Unit-1 Introduction to Python Pandas

In [ ]:
```
pip install pandas
```

In [ ]:
```
import pandas as pd
print(pd.__version__) # 1.1.3
```

Series :

- List : support
- Tuple : support
- Set : not support
- Dictnory : support
- if ? : that given object

In [ ]:
```
# Series :- 1 column,many rows & 1D
import pandas as pd
a = [1,2,3]
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2])

# Output :

# 0    1
# 1    2
# 2    3
# dtype: int64
```

In [ ]:
```
# Task For Tuple
import pandas as pd
a = (1,2,3)
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2])
# print(myvar[3]) # Key Error
```

In [ ]:
```
# For another data type
import pandas as pd
a = (1.0,2.0,3.0)
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2])
```

In [ ]:
```
import pandas as pd
a = (1.0,2,3.0) # convert to float
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2])
```

In [ ]:
```
import pandas as pd
a = (1.0,2,'a') # give object datatype if any character input
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2])
```

In [ ]:
```
import pandas as pd
a = {1.0,2,'a'}
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2]) # TypeError: 'set' type is unordered
```

In [ ]:
```python
import pandas as pd
a = {'A':1,'B':2,'C':3}
myvar = pd.Series(a)
print(myvar)
print(myvar['A'])
print(myvar['B'])
print(myvar['C'])
```

In [ ]:
```python
import pandas as pd
a = {'A':[1,2],'B':2,'C':3} # if we pass dictniory in list give object.
myvar = pd.Series(a)
print(myvar)
print(myvar['A'])
print(myvar['B'])
print(myvar['C'])
```

In [ ]:
```python
import pandas as pd
a = [1,2,3]
myvar = pd.Series(a,index=['x','y']) # we have must pass 3 values.
print(myvar) # ValueError: Length of passed values is 3, index implies 2.

```

In [ ]:
```python
import pandas as pd
a = [1,2,3]
myvar = pd.Series(a,index=['x','y','z'])
print(myvar)

# Output :
# x    1
# y    2
# z    3
# dtype: int64
```

In [ ]:
```python
import pandas as pd
calories = {'day1':420,'day2':380,'day3':390}
myvar = pd.Series(calories)
print(myvar)

# Output :
# day1    420
# day2    380
# day3    390
# dtype: int64
```

In [ ]:
```python
import pandas as pd
calories = {'day1':420,'day2':380,'day3':390}
myvar = pd.Series(calories,index=['x','y','z'])
print(myvar)

# Output :
# x   NaN
# y   NaN
# z   NaN
# dtype: float64
```

In [ ]:
```python
import pandas as pd
calories = {'day1':420,'day2':380,'day3':390}
myvar = pd.Series(calories,index=['x','y','z','day1'])
print(myvar) # NaN convert automacally float.

# Output :
# x         NaN
# y         NaN
# z         NaN
# day1    420.0
# dtype: float64
```

In [ ]:
```python
import pandas as pd
calories = {'day1':420,'day2':380,'day3':390}
myvar = pd.Series(calories,index=['day2','day1'])
print(myvar) # value must be same of particular key.
```

In [ ]:
```python
a = [1,2,3,4,5,6]
myvar = pd.Series(a)
myvar[[0,1,3]] # when we pass multiple value then using list.
# myvar[0,1,3]
myvar[0::2]
```

DataFrame :(2D)

- many rows many columns

In [ ]:
```
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data)
print(df)

#    calories  duration
# 0       420        50
# 1       380        40
# 2       390        45
```

- loc & iloc(integer location)

- loc : accepts labels as well as int
- iloc: accepts only integer not a string

In [ ]:
```
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data)
print(df['calories'][0])
print(df['calories'].loc[0])
print(df['duration'].loc[1])
```

In [ ]:
```
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data)
print(df)
```

In [ ]:
```
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data,index=['day1','day2','day3'])
# print(df['calories'].loc[0]) # give key error because index is change.
print(df['calories'].loc['day1']) # 420

# Output :
#       calories  duration
# day1       420        50
# day2       380        40
# day3       390        45
```

In [ ]:
```
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data)
print(df['calories'].iloc[0]) # 420
```

In [ ]:
```
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data,index=['day1','day2','day3'])
# print(df['calories'].iloc['day1']) # TypeError:Cannot index by location index with a non-integer key
```

For CSV File.

