Global temperature

This is a time series analysis project.

Here we will import monthly Global temperature data since 1880.

Then we will plot autocorrelation function.

Using Augumented Dickey Fuller Test (ADF test) we will check if time series is stationary or not.

If time series is not stationary, we will calculate its first and second difference and their ACF.

Then we will try to fit some AR(1) model, ARMA(1,1) models.

Finally, we will check its diagnostics.

Also we will try to forcast future values using best model.

In [44]:
```
To start this project we will install some packages first
```

```
  File "/tmp/ipykernel_26449/2744402966.py", line 1
    To start this project we will install some packages first
       ^
SyntaxError: invalid syntax
```

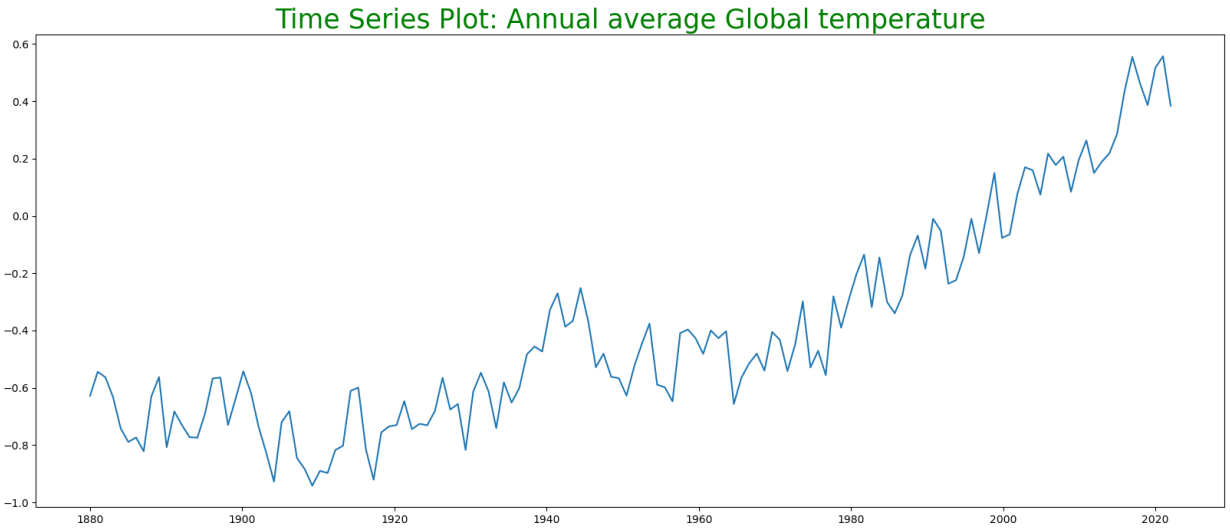
In [45]:
```
from IPython.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.graphics import tsaplots
import matplotlib.pyplot as plt
```

In [46]:
```
data = pd.read_csv(r'/home/gauree/Downloads/TEMP - 1.csv')
#print(data)
```

In [47]:
```
x=data['Year'][0:1704]
#print(x)
temp=data['Anomaly'][0:1704]
#print(temp)
```

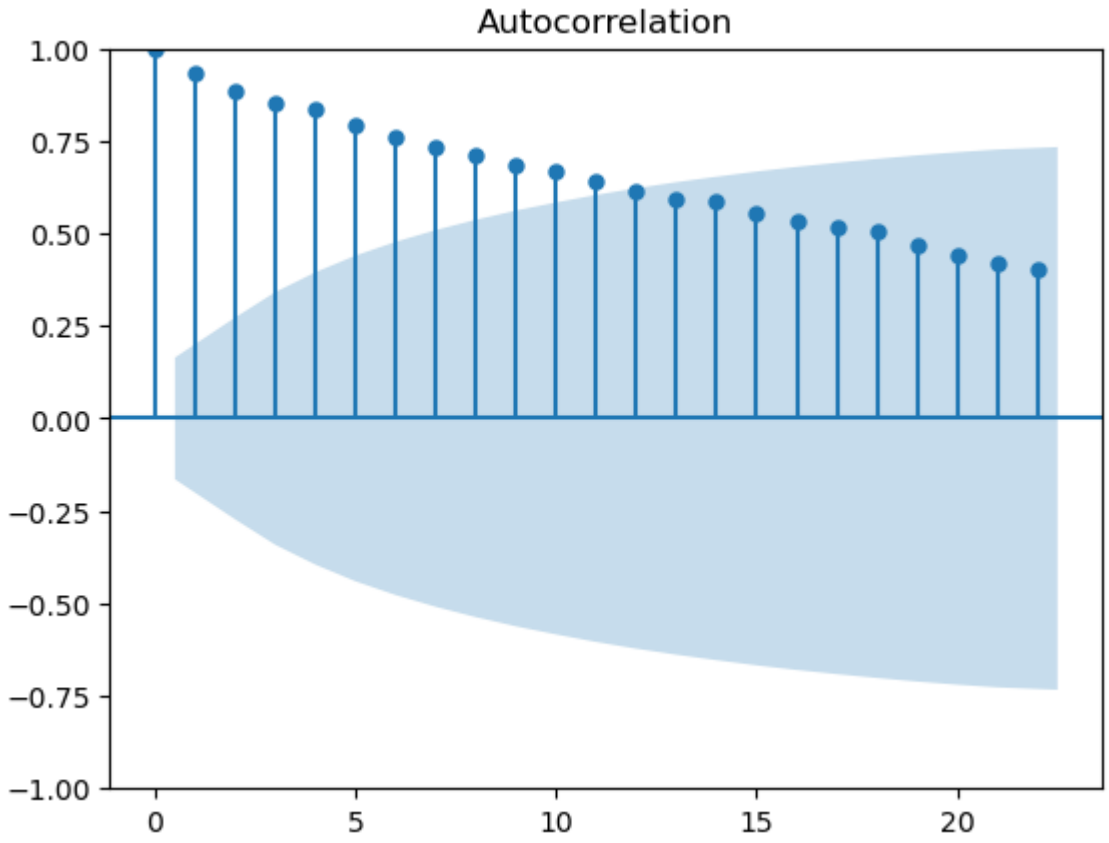
In [48]:
```
interval=12
temp=np.array(temp)
temp_avg=temp.reshape(int(len(temp)/interval),interval)
temp_avg=np.mean(temp_avg,axis=1)
#print(temp_avg)
```

In [49]:
```
x=np.linspace(1880,2022,142)
f=plt.figure(figsize=(20,8))
plt.plot(x,temp_avg)

plt.title("Time Series Plot: Annual average Global temperature",fontsize=25,
          color="green")
plt.show()
```



We will draw the ACF plot of the time series.

