In [5]:
```
from sklearn import datasets
import pandas as pd
iris=datasets.load_iris()
print(iris)
```

```
       [6.2, 2.2, 4.5, 1.5],
       [5.6, 2.5, 3.9, 1.1],
       [5.9, 3.2, 4.8, 1.8],
       [6.1, 2.8, 4. , 1.3],
       [6.3, 2.5, 4.9, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [6.4, 2.9, 4.3, 1.3],
       [6.6, 3. , 4.4, 1.4],
       [6.8, 2.8, 4.8, 1.4],
       [6.7, 3. , 5. , 1.7],
       [6. , 2.9, 4.5, 1.5],
       [5.7, 2.6, 3.5, 1. ],
       [5.5, 2.4, 3.8, 1.1],
       [5.5, 2.4, 3.7, 1. ],
       [5.8, 2.7, 3.9, 1.2],
       [6. , 2.7, 5.1, 1.6],
       [5.4, 3. , 4.5, 1.5],
       [6. , 3.4, 4.5, 1.6],
       [6.7, 3.1, 4.7, 1.5],
```

In [6]:
```
print(type(iris))
```

```
<class 'sklearn.utils.Bunch'>
```

In [7]:
```
print(iris.keys())
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_na
mes', 'filename', 'data_module'])
```

In [8]:
```
print(type(object))
```

```
<class 'type'>
```

In [9]:
```
print(type(iris.data))
```

```
<class 'numpy.ndarray'>
```

In [46]:
```
print(type(iris.target))
```

```
<class 'numpy.ndarray'>
```

In [11]:
```
print(iris.data.shape)
```

```
(150, 4)
```

In [12]:
```
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

In [13]:
```
x=iris.data
y=iris.target
print(x)
print(y)
```

```
 [7.2 3.  5.8 1.6]
 [7.4 2.8 6.1 1.9]
 [7.9 3.8 6.4 2. ]
 [6.4 2.8 5.6 2.2]
 [6.3 2.8 5.1 1.5]
 [6.1 2.6 5.6 1.4]
 [7.7 3.  6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
```

In [16]:
```
ir=pd.DataFrame(x,columns=iris.feature_names)
print(ir)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)
0                  5.1               3.5                1.4
0.2
1                  4.9               3.0                1.4
0.2
2                  4.7               3.2                1.3
0.2
3                  4.6               3.1                1.5
0.2
4                  5.0               3.6                1.4
0.2
..                 ...               ...                ...
...
145                6.7               3.0                5.2
2.3
146                6.3               2.5                5.0
1.9
147                6.5               3.0                5.2
2.0
148                6.2               3.4                5.4
2.3
149                5.9               3.0                5.1
1.8

[150 rows x 4 columns]
```

In [17]:
```
print(ir.head())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (c
m)
0                5.1               3.5                1.4               0.
2
1                4.9               3.0                1.4               0.
2
2                4.7               3.2                1.3               0.
2
3                4.6               3.1                1.5               0.
2
4                5.0               3.6                1.4               0.
2
```

In [18]:
```
print(ir.tail())
```

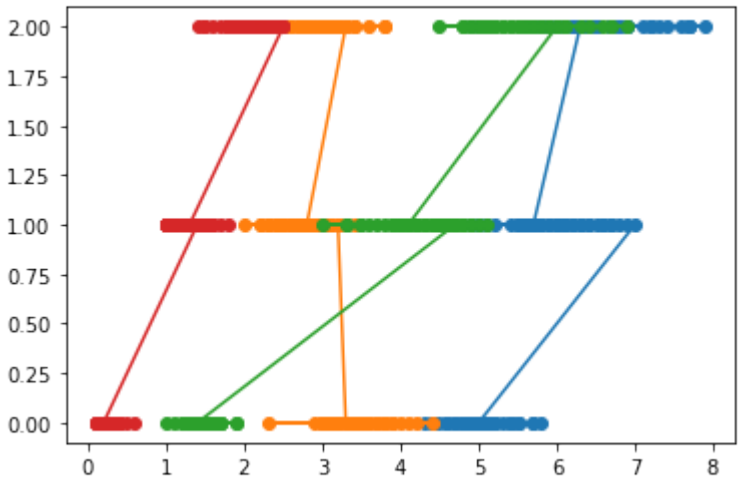
```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)
145                6.7               3.0                5.2
2.3
146                6.3               2.5                5.0
1.9
147                6.5               3.0                5.2
2.0
148                6.2               3.4                5.4
2.3
149                5.9               3.0                5.1
1.8
```

In [19]:
```
print(ir.describe())
```

