In [13]:
```
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
from math import sqrt
import warnings
warnings.filterwarnings('ignore')
```

In [23]:
```
df = pd.read_csv('../Country Quater Wise Visitors Imputed.csv')
df.head()
```

Out[23]:

	Country of Nationality	2014 1st quarter (Jan-March)	2014 2nd quarter (Apr-June)	2014 3rd quarter (July-Sep)	2014 4th quarter (Oct-Dec))	2015 1st quarter (Jan-March)	2015 2nd quarter (Apr-June)	2015 3rd quarter (July-Sep)	2015 4th quarter (Oct-Dec)	2016 1st quarter (Jan-March)	...
0	Canada	33.1	14.5	15.7	36.7	34.2	14.5	15.9	35.4	34.5	...
1	United States Of America	25.7	22.0	20.6	31.7	26.4	22.1	19.6	31.9	25.7	...
2	Argentina	46.8	15.6	13.9	23.7	40.6	17.5	14.6	27.3	41.7	...
3	Brazil	31.0	18.8	18.7	31.5	34.8	19.0	19.6	26.6	29.7	...
4	Mexico	23.6	20.3	26.5	29.6	28.1	21.3	23.4	27.2	25.4	...

5 rows × 29 columns

In [15]:
```
def parser(x):
    return datetime.strptime(x, '%Y-%m')

def getDate(x):
    d = x.split(' ')[0: 2]
    d[1] = str(int(d[1][0])*3)
    return parser("-".join(d))
```

In [16]:
```
def generateDate():
    x = []
    for series_name, series in df.items():
        if series_name == 'Country of Nationality': continue
        x.append(getDate(series_name))
    return x
```

In [17]:
```
new_df = pd.DataFrame()
new_df['time'] = generateDate()
for series_name, series in df.T.items():
    if series_name == 1:
        new_df['value'] = list(series)[1:]
```

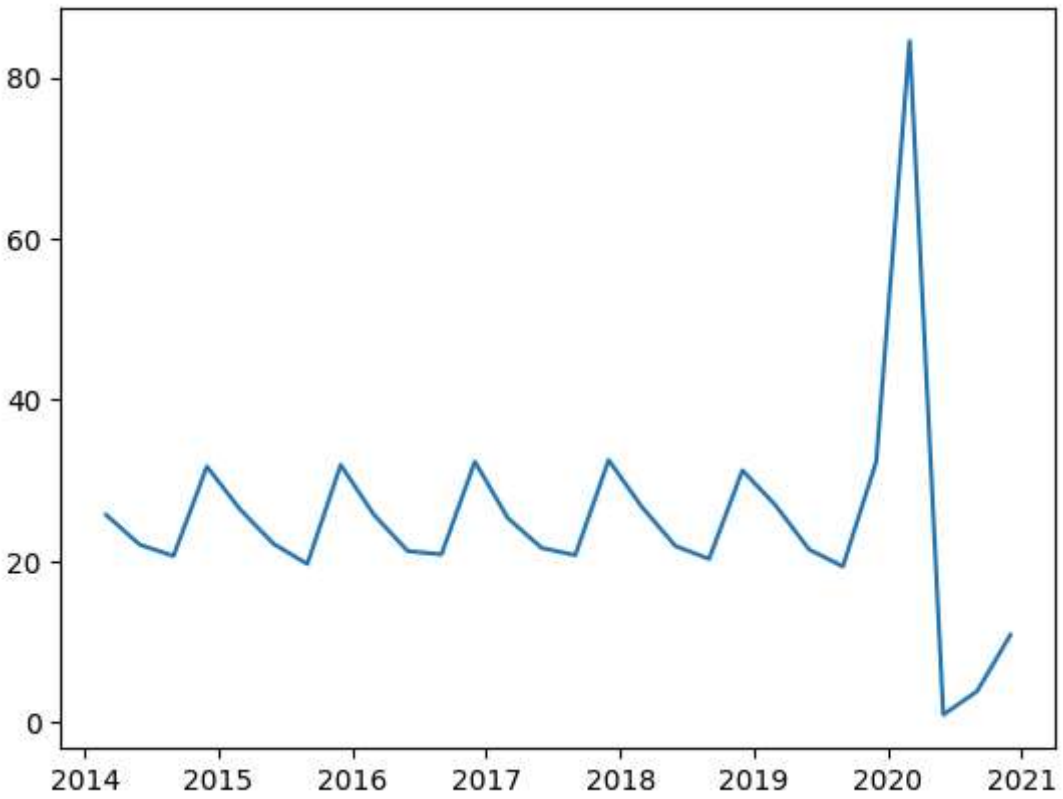
In [18]:
```
new_df.to_csv('data.csv', index=False)
```