In [50]:
```
#plot autocorrelation function
fig = tsaplots.plot_acf(temp_avg)
plt.show()
```

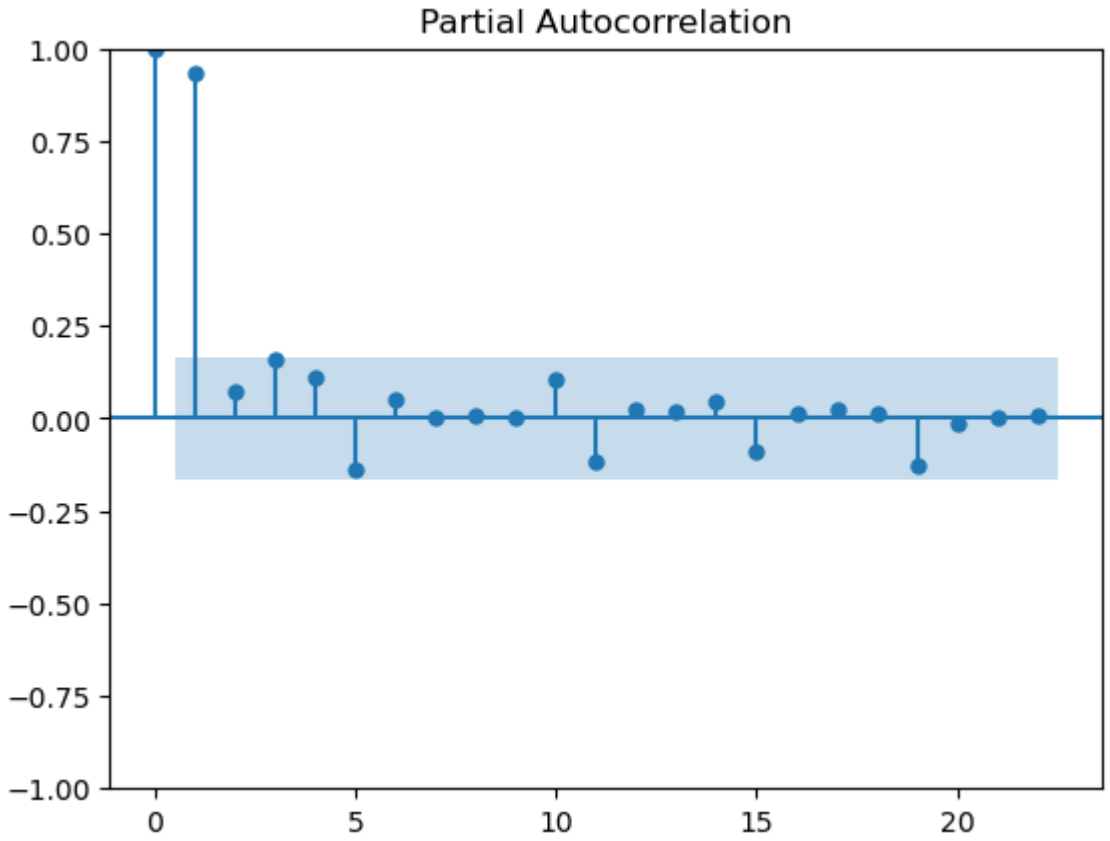


the autocorrelations decay slowly. This also confirms that, the data is not stationary as the the

spikes of the ACF plot are over the threshold region shown in blue.

In [ ]:

In [51]:
```
#plot PACT partial autocorrelation function
fig = tsaplots.plot_pacf(temp_avg, method="ywm")
plt.show()
```



To check if time series is stationary we will use ADF Test.

In [52]:
```
# ADF Test
series = temp_avg
result = adfuller(series, autolag='AIC')
print(f'ADF Statistic: {result[0]}')
print(f'n_lags: {result[1]}')
print(f'p-value: {result[1]}')
for key, value in result[4].items():
    print('Critial Values:')
    print(f'   {key}, {value}')
```

```
ADF Statistic: 1.0582478050392963
n_lags: 0.9948440491482736
p-value: 0.9948440491482736
Critial Values:
   1%, -3.4793722137854926
Critial Values:
   5%, -2.8830370378332995
Critial Values:
   10%, -2.578233635380623
```

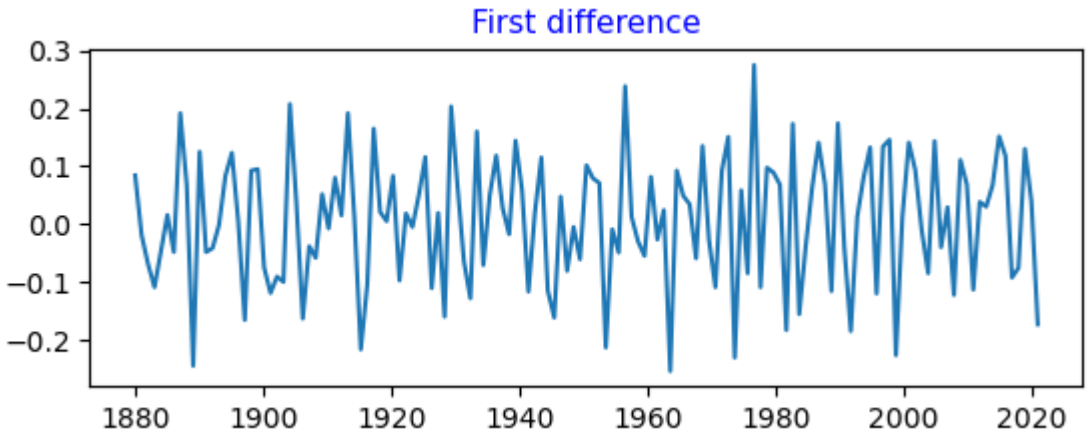
The p-value is obtained is greater than significance level of 0.05 and the ADF statistic is higher than any of the critical values.

