Lab 5: ARIMA models 1 Task 1 Correlations in AR models

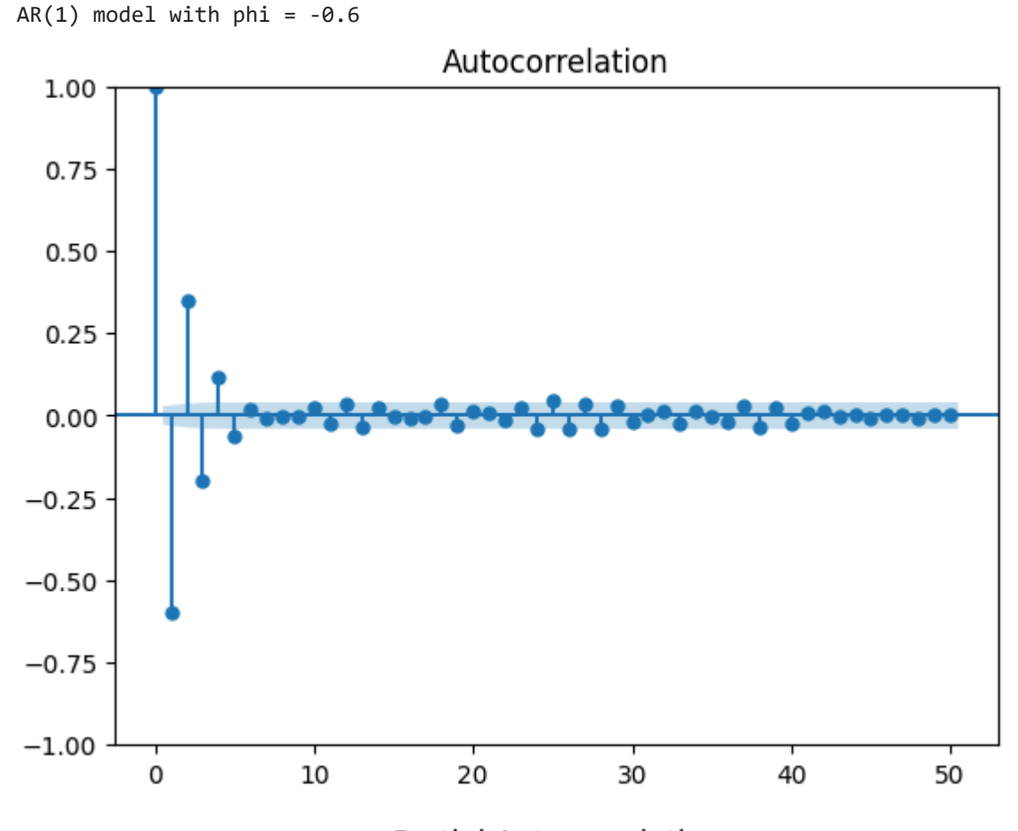
Function that calculates the values of AR(p) model(parameter burnin determines how many initial values are discarded):

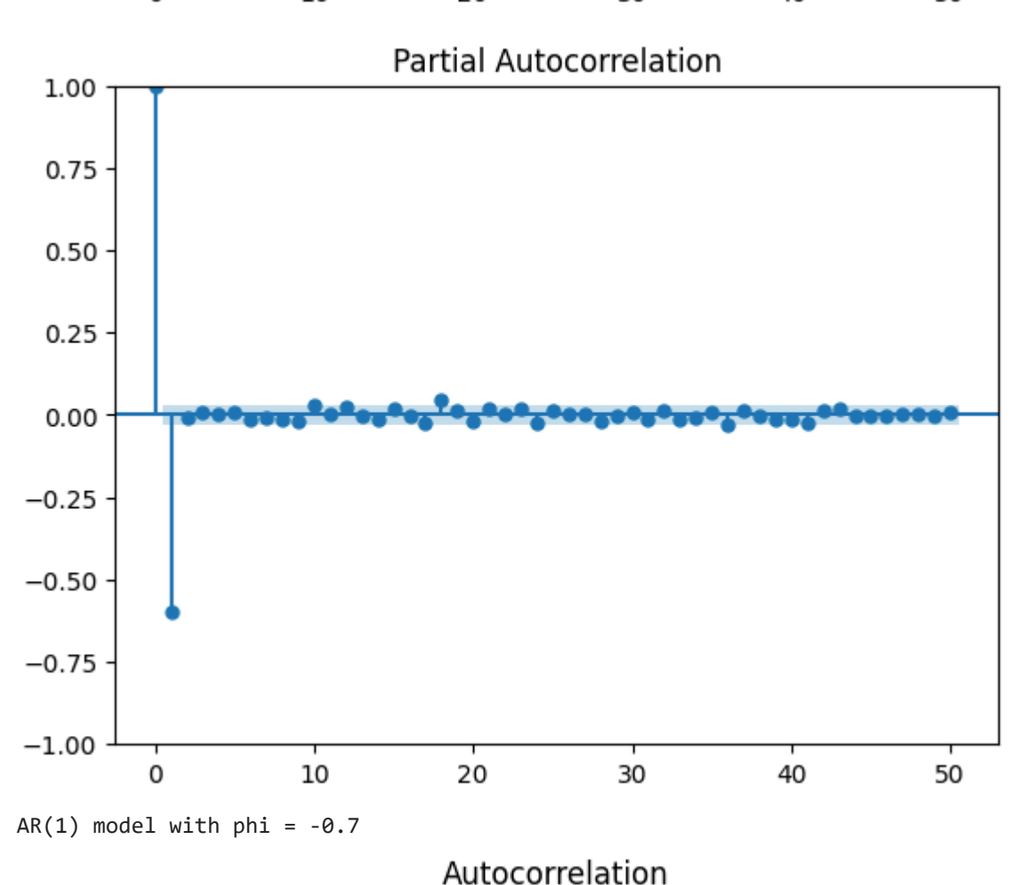
In [59]:
```python
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

def ar_model(p, phi, c, n, burnin=100):
    y = np.zeros(n + burnin)
    # Generate the time series according to the AR(p) model
    for t in range(p, n + burnin):
        y[t] = c + np.dot(phi, y[t-p:t][::-1]) + np.random.normal()
    # Discard the initial values
    return y[burnin:]
```

