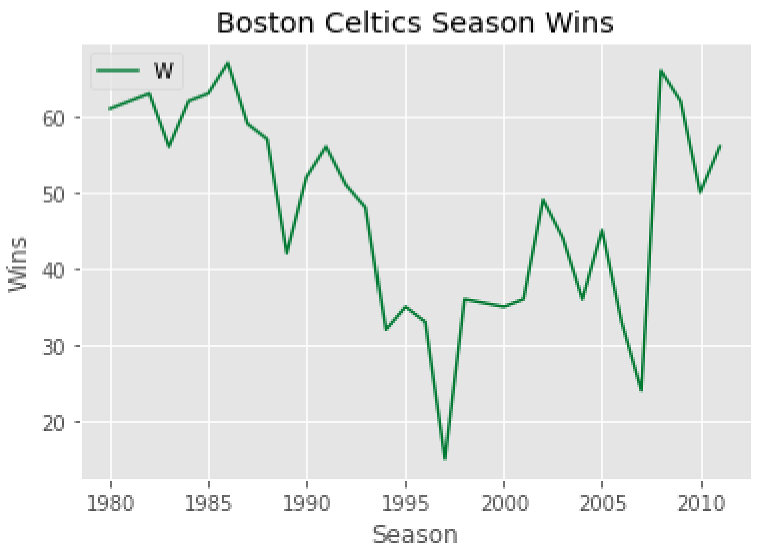
Machine Learning in Python (Linear Regression and Classifier Algorithms)

Keith Letourneau

Northeastern Univ.

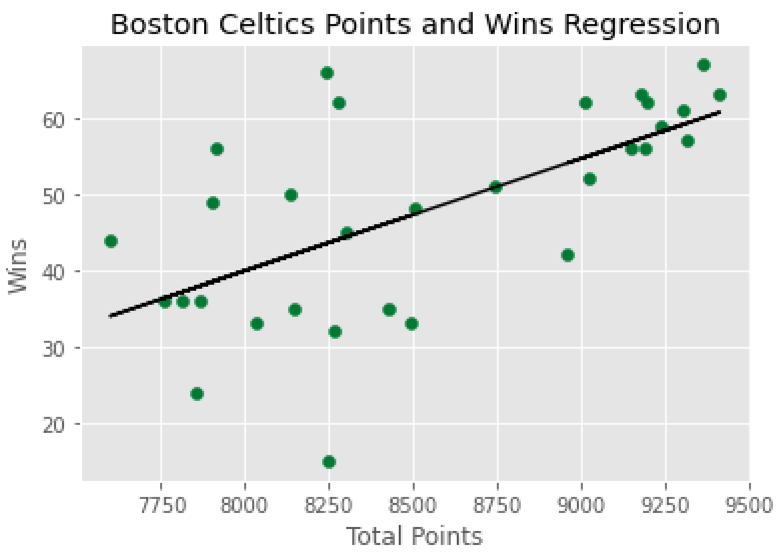
In [3]:
```
#import libraries
import pandas as pd
from pandas import DataFrame
import matplotlib.pyplot as plt
import numpy as np
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
plt.style.use('ggplot')
```

In [32]:
```
#load data for NBA
nba = pd.read_csv(r'C:\Users\keith\Downloads\NBA.csv')

bos = DataFrame(nba[nba['Team'] == 'Boston Celtics'])
bos.head()

bos.plot(x='SeasonEnd', y='W', kind='line', color='#007A33')
plt.ylabel('Wins')
plt.xlabel('Season')
plt.title('Boston Celtics Season Wins')
plt.show()
```



In [33]:
```
#linear regression model
celtics = DataFrame(bos, columns=['PTS','W'])
lm = LinearRegression()

x = celtics['PTS'].values.reshape(-1,1)
y = celtics['W'].values.reshape(-1,1)
```

```
fit = lm.fit(x,y)

prediction = lm.predict(x)

plt.scatter(x, y, color='#007A33')
plt.plot(x, prediction, color='black')
plt.ylabel('Wins')
plt.xlabel('Total Points')
plt.title('Boston Celtics Points and Wins Regression')
plt.show()

x_new = [[8523]]
print('Predicted amount of wins for the season with',
      x_new, 'points:', fit.predict(x_new))
```



```
Predicted amount of wins for the season with [[8523]] points: [[47.63110388]]
```