```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000
```

In [33]:
```
import matplotlib.pyplot
import matplotlib.pyplot as pl
pl.plot(x,y,marker='o')
pl.show()
```



In [ ]:
```
```

In [ ]:
```
print(ir.min())
```

In [22]:
```
print(ir.max())
```

```
sepal length (cm)    7.9
sepal width (cm)     4.4
petal length (cm)    6.9
petal width (cm)     2.5
dtype: float64
```

In [23]:
```
from sklearn import datasets
import pandas as pd
d=datasets.load_diabetes()
print(d)
```

{'data': array([[0.03807591, 0.05068012, 0.06169621, ..., -0.00259226,
         0.01990842, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06832974, -0.09220405],
       [0.08529891, 0.05068012, 0.04445121, ..., -0.00259226,
         0.00286377, -0.02593034],
       ...,
       [0.04170844, 0.05068012, -0.01590626, ..., -0.01107952,
        -0.04687948, 0.01549073],
       [-0.04547248, -0.04464164, 0.03906215, ..., 0.02655962,
         0.04452837, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00421986, 0.00306441]]), 'target': array([151., 75., 141., 206., 135., 97., 138., 63., 110., 310., 101.,
        69., 179., 185., 118., 171., 166., 144., 97., 168., 68., 49.,
        68., 245., 184., 202., 137., 85., 131., 283., 129., 59., 341.,
        87., 65., 102., 265., 276., 252., 90., 100., 55., 61., 92.,
       259., 53., 190., 142., 75., 142., 155., 225., 59., 104., 182.,
       128., 52., 37., 170., 170., 61., 144., 52., 128., 71., 163.,
       150., 97., 160., 178., 48., 270., 202., 111., 85., 42., 170.,
       200., 252., 113., 143., 51., 52., 210., 65., 141., 55., 134.,
        42., 111., 98., 164., 48., 96., 90., 162., 150., 279., 92.,
        83., 128., 102., 302., 198., 95., 53., 134., 144., 232., 81.,
       104., 59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
       173., 180., 84., 121., 161., 99., 109., 115., 268., 274., 158.,
       107., 83., 103., 272., 85., 280., 336., 281., 118., 317., 235.,
        60., 174., 259., 178., 128., 96., 126., 288., 88., 292., 71.,
       197., 186., 25., 84., 96., 195., 53., 217., 172., 131., 214.,
        59., 70., 220., 268., 152., 47., 74., 295., 101., 151., 127.,
       237., 225., 81., 151., 107., 64., 138., 185., 265., 101., 137.,
       143., 141., 79., 292., 178., 91., 116., 86., 122., 72., 129.,
       142., 90., 158., 39., 196., 222., 277., 99., 196., 202., 155.,
        77., 191., 70., 73., 49., 65., 263., 248., 296., 214., 185.,
        78., 93., 252., 150., 77., 208., 77., 108., 160., 53., 220.,
       154., 259., 90., 246., 124., 67., 72., 257., 262., 275., 177.,
        71., 47., 187., 125., 78., 51., 258., 215., 303., 243., 91.,
       150., 310., 153., 346., 63., 89., 50., 39., 103., 308., 116.,
       145., 74., 45., 115., 264., 87., 202., 127., 182., 241., 66.,
        94., 283., 64., 102., 200., 265., 94., 230., 181., 156., 233.,
        60., 219., 80., 68., 332., 248., 84., 200., 55., 85., 89.,
        31., 129., 83., 275., 65., 198., 236., 253., 124., 44., 172.,
       114., 142., 109., 180., 144., 163., 147., 97., 220., 190., 109.,
       191., 122., 230., 242., 248., 249., 192., 131., 237., 78., 135.,
       244., 199., 270., 164., 72., 96., 306., 91., 214., 95., 216.,
       263., 178., 113., 200., 139., 139., 88., 148., 88., 243., 71.,
        77., 109., 272., 60., 54., 221., 90., 311., 281., 182., 321.,
        58., 262., 206., 233., 242., 123., 167., 63., 197., 71., 168.,
       140., 217., 121., 235., 245., 40., 52., 104., 132., 88., 69.,
       219., 72., 201., 110., 51., 277., 63., 118., 69., 273., 258.,
        43., 198., 242., 232., 175., 93., 168., 275., 293., 281., 72.,
       140., 189., 181., 209., 136., 261., 113., 131., 174., 257., 55.,
        84., 42., 146., 212., 233., 91., 111., 152., 120., 67., 310.,
        94., 183., 66., 173., 72., 49., 64., 48., 178., 104., 132.,
       220., 57.]), 'frame': None, 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTen baseline variables, age, sex, body mass index, average blood\npressure, and six blood serum measurements were obtained for each of n =\n442 diabetes patients, as well as the response of interest, a\nquantitative measure of disease progression one year after baseline.\n\n**Data Set Characteristics:**\n\n :Number of Instances: 442\n\n :Number of Attributes: First 10 columns are numeric predictive values\n\n :Target: Column 11 is a quantitative measure of disease progression