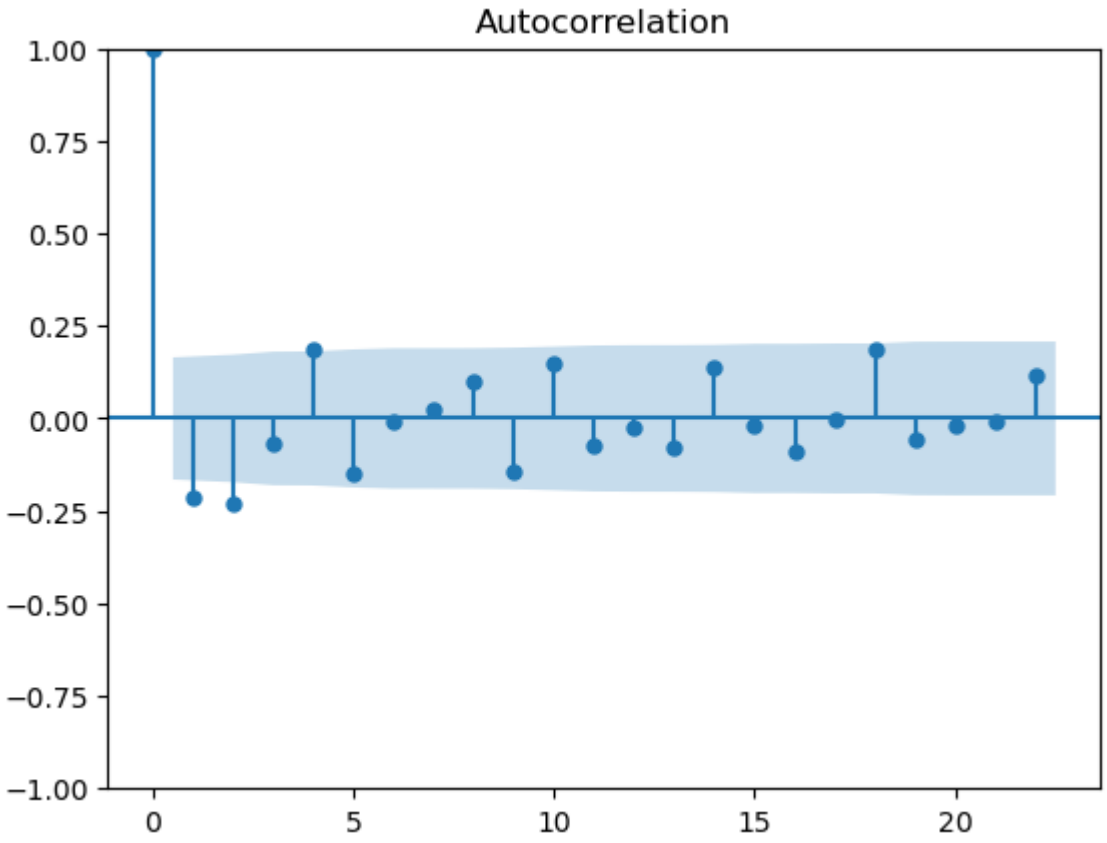
Clearly, there is no reason to reject the null hypothesis. So, the time series is in fact non-stationary.

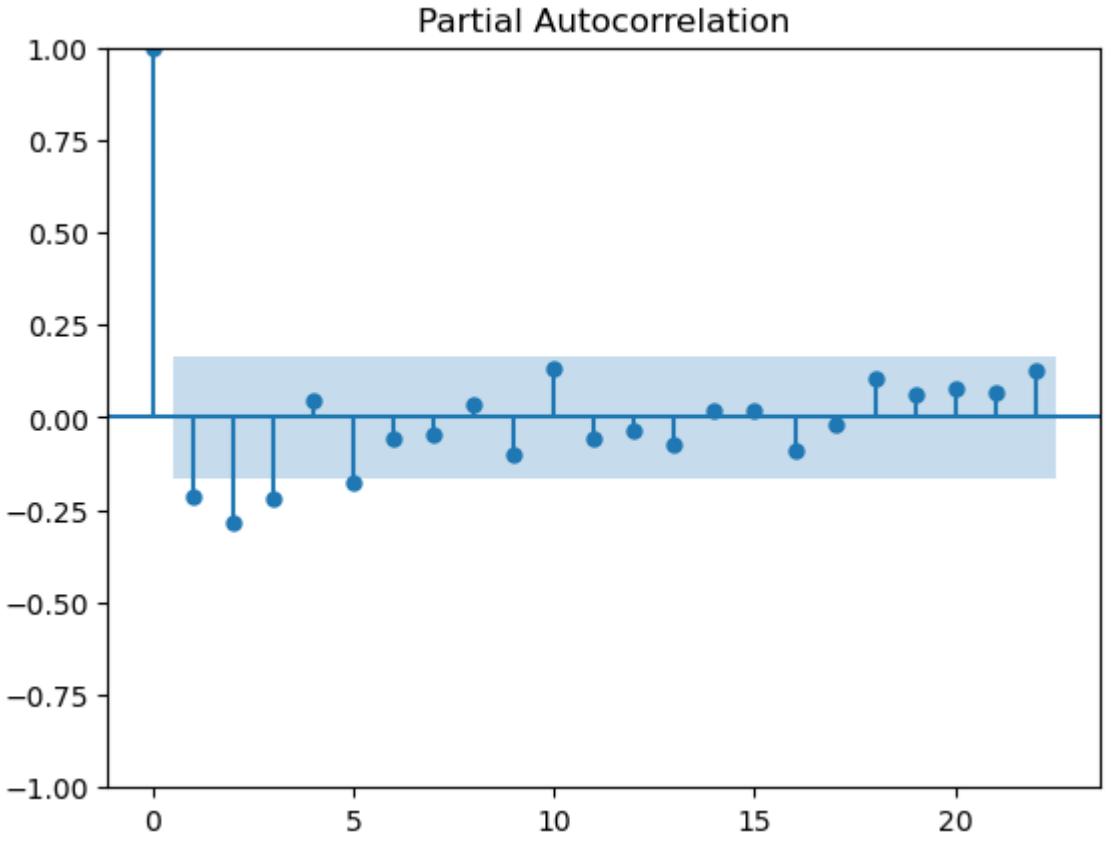
In [ ]:

We will plot the first and second differences of the series.

