In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
from datetime import timedelta
from pandas.plotting import register_matplotlib_converters
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima_model import ARMA
register_matplotlib_converters()
from time import time
```

Ice Cream Production Data

In [2]:
```python
def parser(s):
    return datetime.strptime(s, '%Y-%m-%d')
```

In [3]:
```python
#read data
production_ice_cream = pd.read_csv('ice_cream.csv', parse_dates=[0], index_
```

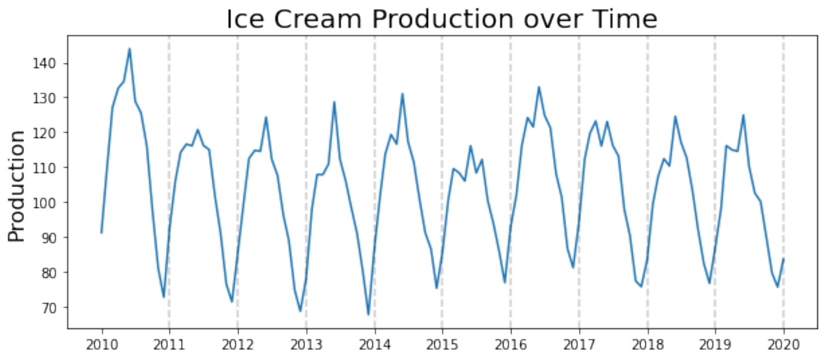
In [4]:
```python
production_ice_cream.rename('production', inplace = True)
```

Out[4]:
```
DATE
1972-01-01     59.9622
1972-02-01     67.0605
1972-03-01     74.2350
1972-04-01     78.1120
1972-05-01     84.7636
                ...
2019-09-01    100.1741
2019-10-01     90.1684
2019-11-01     79.7223
2019-12-01     75.7094
2020-01-01     83.6290
Name: production, Length: 577, dtype: float64
```

In [5]:
```python
#infer the frequency of the data
catfish_sales = production_ice_cream.asfreq(pd.infer_freq(production_ice_cr
```

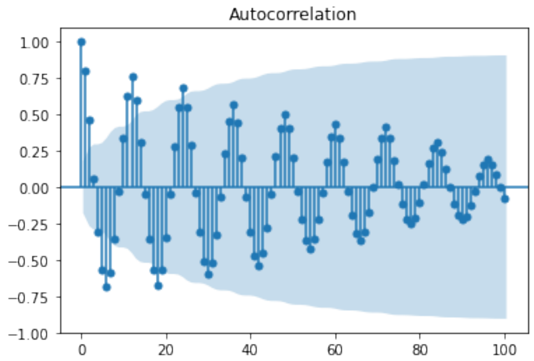
In [6]:
```python
start_date = pd.to_datetime('2010-01-01')
production_ice_cream = production_ice_cream[start_date:]
```

In [7]:
```python
plt.figure(figsize=(10,4))
plt.plot(production_ice_cream)
plt.title('Ice Cream Production over Time', fontsize=20)
plt.ylabel('Production', fontsize=16)
for year in range(2011, 2021):
    plt.axvline(pd.to_datetime(str(year)+'-01-01'), color='k', linestyle='-
```



ACF

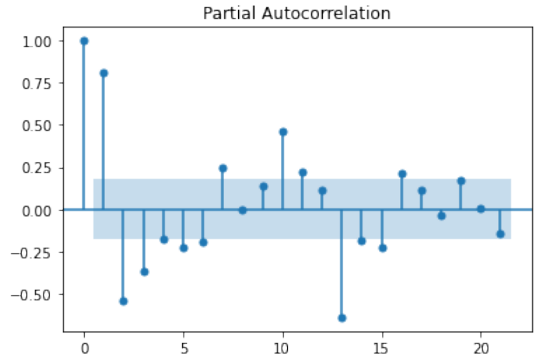
In [8]:
```python
acf_plot = plot_acf(production_ice_cream, lags=100)
```



Based on decaying ACF, we are likely dealing with an Auto Regressive process

PACF

In [9]:
```python
pacf_plot = plot_pacf(production_ice_cream)
```



Based on PACF, we should start with an Auto Regressive model with lags 1, 2, 3

Get training and testing sets

In [10]:
```
train_end = datetime(2018,12,1)
test_end = datetime(2019,12,1)

train_data = production_ice_cream[:train_end]
test_data = production_ice_cream[train_end + timedelta(days=1):test_end]
```

Fit the AR Model

In [11]:
```
# define model
model = ARMA(train_data, order=(3,0))
```

```
/Users/siamakfarjami/opt/anaconda3/envs/UNI/lib/python3.8/site-packages/sta
tsmodels/tsa/arima_model.py:472: FutureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Users/siamakfarjami/opt/anaconda3/envs/UNI/lib/python3.8/site-packages/sta
tsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information
was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

In [12]:
```
#fit the model
start = time()
model_fit = model.fit()
end = time()
print('Model Fitting Time:', end - start)
```

```
Model Fitting Time: 10.277784824371338
```