```
one year after baseline\n\n  :Attribute Information:\n      - age     age
in years\n      - sex\n      - bmi     body mass index\n      - bp      av
erage blood pressure\n      - s1      tc, total serum cholesterol\n      -
s2      ldl, low-density lipoproteins\n      - s3      hdl, high-density l
ipoproteins\n      - s4      tch, total cholesterol / HDL\n      - s5
ltg, possibly log of serum triglycerides level\n      - s6      glu, blood
sugar level\n\nNote: Each of these 10 feature variables have been mean cen
tered and scaled by the standard deviation times `n_samples` (i.e. the sum
of squares of each column totals 1).\n\nSource URL:\nhttps://www4.stat.ncs
u.edu/~boos/var.select/diabetes.html\n\nFor more information see:\nBradley
Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least A
ngle Regression," Annals of Statistics (with discussion), 407-499.\n(http
s://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)', 'feature_n
ames': ['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6'], 'd
ata_filename': 'diabetes_data.csv.gz', 'target_filename': 'diabetes_targe
t.csv.gz', 'data_module': 'sklearn.datasets.data'}
```

In [37]:
```
from sklearn import datasets
import pandas as pd
b=datasets.load_breast_cancer()
print(b)
```

```
dimensional space is that described in:\n[K. P. Bennett and O. L. Mangas
arian: "Robust Linear\nProgramming Discrimination of Two Linearly Insepa
rable Sets",\nOptimization Methods and Software 1, 1992, 23-34].\n\nThis
database is also available through the UW CS ftp server:\n\nftp ftp.cs.w
isc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. topic:: Refe
rences\n\n   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear f
eature extraction \n     for breast tumor diagnosis. IS&T/SPIE 1993 Inte
rnational Symposium on \n     Electronic Imaging: Science and Technolog
y, volume 1905, pages 861-870,\n     San Jose, CA, 1993.\n   - O.L. Mang
asarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and \n
prognosis via linear programming. Operations Research, 43(4), pages 570-
577, \n     July-August 1995.\n   - W.H. Wolberg, W.N. Street, and O.L.
Mangasarian. Machine learning techniques\n     to diagnose breast cancer
from fine-needle aspirates. Cancer Letters 77 (1994) \n     163-171.',
'feature_names': array(['mean radius', 'mean texture', 'mean perimeter',
'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
```

In [38]:
```
print(type(d))
```

```
<class 'sklearn.utils.Bunch'>
```

In [39]:
```
print(d.keys())
```

```
dict_keys(['data', 'target', 'frame', 'DESCR', 'feature_names', 'data_file
name', 'target_filename', 'data_module'])
```

In [42]:
```
print(type(object))
```