In [53]:
```
original = temp_avg
first_diff = np.diff(original)
second_diff = np.diff(np.diff(original))


y=x[:-1]
plt.subplot(2, 1, 1)
plt.plot(y,first_diff)
plt.title("First difference",fontsize=11,
          color="blue")


#plt.subplot(2, 3, 4)
plot_acf(first_diff)
plt.show()
#fig = tsaplots.plot_acf(first_diff)
#plt.show()

plot_pacf(first_diff,method="ywm")
plt.show()
```

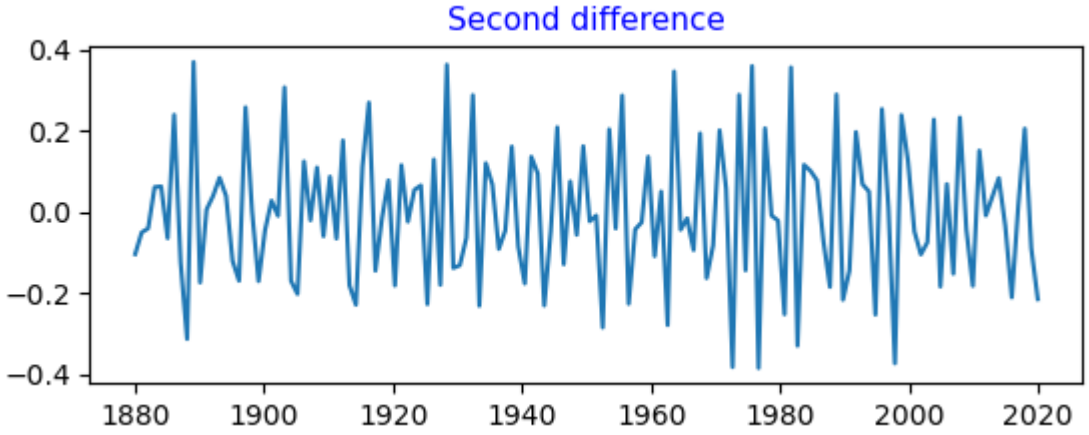


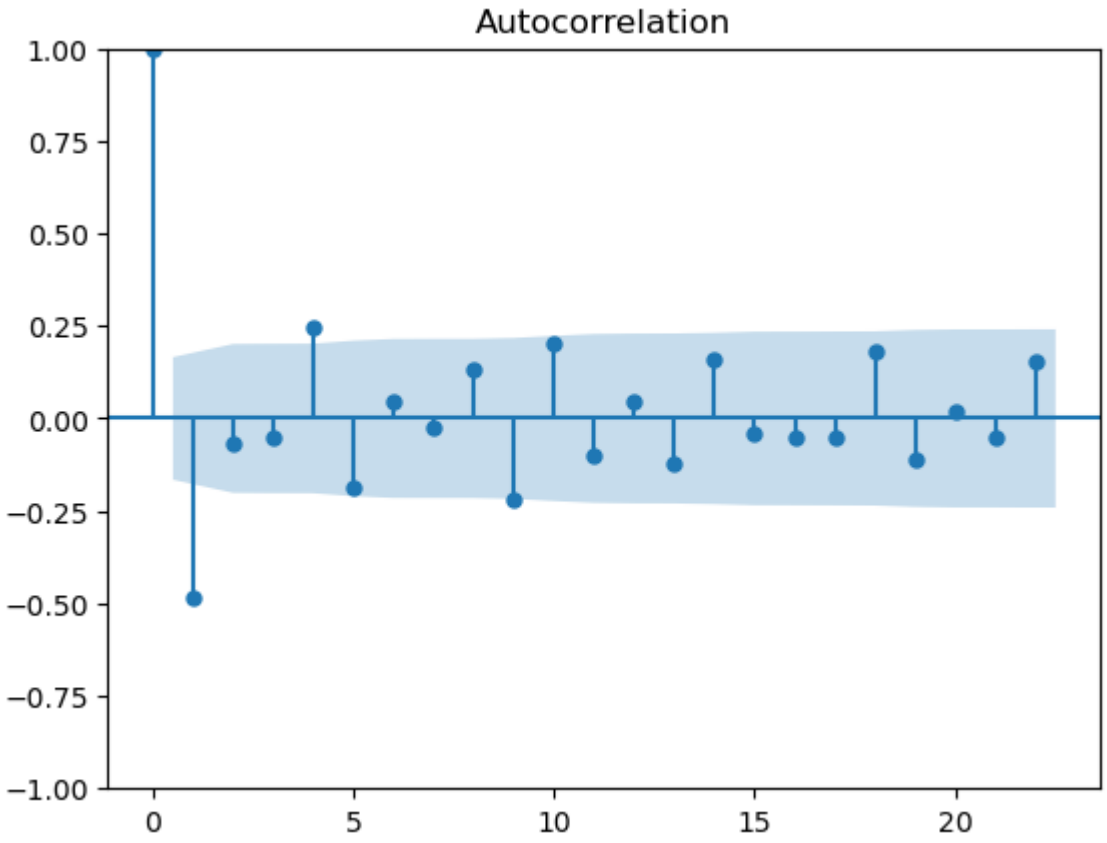


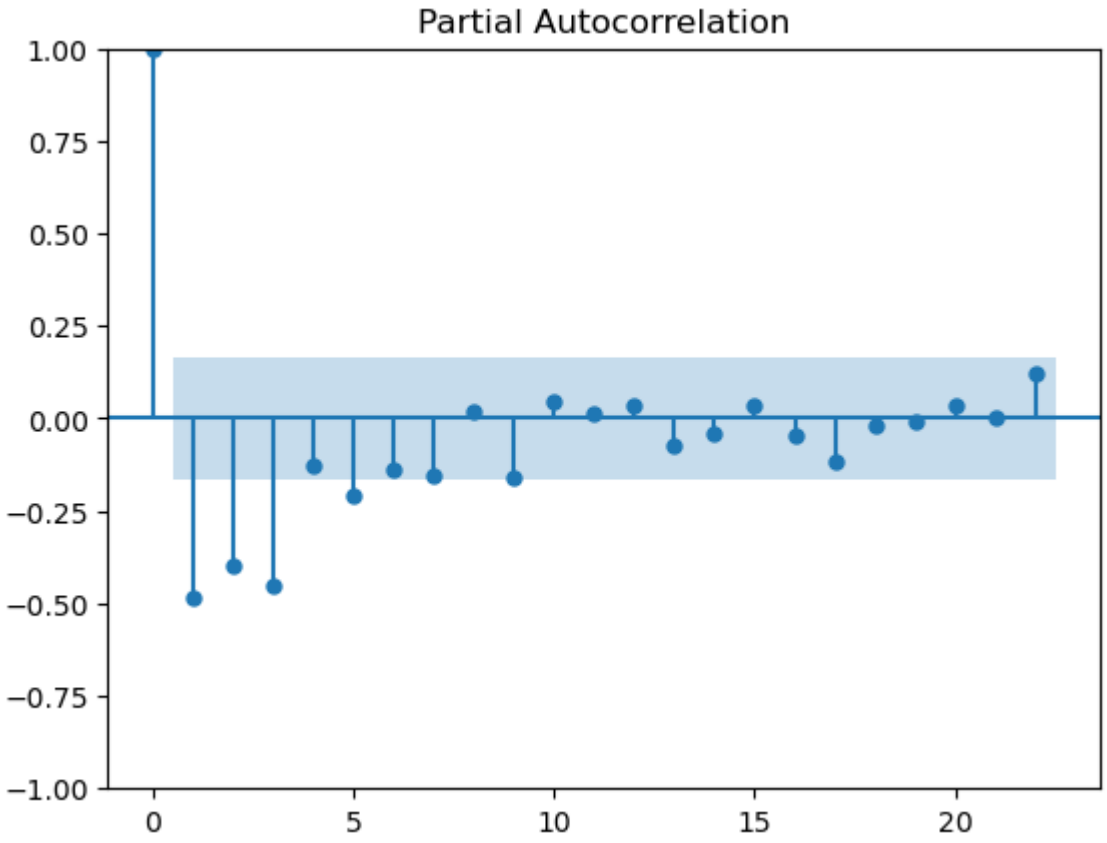


In [54]:
```
z=y[:-1]
plt.subplot(2, 1, 1)
plt.plot(z,second_diff)
plt.title("Second difference",fontsize=11,
          color="blue")
```

```
#plt.subplot(2, 3, 6)
fig = tsaplots.plot_acf(second_diff)
plt.show()



plot_pacf(second_diff,method="ywm")
plt.show()
```







In [ ]:

In [55]:

ARMA(1,1)

In [ ]:

In [ ]:

In [88]:
```
!pip install pmdarima
import pmdarima as pm
from pmdarima.arima import auto_arima
```

```
Requirement already satisfied: pmdarima in /home/gauree/anaconda3/lib/python3.
9/site-packages (2.0.1)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /home/gauree/ana
conda3/lib/python3.9/site-packages (from pmdarima) (63.4.1)
Requirement already satisfied: joblib>=0.11 in /home/gauree/anaconda3/lib/pyth
on3.9/site-packages (from pmdarima) (1.1.0)
Requirement already satisfied: numpy>=1.21 in /home/gauree/anaconda3/lib/pytho
n3.9/site-packages (from pmdarima) (1.21.5)
Requirement already satisfied: pandas>=0.19 in /home/gauree/anaconda3/lib/pyth
on3.9/site-packages (from pmdarima) (1.4.4)
Requirement already satisfied: statsmodels>=0.13.2 in /home/gauree/anaconda3/l
ib/python3.9/site-packages (from pmdarima) (0.13.5)
Requirement already satisfied: scipy>=1.3.2 in /home/gauree/anaconda3/lib/pyth
on3.9/site-packages (from pmdarima) (1.9.1)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /home/gaure
e/anaconda3/lib/python3.9/site-packages (from pmdarima) (0.29.32)
Requirement already satisfied: urllib3 in /home/gauree/anaconda3/lib/python3.
9/site-packages (from pmdarima) (1.26.11)
Requirement already satisfied: scikit-learn>=0.22 in /home/gauree/anaconda3/li
b/python3.9/site-packages (from pmdarima) (1.0.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/gauree/anaconda
3/lib/python3.9/site-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /home/gauree/anaconda3/lib/pyth
on3.9/site-packages (from pandas>=0.19->pmdarima) (2022.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /home/gauree/anaconda3/
lib/python3.9/site-packages (from scikit-learn>=0.22->pmdarima) (2.2.0)