In [13]:
```
#summary of the model
print(model_fit.summary())
```

```
                              ARMA Model Results
==============================================================================
===
Dep. Variable:             production   No. Observations:
108
Model:                     ARMA(3, 0)   Log Likelihood                  -374.
085
Method:                       css-mle   S.D. of innovations                7.
642
Date:                Thu, 26 Nov 2020   AIC                              758.
170
Time:                        00:18:44   BIC                              771.
580
Sample:                    01-01-2010   HQIC                             763.
607
                         - 12-01-2018
==============================================================================
=========
                       coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const              103.5743      1.809     57.265      0.000     100.029
107.119
ar.L1.production     1.0469      0.088     11.960      0.000       0.875
1.218
ar.L2.production    -0.0523      0.134     -0.391      0.695      -0.314
0.210
ar.L3.production    -0.4044      0.089     -4.542      0.000      -0.579
-0.230
                                    Roots
==============================================================================
==
                  Real          Imaginary           Modulus         Frequen
cy
------------------------------------------------------------------------------
--
AR.1            0.9446           -0.5767j            1.1068           -0.08
72
AR.2            0.9446           +0.5767j            1.1068            0.08
72
AR.3           -2.0186           -0.0000j            2.0186           -0.50
00
------------------------------------------------------------------------------
--
```

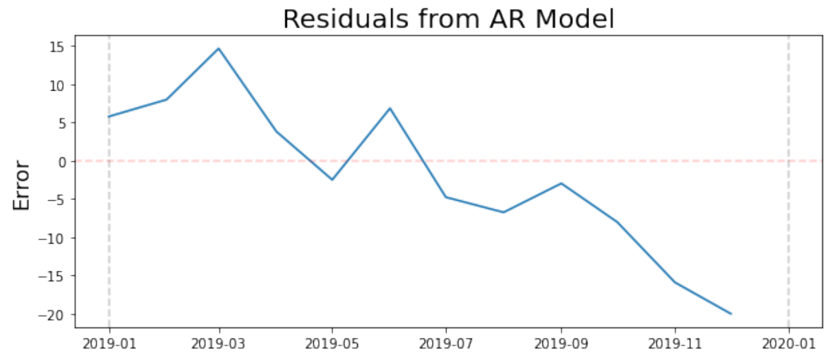
So the AR(3) model is:

$$\hat{y}_t = 103 + 1.04y_{t-1} - 0.05y_{t-2} - 0.40y_{t-3}$$

In [14]:
```
#get prediction start and end dates
pred_start_date = test_data.index[0]
pred_end_date = test_data.index[-1]
```

In [15]:
```
#get the predictions and residuals
predictions = model_fit.predict(start=pred_start_date, end=pred_end_date)
residuals = test_data - predictions
```

In [16]:
```
plt.figure(figsize=(10,4))
plt.plot(residuals)
plt.title('Residuals from AR Model', fontsize=20)
plt.ylabel('Error', fontsize=16)
plt.axhline(0, color='r', linestyle='--', alpha=0.2)
for year in range(2019,2021):
    plt.axvline(pd.to_datetime(str(year)+'-01-01'), color='k', linestyle='-
```

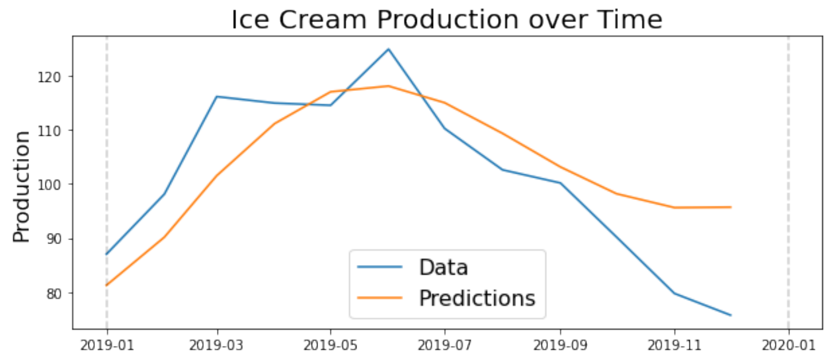


In [17]:
```
plt.figure(figsize=(10,4))

plt.plot(test_data)
plt.plot(predictions)

plt.legend(('Data', 'Predictions'), fontsize=16)

plt.title('Ice Cream Production over Time', fontsize=20)
plt.ylabel('Production', fontsize=16)
for year in range(2019,2021):
    plt.axvline(pd.to_datetime(str(year)+'-01-01'), color='k', linestyle='-
```



In [19]:
```
print('Mean Absolute Percent Error:', round(np.mean(abs(residuals/test_data
```

```
Mean Absolute Percent Error: 0.0895
```

In [20]:
```
print('Root Mean Squared Error:', np.sqrt(np.mean(residuals**2)))
```

```
Root Mean Squared Error: 9.884059832193314
```

In [ ]:

```

```