```
<class 'type'>
```

In [47]:
```
print(type(d.data))
```

```
<class 'numpy.ndarray'>
```

In [48]:
```
print(type(d.target))
```

```
<class 'numpy.ndarray'>
```

In [49]:
```
print(d.data.shape)
```

```
(442, 10)
```

In [51]:
```
x=d.data
y=d.target
print(x)
print(y)
```

```
[[ 0.03807591  0.05068012  0.06169621 ... -0.00259226  0.01990842
  -0.01764613]
 [-0.00188202 -0.04464164 -0.05147406 ... -0.03949338 -0.06832974
  -0.09220405]
 [ 0.08529891  0.05068012  0.04445121 ... -0.00259226  0.00286377
  -0.02593034]
 ...
 [ 0.04170844  0.05068012 -0.01590626 ... -0.01107952 -0.04687948
   0.01549073]
 [-0.04547248 -0.04464164  0.03906215 ...  0.02655962  0.04452837
  -0.02593034]
 [-0.04547248 -0.04464164 -0.0730303  ... -0.03949338 -0.00421986
   0.00306441]]
[151.  75. 141. 206. 135.  97. 138.  63. 110. 310. 101.  69. 179. 185.
 118. 171. 166. 144.  97. 168.  68.  49.  68. 245. 184. 202. 137.  85.
 131. 283. 129.  59. 341.  87.  65. 102. 265. 276. 252.  90. 100.  55.
  61.  92. 259.  53. 190. 142.  75. 142. 155. 225.  59. 104. 182. 128.
  52.  37. 170. 170.  61. 144.  52. 128.  71. 163. 150.  97. 160. 178.
  48. 270. 202. 111.  85.  42. 170. 200. 252. 113. 143.  51.  52. 210.
  65. 141.  55. 134.  42. 111.  98. 164.  48.  96.  90. 162. 150. 279.
  92.  83. 128. 102. 302. 198.  95.  53. 134. 144. 232.  81. 104.  59.
 246. 297. 258. 229. 275. 281. 179. 200. 200. 173. 180.  84. 121. 161.
  99. 109. 115. 268. 274. 158. 107.  83. 103. 272.  85. 280. 336. 281.
 118. 317. 235.  60. 174. 259. 178. 128.  96. 126. 288.  88. 292.  71.
 197. 186.  25.  84.  96. 195.  53. 217. 172. 131. 214.  59.  70. 220.
 268. 152.  47.  74. 295. 101. 151. 127. 237. 225.  81. 151. 107.  64.
 138. 185. 265. 101. 137. 143. 141.  79. 292. 178.  91. 116.  86. 122.
  72. 129. 142.  90. 158.  39. 196. 222. 277.  99. 196. 202. 155.  77.
 191.  70.  73.  49.  65. 263. 248. 296. 214. 185.  78.  93. 252. 150.
  77. 208.  77. 108. 160.  53. 220. 154. 259.  90. 246. 124.  67.  72.
 257. 262. 275. 177.  71.  47. 187. 125.  78.  51. 258. 215. 303. 243.
  91. 150. 310. 153. 346.  63.  89.  50.  39. 103. 308. 116. 145.  74.
  45. 115. 264.  87. 202. 127. 182. 241.  66.  94. 283.  64. 102. 200.
 265.  94. 230. 181. 156. 233.  60. 219.  80.  68. 332. 248.  84. 200.
  55.  85.  89.  31. 129.  83. 275.  65. 198. 236. 253. 124.  44. 172.
 114. 142. 109. 180. 144. 163. 147.  97. 220. 190. 109. 191. 122. 230.
 242. 248. 249. 192. 131. 237.  78. 135. 244. 199. 270. 164.  72.  96.
 306.  91. 214.  95. 216. 263. 178. 113. 200. 139. 139.  88. 148.  88.
 243.  71.  77. 109. 272.  60.  54. 221.  90. 311. 281. 182. 321.  58.
 262. 206. 233. 242. 123. 167.  63. 197.  71. 168. 140. 217. 121. 235.
 245.  40.  52. 104. 132.  88.  69. 219.  72. 201. 110.  51. 277.  63.
 118.  69. 273. 258.  43. 198. 242. 232. 175.  93. 168. 275. 293. 281.
  72. 140. 189. 181. 209. 136. 261. 113. 131. 174. 257.  55.  84.  42.
 146. 212. 233.  91. 111. 152. 120.  67. 310.  94. 183.  66. 173.  72.
  49.  64.  48. 178. 104. 132. 220.  57.]
```

In [ ]:
```
```