In [19]:
```
df2 = pd.read_csv('data.csv', parse_dates=['time'], header=0).squeeze('columns
df3 = pd.read_csv('data.csv', parse_dates=['time'], header=0, index_col='time
# df2['hello'] = df2.index
df2.head()
```

Out[19]:

	time	value
0	2014-03-01	25.7
1	2014-06-01	22.0
2	2014-09-01	20.6
3	2014-12-01	31.7
4	2015-03-01	26.4

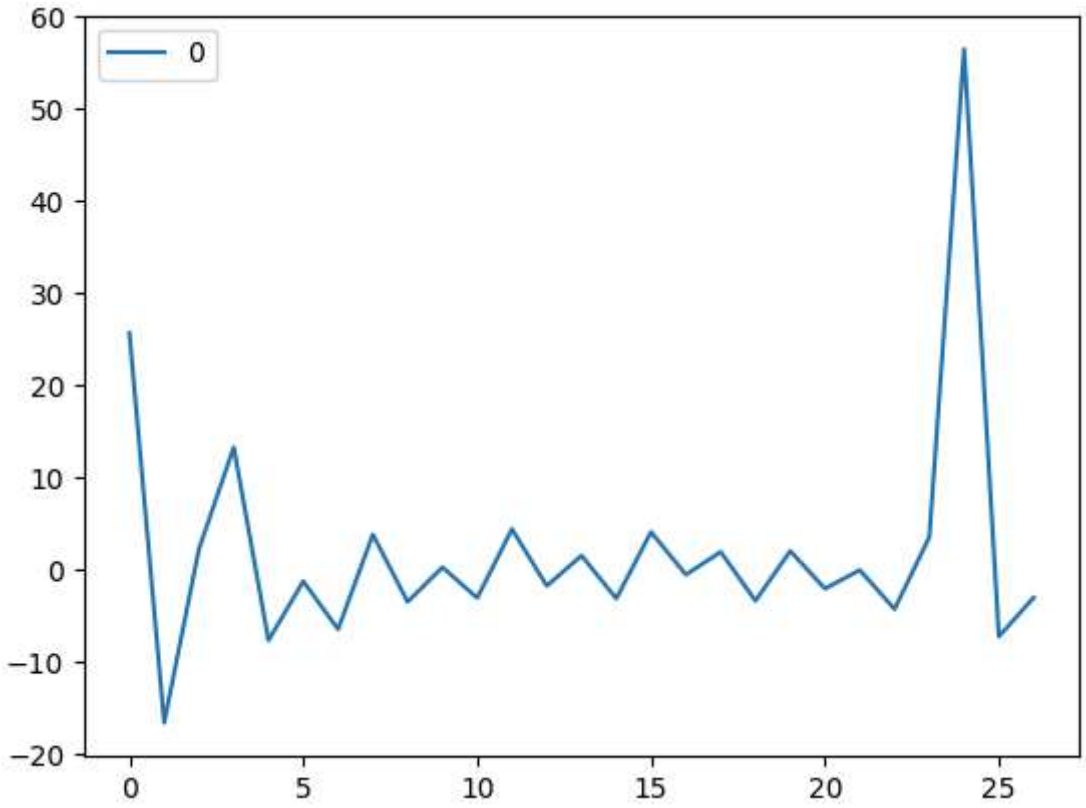
In [20]:
```
plt.plot(df2['time'], df2['value'])
plt.show()
```

