In [19]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [18]:
```
#reading data from the given url
url='http://bit.ly/w-data'
b_data=pd.read_csv(url)
b_data.head(10)
```

Out[18]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25

In [20]:
```
#getting the shape of the data
b_data.shape
```

Out[20]:
```
(25, 2)
```

In [21]:
```
# check for the descrpition of the data
b_data.describe()
```

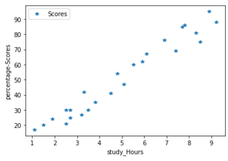
Out[21]:

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [22]:
```
# check for the information about the given data
b_data.info
```

Out[22]:
```
<bound method DataFrame.info of     Hours  Scores
0     2.5      21
1     5.1      47
2     3.2      27
3     8.5      75
4     3.5      30
5     1.5      20
6     9.2      88
7     5.5      60
8     8.3      81
9     2.7      25
10    7.7      85
11    5.9      62
12    4.5      41
13    3.3      42
14    1.1      17
15    8.9      95
16    2.5      30
17    1.9      24
18    6.1      67
19    7.4      69
20    2.7      30
21    4.8      54
22    3.8      35
23    6.9      76
24    7.8      86>
```

In [3]:
```
#plotting the data scores vs study hours
b_data.plot(x='Hours',y='Scores',style='*')
plt.xlabel('study_Hours')
plt.ylabel('percentage-Scores')
plt.show()
```



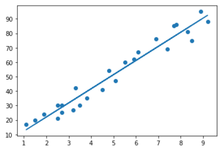
In [4]:
```
#prepare the data
x=b_data.iloc[:,:-1].values
y=b_data.iloc[:,1].values
```

In [24]:
```
#split the data into test data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state=0)
```

In [25]:
```
#Training the algorithm for the model
 from sklearn.linear_model import LinearRegression
regressor1=LinearRegression()
regressor1.fit(x,y)
print("TRAINING COMPLETED")
```

```
TRAINING COMPLETED
```

In [6]:
```
#plotting the regression line
line=regressor1.coef_*x+regressor1.intercept_
plt.scatter(x,y)
plt.plot(x,line)
plt.show()
```



In [7]:
```
#testing data
print(x_test)
#predicting the scores
y_pred=regressor1.predict(x_test)
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
```

In [26]:
```
y_pred
```

Out[26]:
```
array([17.14737849, 33.76624426, 74.8246185 , 26.92318188, 60.16091341])
```

In [8]:
```
# Comparing actual and predicted data
df=pd.DataFrame({'actual':y_test , 'predicted':y_pred})
df
```

Out[8]:

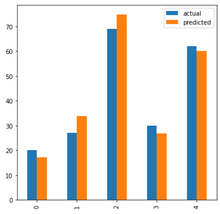
	actual	predicted
0	20	17.147378
1	27	33.766244
2	69	74.824618
3	30	26.923182
4	62	60.160913

In [9]:
```
print('training score:', regressor1.score(x_train,y_train))
print('testing score:', regressor1.score(x_test,y_test))
```

```
training score: 0.9512837351709387
testing score: 0.9491748734859172
```

In [14]:
```
df.plot(kind='bar', figsize=(6,6))
```

Out[14]:
```
<AxesSubplot:>
```



In [16]:
```
# testing the own data with the predicted data
hours=9.25
test=np.array([hours])
test=test.reshape(-1,1)
own_pred= regressor1.predict(test)
print('no of hours={}'.format(hours))
print('predicted scores={}'.format(own_pred[0]))
```

```
no of hours=9.25
predicted scores=92.9098547701573
```

In [17]:
```
#checking the efficiency of the model
import numpy as np
from sklearn import metrics
print('mean absolute error:',metrics.mean_absolute_error(y_test,y_pred))
print('mean squared error:',metrics.mean_squared_error(y_test,y_pred))
print('root mean squared error:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
print('explained variance score:',metrics.explained_variance_score(y_test,y_pred))
```

```
mean absolute error: 4.071877793635605
mean squared error: 20.138948129940175
root mean squared error: 4.487643939746131
explained variance score: 0.9515224335188082
```