In [ ]:
```
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df
```

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.info()

# Output :

# <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 [ ]:
```python
help(pd)
```

- head() & tail()

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.head() # given first 5 row print by default when args not pass.
```

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.tail() # given last 5 row print by default when args not pass.
```

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.head(10)
```

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.tail(10)
```

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.loc[34:56]
```

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.loc[[34,56]] # for particular row show then we pass list
```

- df: df ni bajuma hamesha column ave.
- loc: loc ni bajuma hamesha row ave.

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df['mpg'].loc[[34,56]]
```

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df[['mpg','displacement']].loc[[34,56]]
```

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df[['mpg','cylinders']].loc[0:20]
```

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.shape # (398, 9)
```

In [ ]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.loc[-1] # ValueError
```

In [ ]:
```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.loc[:-1] # only give column name.
```

In [ ]:
```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
print(df.loc[:-1])

# Empty DataFrame
# Columns: [mpg, cylinders, displacement, horsepower, weight, acceleration, model year, origin, car name]
# Index: []
```

Statistics

- not analysis of object only Integer & Float.

In [ ]:
```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.describe()
```

Statistics All Operations :

In [ ]:
```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.describe(include="all")
```

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(include=[np.number])
# df.describe(include=[np.object_]) # For laptop[np.object]
```

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(include=[np.object])
```

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(exclude=[np.number])
```

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(exclude=[np.object])
```

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(exclude=["O"])

#  "O" = Object.
```

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df["mpg"].describe()
```

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.loc[0:5].describe()
```

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.iloc[0:5].describe()
```

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df["mpg"].loc[0:5].describe()
```

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(percentiles=[0.3,0.57,0.83])
```

Corr :- corelation of cofficient

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.corr()
# df.corr(numeric_only=True)
```

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('auto-mpg.csv')
pd.plotting.scatter_matrix(df,figsize=[40,40])
plt.show()
```

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('auto-mpg.csv')
pd.plotting.scatter_matrix(df,figsize=[20,20],marker="*",alpha=0.7)
plt.show()
```

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('auto-mpg.csv')
pd.plotting.scatter_matrix(df,figsize=[20,20],diagonal="kde")
plt.show()

# kde = kernal density estimator.
```

In [ ]:
```
import pandas as pd
from pandas.plotting import parallel_coordinates
df = pd.read_csv('auto-mpg.csv')
pll = parallel_coordinates(df,"cylinders",cols=["acceleration","mpg"],color=["red","blue","green","yellow","orange"]
```

In [ ]:
```
import pandas as pd
from pandas.plotting import parallel_coordinates
df = pd.read_csv('auto-mpg.csv')
pll = parallel_coordinates(df,"displacement",cols=["acceleration","mpg"])
```

In [ ]:
```
import pandas as pd
from pandas.plotting import parallel_coordinates
df = pd.read_csv('auto-mpg.csv')
pll = parallel_coordinates(df,"origin",cols=["acceleration","mpg"])
```

In [ ]:
```
import pandas as pd
from pandas.plotting import parallel_coordinates
df = pd.read_csv('auto-mpg.csv')
pll = parallel_coordinates(df,"cylinders",cols=["acceleration","mpg","origin"])
# pll = parallel_coordinates(df,"model year",cols=["acceleration","mpg","origin"])
```

In [ ]:
```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
pd.crosstab(df["cylinders"],df["model year"],
rownames = ["cylinders"],colnames=["model year"])
```

Data cleaning :-

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.isna() # boolean type
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.isna().sum()
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna() # nan value row remove.
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(how="all") # how=all :- je row ma badha nan hoy to ej row kadhe.
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(how="any")
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(subset=["sales"])
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(subset=["sales","region"])
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(thresh=3) # thresh check value atleast 3(3 or>3)
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(thresh=2) # thresh check value atleast 2(2 or>2)
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(axis=0) # axis 0 = rowwise
```

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

sales_data = pd.DataFrame({'name':['a','b','c','d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(axis=1) # axis 1 = columnwise
```

fillna()

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.fillna(0)
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.fillna(sales_data["sales"].mean())
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].mean())
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].median())
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].mode()
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].mode()[0])
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].mode()[1])
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].mode()[2])
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,30,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,30,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].mode()
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,30,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].mode()[0])
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,20,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].mode()
sales_data
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,20,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(inplace=True)
sales_data
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,20,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data = sales_data.dropna()
sales_data
```

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

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,20,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(30,inplace=True)
sales_data
```

In [5]:
```
import pandas as pd
data = {
    'A':['TA','TB','TB','TC','TA'],
    'B':[50,40,40,30,50],
    'C':[True,False,False,False,True]
}
df = pd.DataFrame(data)
dups = df.duplicated()
print(dups)
```