Requirement already satisfied: packaging>=21.3 in /home/gauree/anaconda3/lib/p
ython3.9/site-packages (from statsmodels>=0.13.2->pmdarima) (21.3)
Requirement already satisfied: patsy>=0.5.2 in /home/gauree/anaconda3/lib/pyth
on3.9/site-packages (from statsmodels>=0.13.2->pmdarima) (0.5.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /home/gauree/anacon
da3/lib/python3.9/site-packages (from packaging>=21.3->statsmodels>=0.13.2->pm
darima) (3.0.9)
Requirement already satisfied: six in /home/gauree/anaconda3/lib/python3.9/sit
e-packages (from patsy>=0.5.2->statsmodels>=0.13.2->pmdarima) (1.16.0)
```

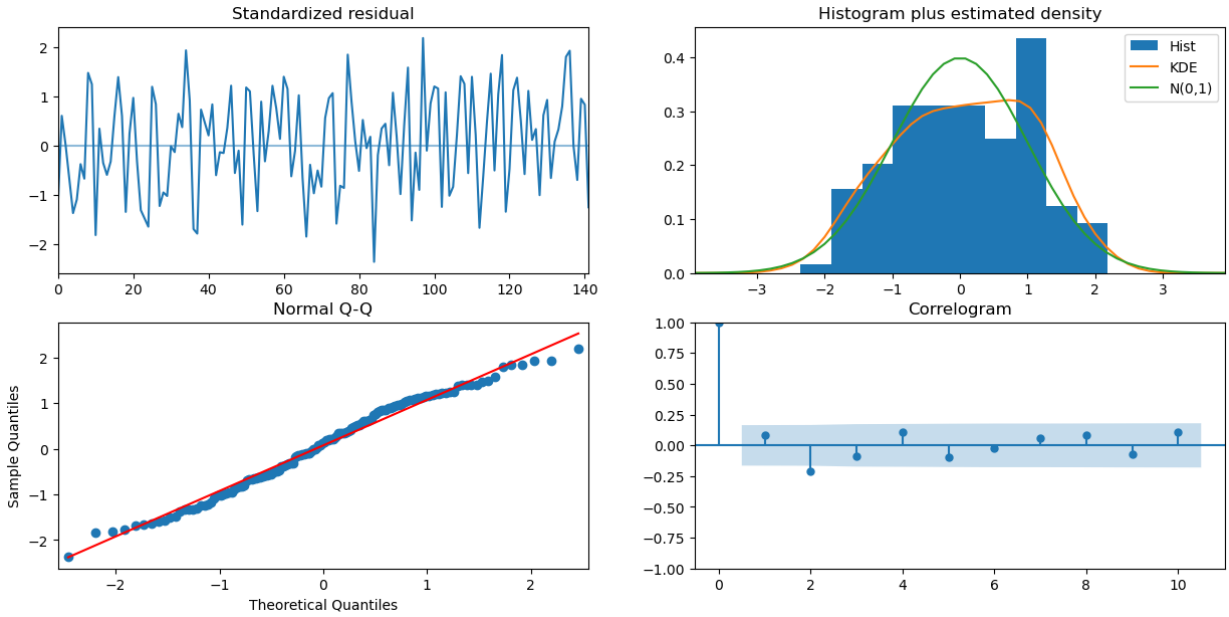
In [107…

```
model = pm.auto_arima(temp_avg, start_p=1, start_q=1,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=1, max_q=1, # maximum p and q
                      m=1,              # frequency of series
                      d=0,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

print(model.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=-226.313, Time=0.05 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=231.072, Time=0.01 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=-215.148, Time=0.01 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=86.124, Time=0.02 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=-224.268, Time=0.03 sec

Best model:  ARIMA(1,0,1)(0,0,0)[0]
Total fit time: 0.120 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  142
Model:               SARIMAX(1, 0, 1)   Log Likelihood                 116.156
Date:                Thu, 03 Nov 2022   AIC                           -226.313
Time:                        11:42:56   BIC                           -217.445
Sample:                             0   HQIC                          -222.709
                                  - 142
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9938      0.009    107.567      0.000       0.976       1.012
ma.L1         -0.4365      0.075     -5.847      0.000      -0.583      -0.290
sigma2         0.0111      0.002      6.324      0.000       0.008       0.015
===================================================================================
Ljung-Box (L1) (Q):                   0.98   Jarque-Bera (JB):
4.51
Prob(Q):                              0.32   Prob(JB):
0.10
Heteroskedasticity (H):               1.11   Skew:
-0.13
Prob(H) (two-sided):                  0.73   Kurtosis:
2.17
===================================================================================

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

In [108…

