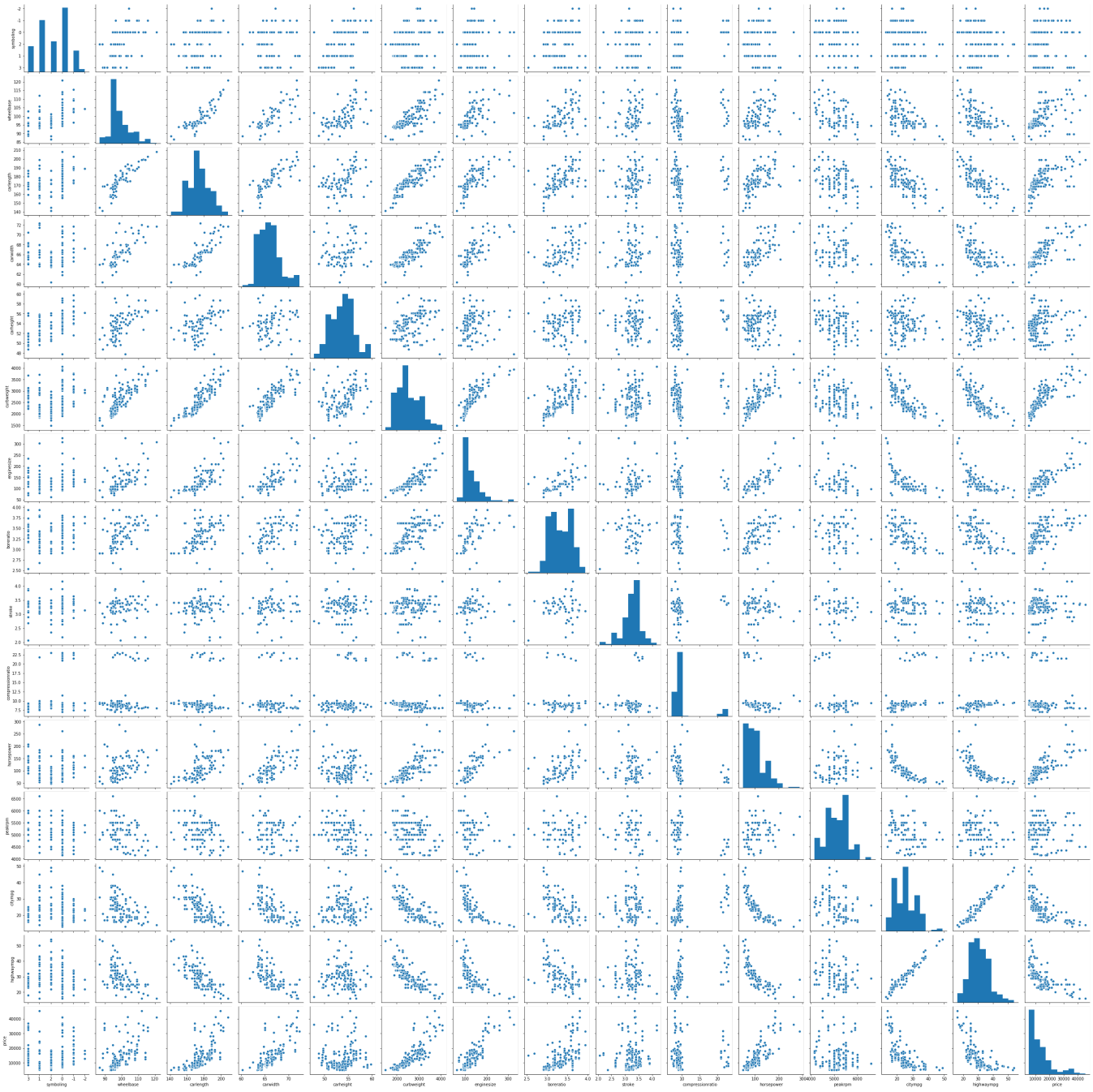
In [313]:
```
sns.pairplot(data[col])
```