```
0    False
1    False
2     True
3    False
4     True
dtype: bool
```

In [6]:
```
import pandas as pd
data = {
    'A':['TA','TB','TB','TC','TA'],
    'B':[50,40,40,30,50],
    'C':[True,False,False,False,True]
}
df = pd.DataFrame(data)
dups = df.duplicated()
print(dups)

df = df.drop_duplicates()
df
```

```
0    False
1    False
2     True
3    False
4     True
dtype: bool
```

Out[6]:

	A	B	C
0	TA	50	True
1	TB	40	False
3	TC	30	False

In [7]:
```
import pandas as pd
data = {
    'A':['TA','TB','TB','TC','TA'],
    'B':[50,40,40,30,50],
    'C':[True,False,False,False,True]
}
df = pd.DataFrame(data)
dups = df.duplicated()
print(dups)

df = df.reset_index(drop=True)
df
```

```
0    False
1    False
2     True
3    False
4     True
dtype: bool
```

Out[7]:

	A	B	C
0	TA	50	True
1	TB	40	False
2	TB	40	False
3	TC	30	False
4	TA	50	True

In [8]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df
```

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
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

In [9]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df['horsepower']=="?"
```

Out[9]:
```
0      False
1      False
2      False
3      False
4      False
       ...
393    False
394    False
395    False
396    False
397    False
Name: horsepower, Length: 398, dtype: bool
```

In [10]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df[df['horsepower']=="?"]
```

Out[10]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
32	25.0	4	98.0	?	2046	19.0	71	1	ford pinto
126	21.0	6	200.0	?	2875	17.0	74	1	ford maverick
330	40.9	4	85.0	?	1835	17.3	80	2	renault lecar deluxe
336	23.6	4	140.0	?	2905	14.3	80	1	ford mustang cobra
354	34.5	4	100.0	?	2320	15.8	81	2	renault 18i
374	23.0	4	151.0	?	3035	20.5	82	1	amc concord dl

In [13]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('auto-mpg.csv')
df.loc[[32,126,330,336,354,374]]
```

Out[13]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
32	25.0	4	98.0	?	2046	19.0	71	1	ford pinto
126	21.0	6	200.0	?	2875	17.0	74	1	ford maverick
330	40.9	4	85.0	?	1835	17.3	80	2	renault lecar deluxe
336	23.6	4	140.0	?	2905	14.3	80	1	ford mustang cobra
354	34.5	4	100.0	?	2320	15.8	81	2	renault 18i
374	23.0	4	151.0	?	3035	20.5	82	1	amc concord dl

In [15]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('auto-mpg.csv')
df[df['horsepower']!="?"]
```

Out[15]:

	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
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

392 rows × 9 columns

In [16]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('auto-mpg.csv')
df.drop('mpg',axis=1) # axis 1 column delete.
```

Out[16]:

	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...
393	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	4	97.0	52	2130	24.6	82	2	vw pickup
395	4	135.0	84	2295	11.6	82	1	dodge rampage
396	4	120.0	79	2625	18.6	82	1	ford ranger
397	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 8 columns

In [17]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('auto-mpg.csv')
df.drop(2,axis=0) # 2 nd row will be delete.
```

Out[17]:

	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
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
5	15.0	8	429.0	198	4341	10.0	70	1	ford galaxie 500
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

397 rows × 9 columns

Outlier :

In [25]:
```
import pandas as pd
import numpy as np

def find_outlier(ds,col):
    Q1 = ds[col].quantile(0.25)
    Q3 = ds[col].quantile(0.75)
    IQR = Q3 - Q1
    low_val = Q1 - (1.5*IQR)
    high_val = Q3 + (1.5*IQR)
    ds = ds.loc[(ds[col]<low_val)|(ds[col]>high_val)]
    return ds

df = pd.read_csv('auto-mpg.csv')
find_outlier(df,"mpg")
```

Out[25]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
322	46.6	4	86.0	65	2110	17.9	80	3	mazda glc

In [24]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('auto-mpg.csv')
df.describe()
```

Out[24]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

In [26]:
```
import pandas as pd
import numpy as np

def find_outlier(ds,col):
    Q1 = ds[col].quantile(0.25)
    Q3 = ds[col].quantile(0.75)
    IQR = Q3 - Q1
    low_val = Q1 - (1.5*IQR)
    high_val = Q3 + (1.5*IQR)
    ds = ds.loc[(ds[col]<low_val)|(ds[col]>high_val)]
    return ds