```
model.plot_diagnostics(figsize=(15,7))
plt.show()
```



In [111…
```
# fit an ARIMA model and plot residual errors
from pandas import datetime
from pandas import read_csv
from pandas import DataFrame
from statsmodels.tsa.arima.model import ARIMA
#from matplotlib import pyplot
```

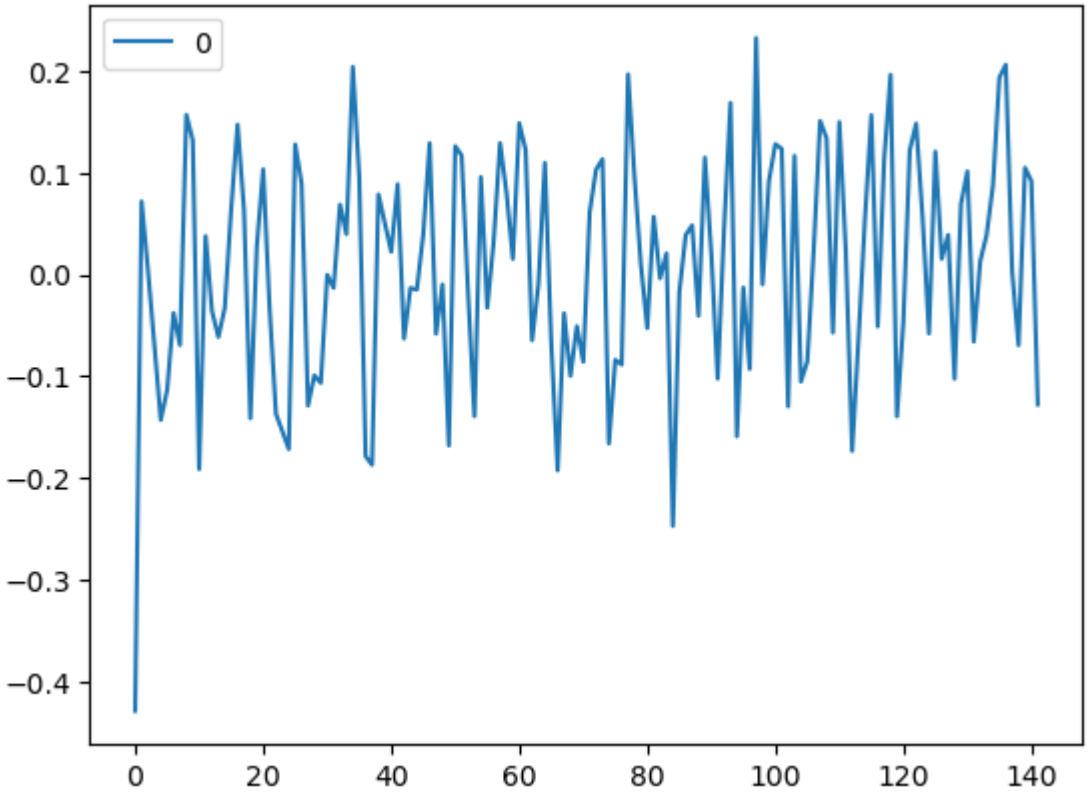
```
/tmp/ipykernel_26449/2479545078.py:2: FutureWarning: The pandas.datetime class is deprecated and will be removed from pandas in a future version. Import from datetime module instead.
  from pandas import datetime
```

In [113…
```
model = ARIMA(temp_avg, order=(1,0,1))
model_fit = model.fit()
```

In [114…
```
import statsmodels.api as sm
```

In [115…
```
from statsmodels.graphics import tsaplots
import matplotlib.pyplot as plt
```

In [118…
```
residuals = DataFrame(model_fit.resid)
residuals.plot()
plt.show()
```



The above is residual plot for ARMA(1,1).

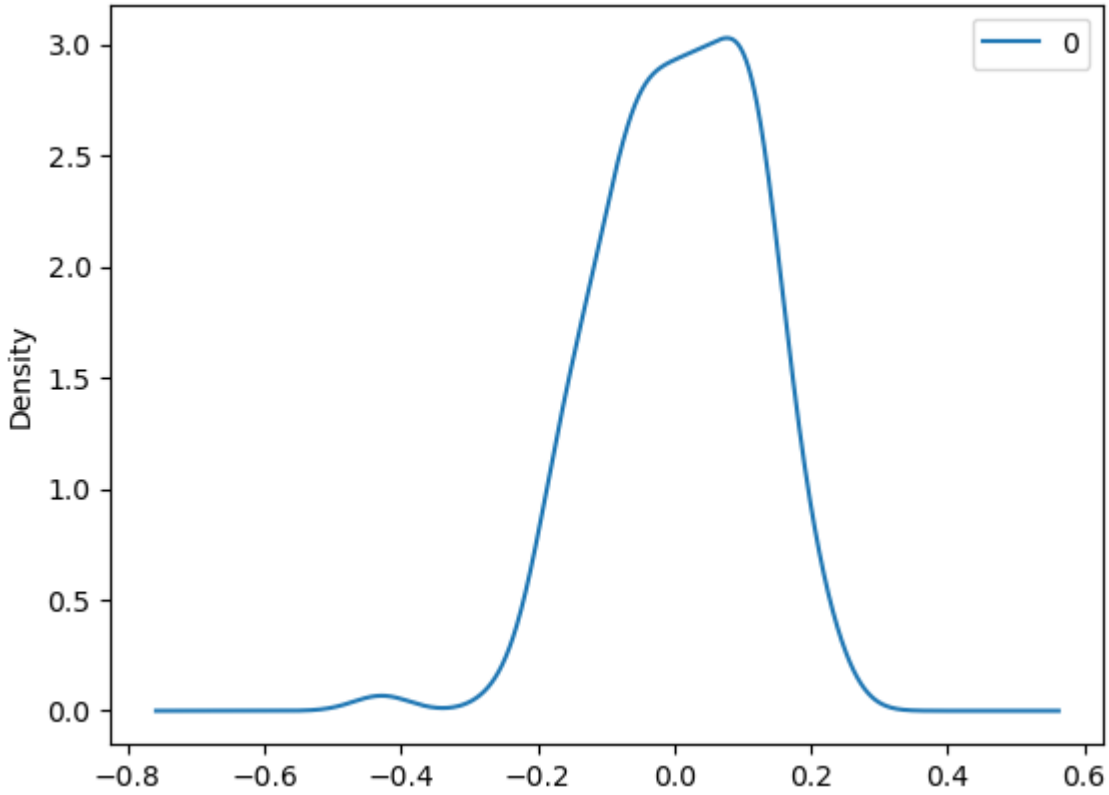
The residual plot shows clearly, that the residuals varry between range [-0.2,0.2]

In [ ]:

In [ ]:

In [ ]:

In [119…
```
residuals.plot(kind='kde')
plt.show()
```



In [120…
```
print(residuals.describe())
```