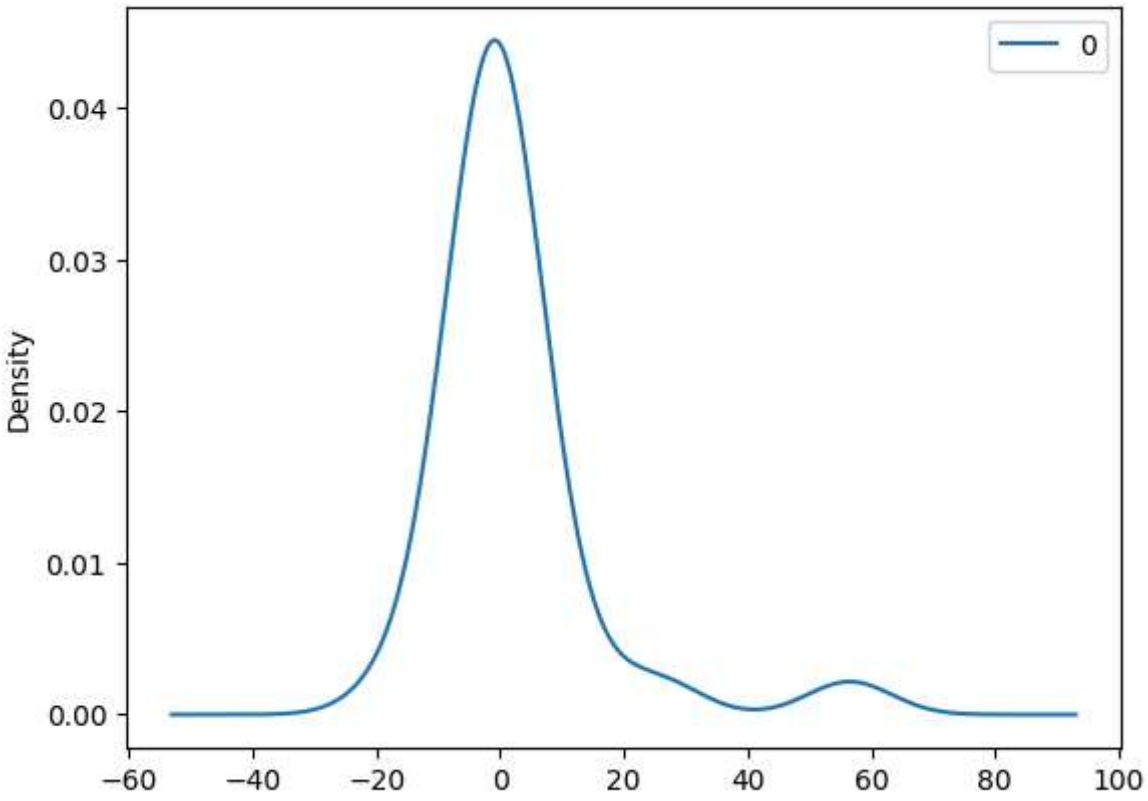


In [26]:
```
X = df3.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()

# walk-forward validation
for t in range(len(test)):
    model = ARIMA(history, order=(5,2,2))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))

# summary of fit model
print(model_fit.summary())
# line plot of residuals
residuals = pd.DataFrame(model_fit.resid)
residuals.plot()
plt.show()
# density plot of residuals
residuals.plot(kind='kde')
plt.show()
# summary stats of residuals
print(residuals.describe())

# evaluate forecasts
rmse = sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)
```

```
predicted=21.027372, expected=20.230000
predicted=32.260418, expected=31.220000
predicted=26.616906, expected=26.900000
predicted=22.125562, expected=21.400000
predicted=20.352302, expected=19.300000
predicted=31.102871, expected=32.300000
predicted=27.407853, expected=84.500000
predicted=150.109124, expected=0.900000
predicted=34.886835, expected=3.800000
predicted=59.706075, expected=10.800000
                               SARIMAX Results
==========================================================================
==
Dep. Variable:                        y   No. Observations:
27
Model:                 ARIMA(5, 2, 2)   Log Likelihood                -101.6
93
Date:                Wed, 10 Apr 2024   AIC                            219.3
87
Time:                        00:21:05   BIC                            229.1
38
Sample:                             0   HQIC                           222.0
91
                                 - 27
Covariance Type:                  opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
ar.L1         -0.5921     12.832     -0.046      0.963     -25.743      24.5
59
ar.L2         -0.8947     15.084     -0.059      0.953     -30.460      28.6
70
ar.L3         -0.4140     19.447     -0.021      0.983     -38.529      37.7
02
ar.L4         -0.0856     14.033     -0.006      0.995     -27.589      27.4
18
ar.L5          0.1259      8.279      0.015      0.988     -16.101      16.3
53
ma.L1         -1.6134     12.743     -0.127      0.899     -26.590      23.3
63
ma.L2          0.6947     13.330      0.052      0.958     -25.432      26.8
21
sigma2       137.3339     90.471      1.518      0.129     -39.986     314.6
54
==========================================================================
=======
Ljung-Box (L1) (Q):                   0.34   Jarque-Bera (JB):
371.59
Prob(Q):                              0.56   Prob(JB):
0.00
Heteroskedasticity (H):              47.01   Skew:
4.17
Prob(H) (two-sided):                  0.00   Kurtosis:
19.95
```

```
==============================================================================
=======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (compl
ex-step).
```





```
                0
count  27.000000
mean    2.087785
std    13.146858
min   -16.552640
25%    -3.207204
50%    -0.497889
75%     2.926790
max    56.493325
Test RMSE: 53.746
```

In [25]:
```
plt.plot(df2['time'], X, label='Actual')
plt.plot(df2['time'][size:len(X)], predictions, 'crimson', label='Predicted')
plt.xticks(rotation=45)
plt.xlabel('Year')
plt.ylabel('Values')
plt.legend()
plt.show()
```