Out[313]:
```
<seaborn.axisgrid.PairGrid at 0x12000e7c788>
```

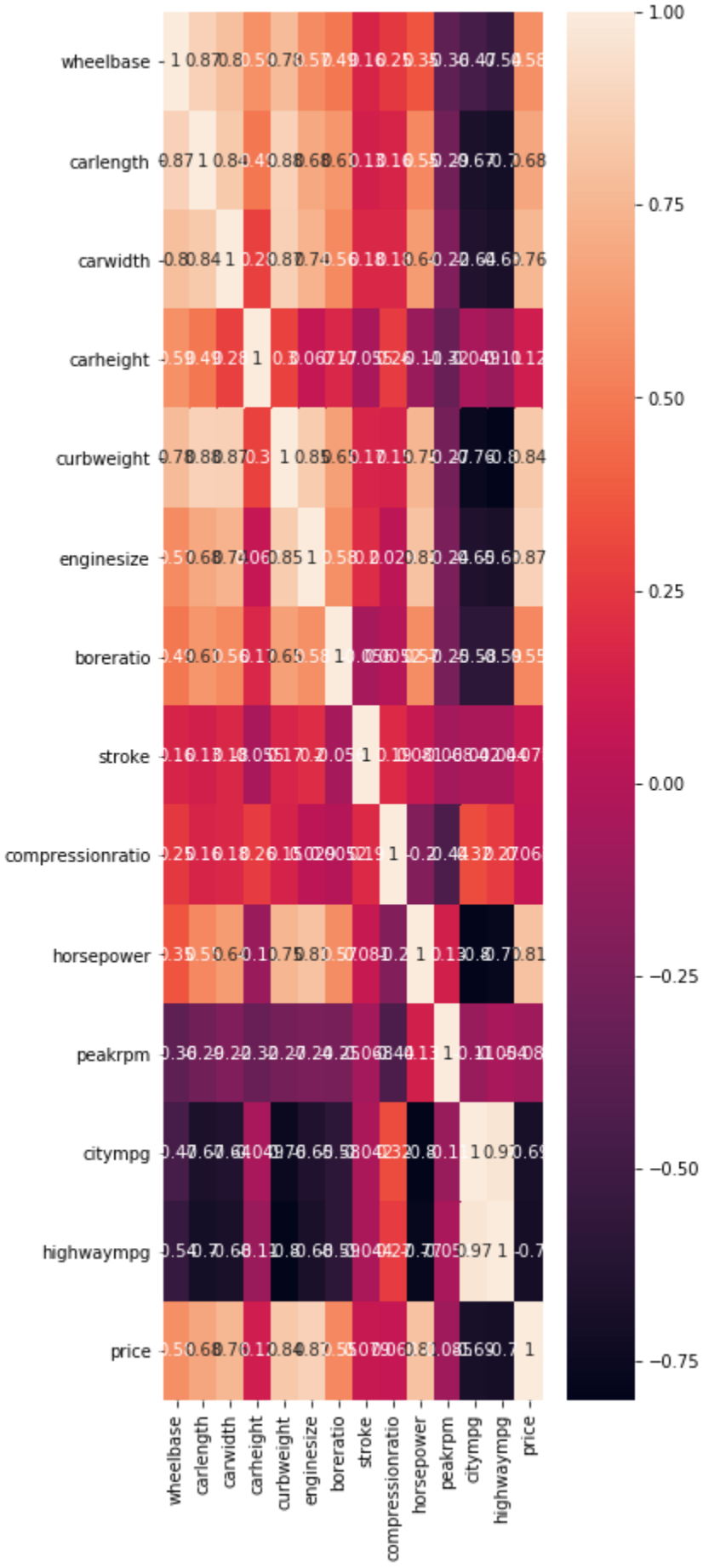


In [314]:

```
sns.pairplot(data)
plt.show()
```

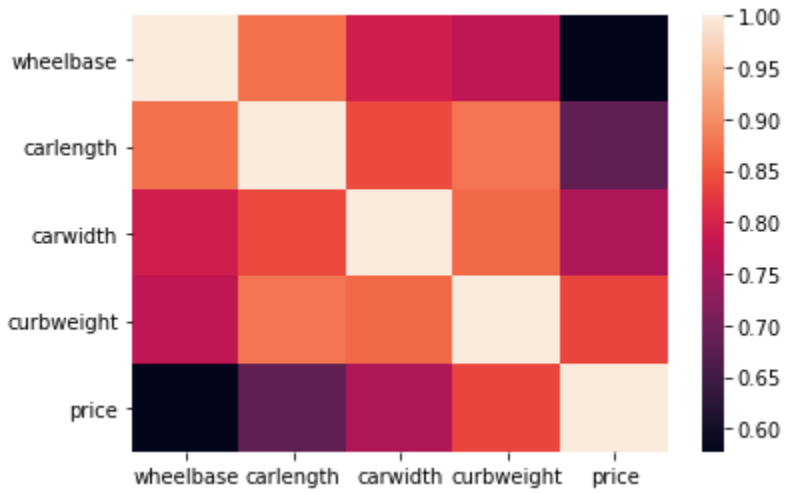


In [315]:
```
plt.figure(figsize=(5,15))
sns.heatmap(data.corr(),annot=True)
plt.show()
```