df = pd.read_csv('auto-mpg.csv')
find_outlier(df,"acceleration")
```

Out[26]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
7	14.0	8	440.0	215	4312	8.5	70	1	plymouth fury iii
9	15.0	8	390.0	190	3850	8.5	70	1	amc ambassador dpl
11	14.0	8	340.0	160	3609	8.0	70	1	plymouth 'cuda 340
59	23.0	4	97.0	54	2254	23.5	72	2	volkswagen type 3
195	29.0	4	85.0	52	2035	22.2	76	1	chevrolet chevette
299	27.2	4	141.0	71	3190	24.8	79	2	peugeot 504
300	23.9	8	260.0	90	3420	22.2	79	1	oldsmobile cutlass salon brougham
326	43.4	4	90.0	48	2335	23.7	80	2	vw dasher (diesel)
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup

In [27]:
```
import pandas as pd
import numpy as np

def find_outlier(ds,col):
    Q1 = ds[col].quantile(0.25)
    Q3 = ds[col].quantile(0.75)
    IQR = Q3 - Q1
    low_val = Q1 - (1.5*IQR)
    high_val = Q3 + (1.5*IQR)
    ds = ds.loc[(ds[col]>low_val)&(ds[col]<high_val)]
    return ds

df = pd.read_csv('auto-mpg.csv')
find_outlier(df,"mpg")
```

Out[27]:

	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
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

397 rows × 9 columns

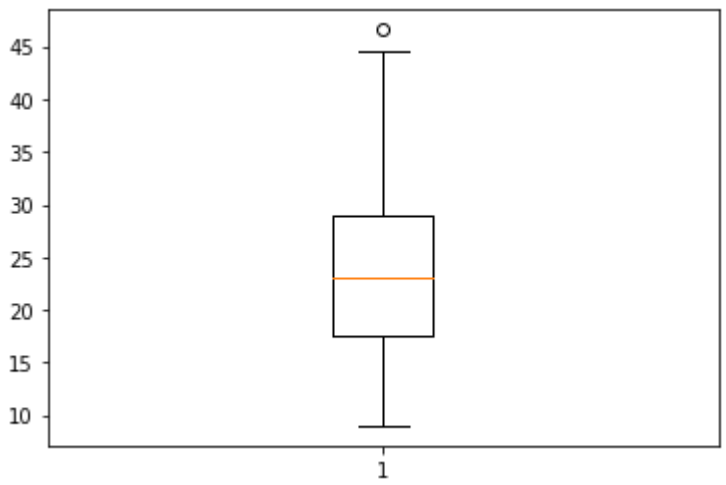
In [29]:
```
import pandas as pd
import numpy as np

def find_outlier(ds,col):
    Q1 = ds[col].quantile(0.25)
    Q3 = ds[col].quantile(0.75)
    IQR = Q3 - Q1
    low_val = Q1 - (1.5*IQR)
    high_val = Q3 + (1.5*IQR)
    ds = ds.loc[(ds[col]>low_val)&(ds[col]<high_val)]
    return ds

df = pd.read_csv('auto-mpg.csv')
find_outlier(df,"acceleration")
```

Out[29]:

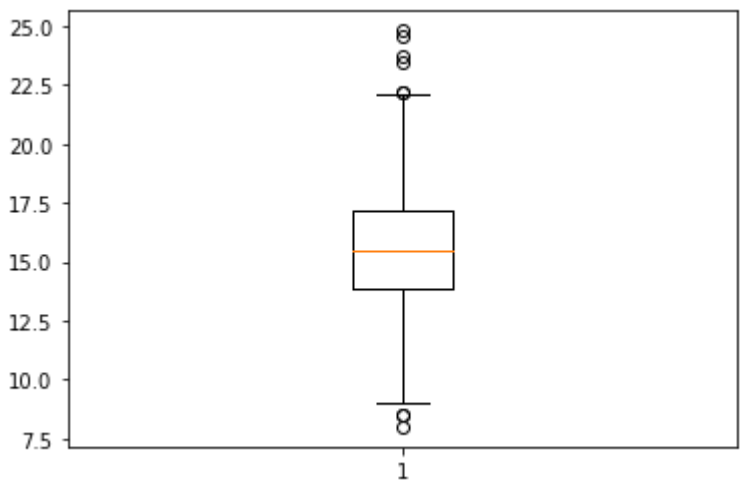
	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
392	27.0	4	151.0	90	2950	17.3	82	1	chevrolet camaro
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

389 rows × 9 columns

In [31]:
```
import matplotlib.pyplot as plt
plt.boxplot(df["mpg"])
plt.show()
```



In [32]:
```
import matplotlib.pyplot as plt
plt.boxplot(df["acceleration"])
plt.show()
```



In [ ]:
```

```