```
                0
count  142.000000
mean     0.007072
std      0.111371
min     -0.428407
25%     -0.067608
50%      0.015306
75%      0.096069
max      0.232144
```

The maximum residual is 2.3 which is much less than maximum residual in AR(1) model.

In [ ]:

AR(1)

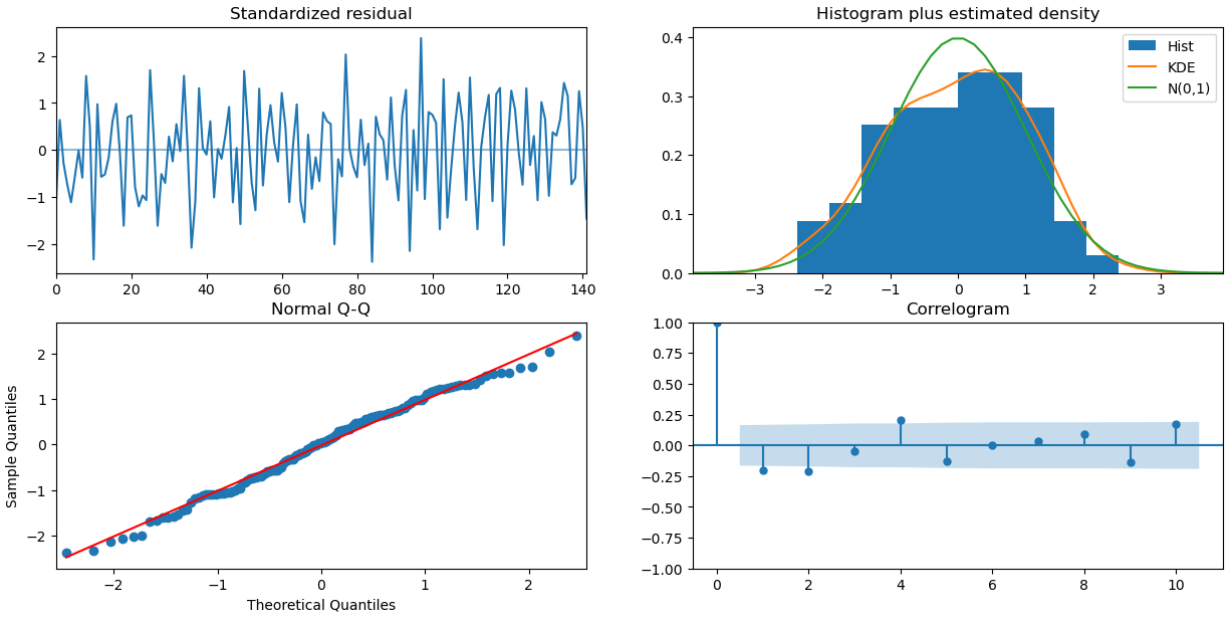
In [121…
```
model = pm.auto_arima(temp_avg, start_p=1, start_q=0,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=1, max_q=0, # maximum p and q
                      m=1,              # frequency of series
                      d=0,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)
```

```
print(model.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=-215.148, Time=0.02 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=231.072, Time=0.01 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=-214.501, Time=0.03 sec

Best model:  ARIMA(1,0,0)(0,0,0)[0]
Total fit time: 0.059 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  142
Model:               SARIMAX(1, 0, 0)   Log Likelihood                 109.574
Date:                Thu, 03 Nov 2022   AIC                           -215.148
Time:                        11:58:08   BIC                           -209.236
Sample:                             0   HQIC                          -212.746
                                  - 142
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9786      0.018     55.357      0.000       0.944       1.013
sigma2         0.0122      0.002      7.132      0.000       0.009       0.016
==============================================================================
=====
Ljung-Box (L1) (Q):                   5.82   Jarque-Bera (JB):
2.70
Prob(Q):                              0.02   Prob(JB):
0.26
Heteroskedasticity (H):               1.26   Skew:
-0.17
Prob(H) (two-sided):                  0.44   Kurtosis:
2.42
==============================================================================
=====