In [19]:
```
#logistic regression on whether or not 50 games will be won
#create binary outcome column for classifier
bos['50 wins'] = ['Yes' if i > 50
             else 'No' for i in bos['W']]

x = bos[['PTS', 'AST', '3P', 'oppPTS']]
y = bos['50 wins']

x_train, x_test, y_train, y_test = train_test_split(x2, y2, random_state=4)

glm = LogisticRegression()

glm.fit(x_train, y_train)

prediction = glm.predict(x_test)

accuracy = metrics.accuracy_score(y_test, prediction)
accuracy_percentage = 100 * accuracy
print('Model accuracy is:', accuracy_percentage)

season_x = glm.predict((np.array([8669, 1753, 589, 7914]).reshape(1, -1)))

print('Will we win atleast 50 games this season?',
      season_x)
```

```
Model accuracy is: 100.0
```

```
Will we win atleast 50 games this season? ['Yes']
```

Look into why model accuracy is 100. Data is broken up into train and test so why not sure what is going on there.

Bootstrap Aggregating

Practice building ensemble model ultilizing bagging method:

- Create random samples of the training data set (sub sets of training data set)
- Build a model for each sample
- Results of these multiple models are combined using average or majority voting

In [20]:

```
#load data
data = pd.read_csv(r'C:\Users\keith\Downloads\programmers.csv')

df = DataFrame(data, columns=['experience', 'score', 'salary'])

print(df.head())

df['score'].mean()
```

```
   experience  score  salary
0           4     78    48.0
1           7    100    86.0
2           1     86    47.4
3           5     82    68.6
4          10     84    76.0
```

Out[20]:

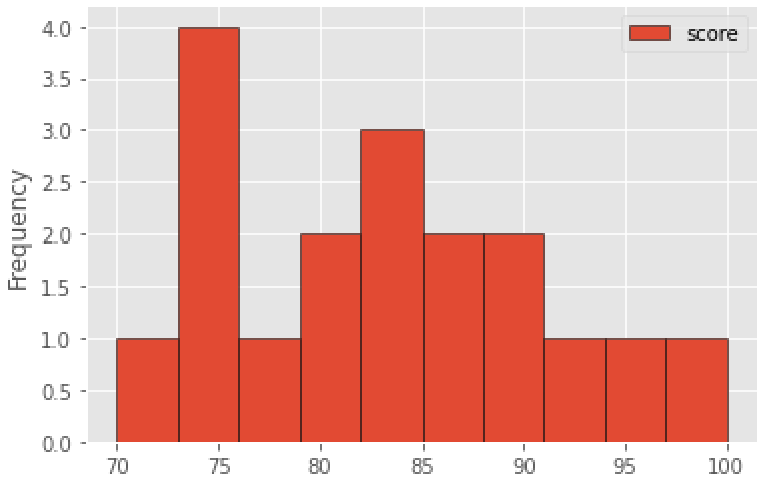
```
82.66666666666667
```

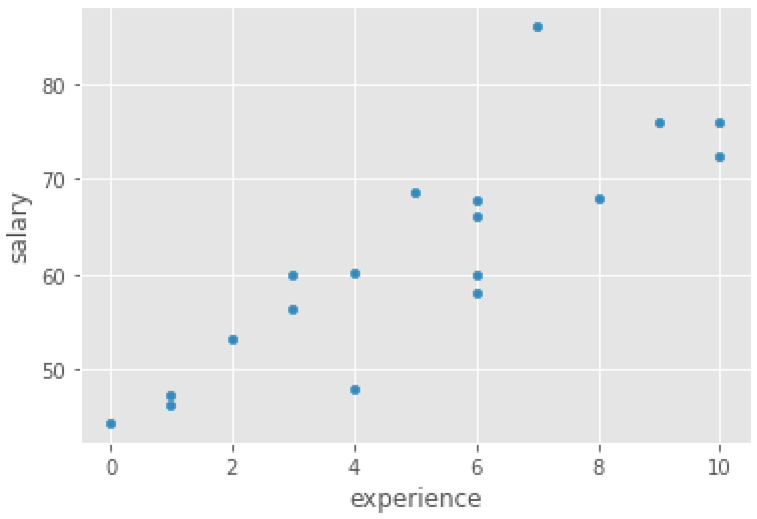
In [21]:

```
#histogram of salaries
h = DataFrame(data, columns=['score'])

h.plot.hist(edgecolor='black')

df.plot(y='salary', x='experience', kind='scatter')
```