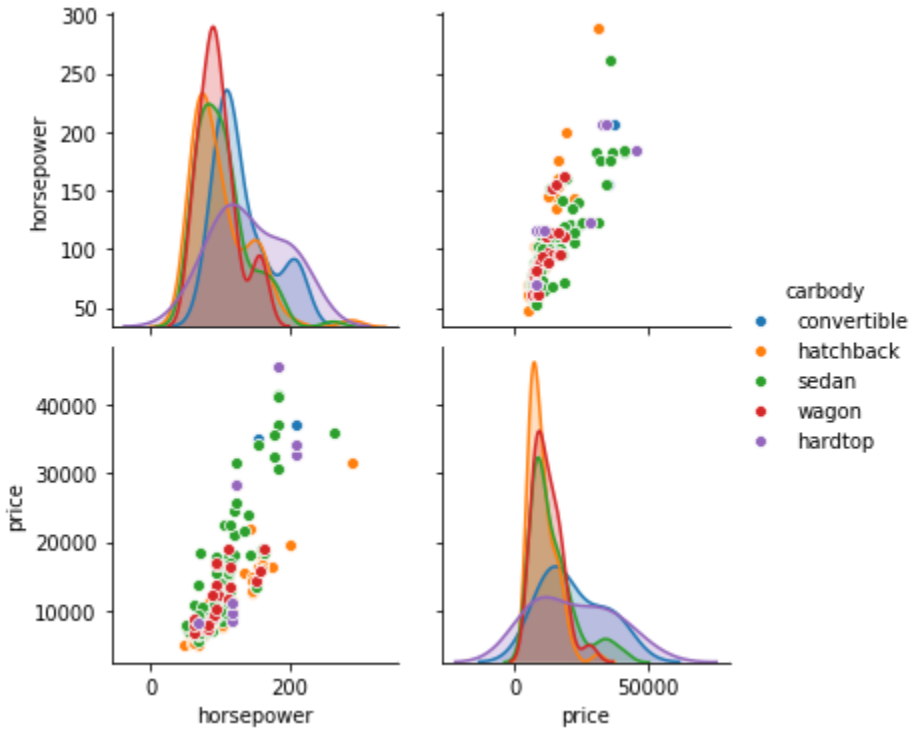


In [316]:
```
sns.heatmap(data[col].corr())
```

Out[316]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1200400c388>
```



In [317]:
```
sns.pairplot(data[['horsepower','price','carbody']], hue="carbody");
```



In [318]:
```
data.isnull()
```

Out[318]:

	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase
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
...	...	...	...	...	...	...	...	...	...
200	False	False	False	False	False	False	False	False	False
201	False	False	False	False	False	False	False	False	False
202	False	False	False	False	False	False	False	False	False
203	False	False	False	False	False	False	False	False	False
204	False	False	False	False	False	False	False	False	False

205 rows × 25 columns

In [319]:
```
data.isnull().sum()
```

Out[319]:
```
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

Feature selection methods are intended to reduce the number of input variables to those that are believed to be most useful to a model in order to predict the target variable.

In [320]:
```
X=data[numerical_cols].drop('price',axis=1)
y=data['price']
```

In [321]:
```
from sklearn import preprocessing
X = data.apply(lambda col: preprocessing.LabelEncoder().fit_transform(col))
X=X.drop(['CarName','price'],axis=1)
y=data['price']
```

In [322]:
```
clf_rf_3 = RandomForestRegressor()
rfe = RFE(estimator=clf_rf_3, n_features_to_select=15, step=1)
rfe = rfe.fit(X, y)
print('Chosen best 15 feature by rfe:',X.columns[rfe.support_])
```

```
Chosen best 15 feature by rfe: Index(['carbody', 'wheelbase', 'carlength', 'carw
idth', 'carheight',
       'curbweight', 'enginetype', 'enginesize', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg'],
      dtype='object')
```

In [323]:
```
features=list(X.columns[rfe.support_])
```

In [324]:
```
x = X[features]
y = data.price
x_train,x_test,y_train,y_test = train_test_split(x,y, random_state = 0)
```

LINEAR REGRESSION

In [325]:
```
#Linear Regression

from sklearn import linear_model
from sklearn.linear_model import LinearRegression
lreg = linear_model.LinearRegression()
lreg.fit(x_train,y_train)
y_train_pred = lreg.predict(x_train)
y_test_pred = lreg.predict(x_test)
lreg.score(x_test,y_test)
```

Out[325]: 0.7299712173396138

DECISION TREE

In [326]:
```
#Decision Tree Regressor

dt_regressor = DecisionTreeRegressor(random_state=0)
dt_regressor.fit(x_train,y_train)
y_train_pred = dt_regressor.predict(x_train)
y_test_pred = dt_regressor.predict(x_test)
dt_regressor.score(x_test,y_test)
```

Out[326]: 0.8699113195927972

RANDOM FOREST

In [327]:
```
#Random Forest regressor

Rf = RandomForestRegressor(n_estimators = 15,
                           criterion = 'mse',
                           random_state = 20,
                           n_jobs = -1)
Rf.fit(x_train,y_train)
Rf_train_pred = Rf.predict(x_train)
Rf_test_pred = Rf.predict(x_test)


r2_score(y_test,Rf_test_pred)
```

Out[327]:
```
0.9146935875209264
```

In [ ]:
```

```