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

In [122…

```
model.plot_diagnostics(figsize=(15,7))
plt.show()
```

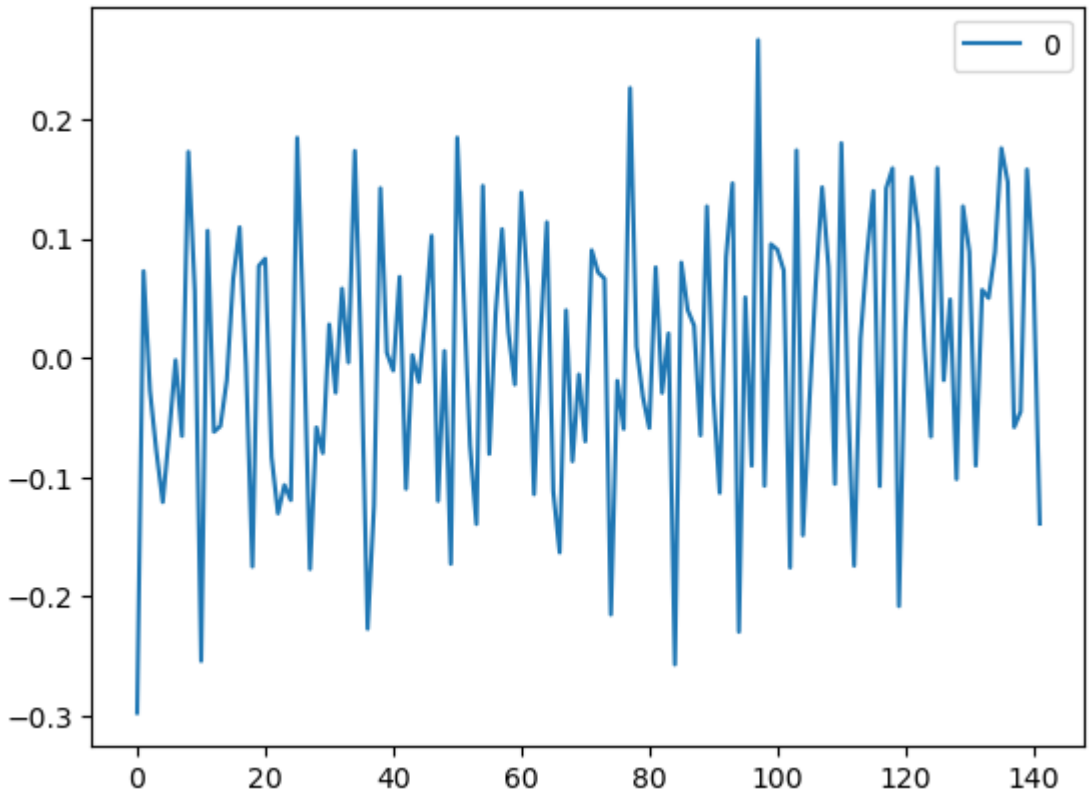


In [128…
```
model = ARIMA(temp_avg, order=(1,0,0))
model_fit = model.fit()
```

In [129…
```
import statsmodels.api as sm
```

In [130…
```
from statsmodels.graphics import tsaplots
import matplotlib.pyplot as plt
```

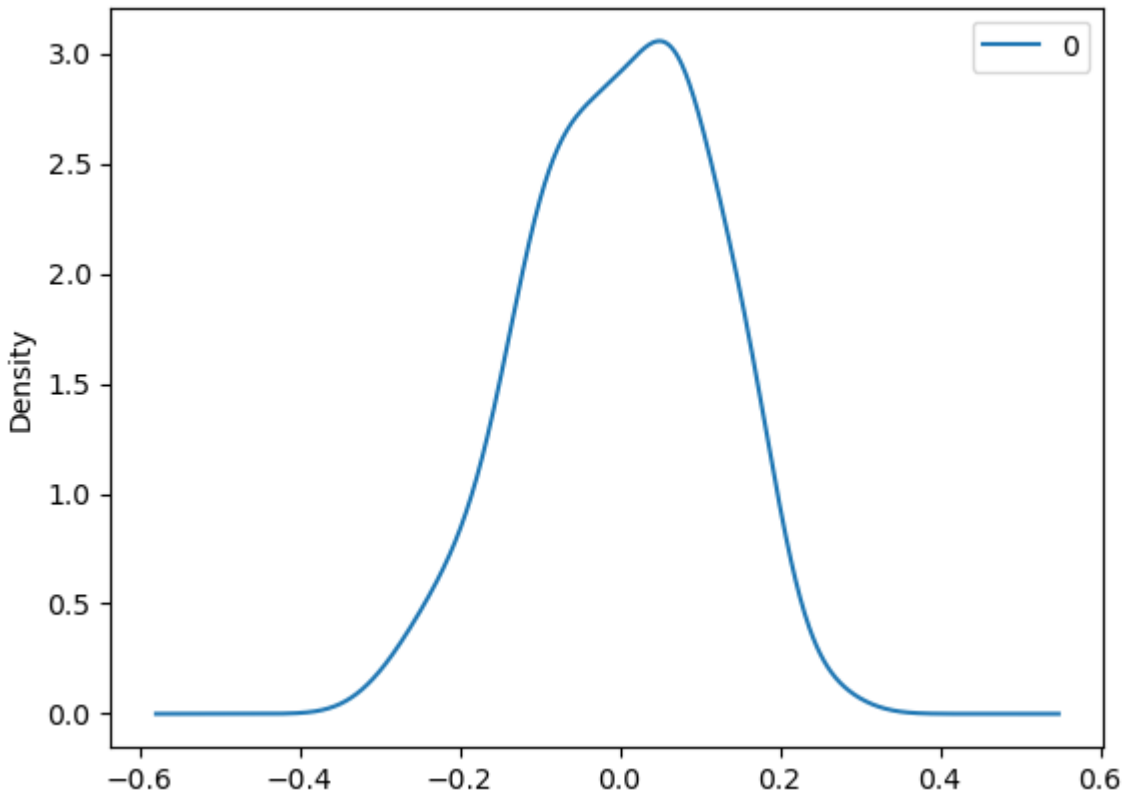
In [131…
```
residuals = DataFrame(model_fit.resid)
residuals.plot()
plt.show()
```



In [ ]:
```
The residuals in AR
```

In [ ]:
```

```

In [132…
```
residuals.plot(kind='kde')
plt.show()
```



In [133…
```
print(residuals.describe())
```

```
                0
count  142.000000
mean     0.002013
std      0.113274
min     -0.297407
25%     -0.079360
50%      0.007192
75%      0.084024
max      0.266279
```

In [ ]:
```

```