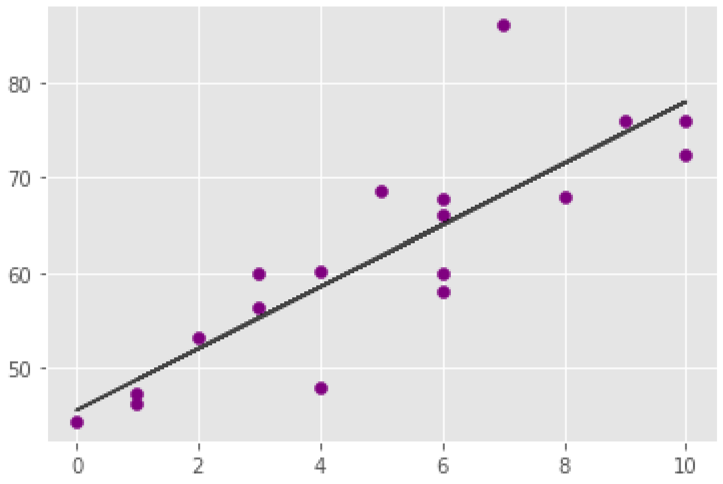
Out[21]:

```
<AxesSubplot:xlabel='experience', ylabel='salary'>
```





In [30]:
```
#build regession model
r = DataFrame(data, columns=['experience', 'salary'])
x = r['experience'].values.reshape(-1,1)
y = r['salary'].values.reshape(-1,1)

fit = lm.fit(x, y)
prediction = lm.predict(x)

plt.scatter(x, y, color='purple')
plt.plot(x, prediction, color='black', alpha=.7)
plt.show()

x_new = [[2.2]]
print('The predicted salary is:',
      fit.predict(x_new))
```

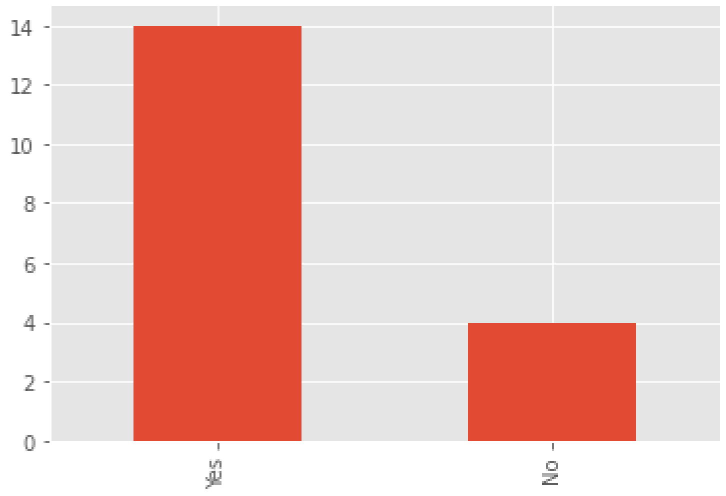


```
The predicted salary is: [[52.67205872]]
```

In [39]:
```
#add new binary column for logistic regression
#predict whether or not a job will be secured
#classifer decided randomly off of salary threshold
df['job'] = ['Yes' if i > 50
             else 'No' for i in df['salary']]

print(df.head())

df['job'].value_counts().plot(kind='bar')
```

```
df.head()

x = df.drop('job', axis = 1)
y = df['job']

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=4)

glm.fit(x_train, y_train)

app_one = glm.predict((np.array([4, 78, 52]).reshape(1, -1)))

print('Will he/she get the job?',
      app_one)
```

```
   experience  score  salary  job
0           4     78    48.0   No
1           7    100    86.0  Yes
2           1     86    47.4   No
3           5     82    68.6  Yes
4          10     84    76.0  Yes
Will he/she get the job? ['Yes']
```


Yes
No