In [27]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

In [21]:
```
# load in iris data
iris = load_iris()

# Create feature (X) and target variable (y)
X = iris.data
y = iris.target
```

In [22]:
```
print(iris.data.shape)
print(iris.target.shape)
```

```
(150, 4)
(150,)
```

In [25]:
```
# Using train/test split with random state
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 2)

# Calculating classification accuracy of KNN with K = 5)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_test, y_test)
y_pred = knn.predict(X_test)
print(metrics.accuracy_score(y_test,y_pred))
```

```
0.9210526315789473
```

In [28]:
```
from sklearn.cross_validation import cross_val_score
```

In [29]:
```
#10-fold cross-validation with K=5 for KNN
knn = KNeighborsClassifier(n_neighbors=5)
scores = cross_val_score(knn, X, y, cv = 10, scoring = 'accuracy')
print(scores)
```

```
[1.         0.93333333 1.         1.         0.86666667 0.93333333
 0.93333333 1.         1.         1.        ]
```

In [30]:
```
# using average of accuracy scores for estimate of out-of-sample accuracy
print(scores.mean())
```

```
0.9666666666666668
```

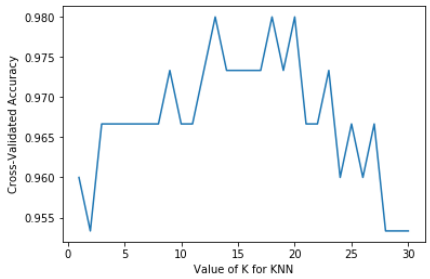
In [34]:
```
# for loop to identify optimal value of k for KNN
k_range = range(1,31) #range of 1 through 30
k_scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X, y, cv = 10, scoring = 'accuracy')
    k_scores.append(scores.mean())
print (k_scores)
```

```
[0.96, 0.9533333333333334, 0.9666666666666666, 0.9666666666666666, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668, 0.9666666666666668,
0.9733333333333334, 0.9666666666666668, 0.9666666666666668, 0.9733333333333334, 0.9800000000000001, 0.9733333333333334, 0.9733333333333334, 0.9733
333333333334, 0.9733333333333334, 0.9800000000000001, 0.9733333333333334, 0.9800000000000001, 0.9666666666666666, 0.9666666666666666, 0.9733333333
333334, 0.96, 0.9666666666666666, 0.96, 0.9666666666666666, 0.9533333333333334, 0.9533333333333334, 0.9533333333333334]
```

In [35]:
```
import matplotlib.pyplot as plt
%matplotlib inline

#visualization of how accuracy changes with different K values
plt.plot(k_range,k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
```

Out[35]:
```
Text(0,0.5,'Cross-Validated Accuracy')
```



In [38]:
```
#CROSS VALIDATION FOR MODEL SELECTION BETWEEN KNN & LOGISTIC REGRESSION

# 10-fold CV with best KNN model from above visualization
knn = KNeighborsClassifier(n_neighbors = 20)
print(cross_val_score(knn, X, y, cv=10, scoring='accuracy').mean())
```

```
0.9800000000000001
```

In [39]:
```
# 10-fold CV with logisitic regression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
print (cross_val_score(logreg, X, y, cv=10, scoring='accuracy').mean())
```

```
0.9533333333333334
```

In [ ]:
```
# One would conclude the KNN model would perform better than LR model.
```