a) calculate 5000 values of the AR(1) model with such coefficients: -0.6, −0.7, −0.8, −0.9 b) calculate the autocorrelation (ACF) and partial autocorrelation (PACF) function for this time series

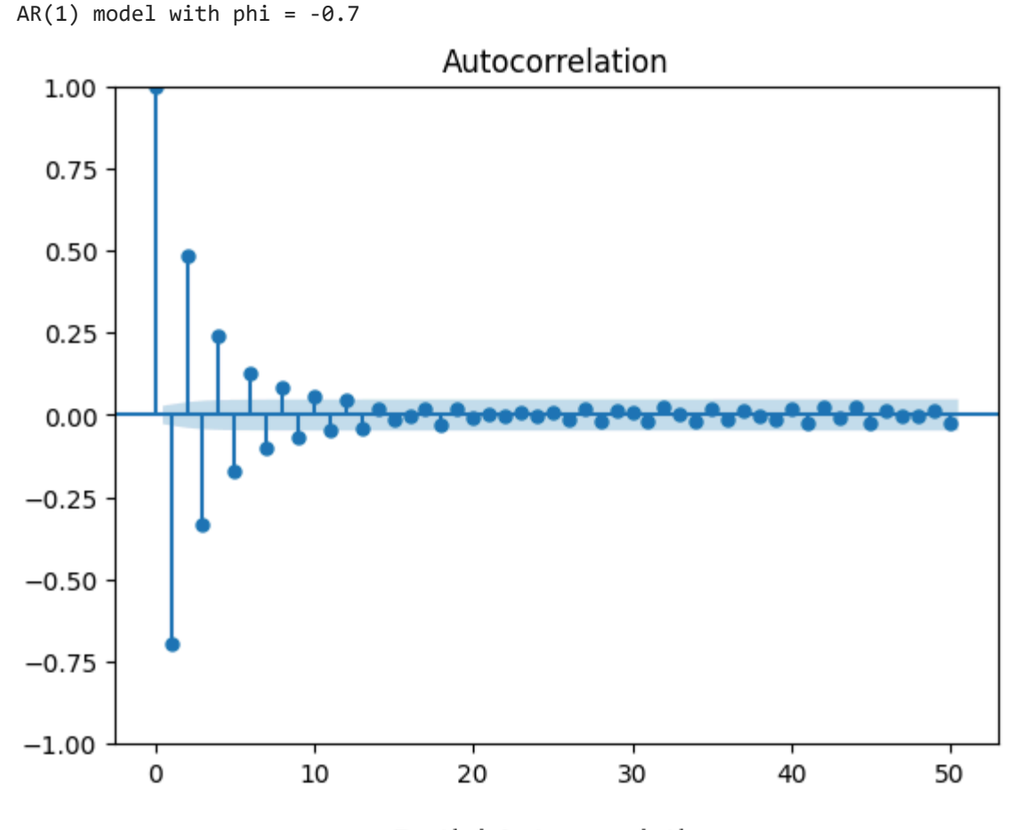
In [60]:
```python
for phi in [-0.6, -0.7, -0.8, -0.9]: # a
    y = ar_model(1, [phi], 18, 5000)
    print(f"AR(1) model with phi = {phi}")
    plot_acf(y, lags=50) # b
    plt.show()
    plot_pacf(y, lags=50) # b
    plt.show()
```

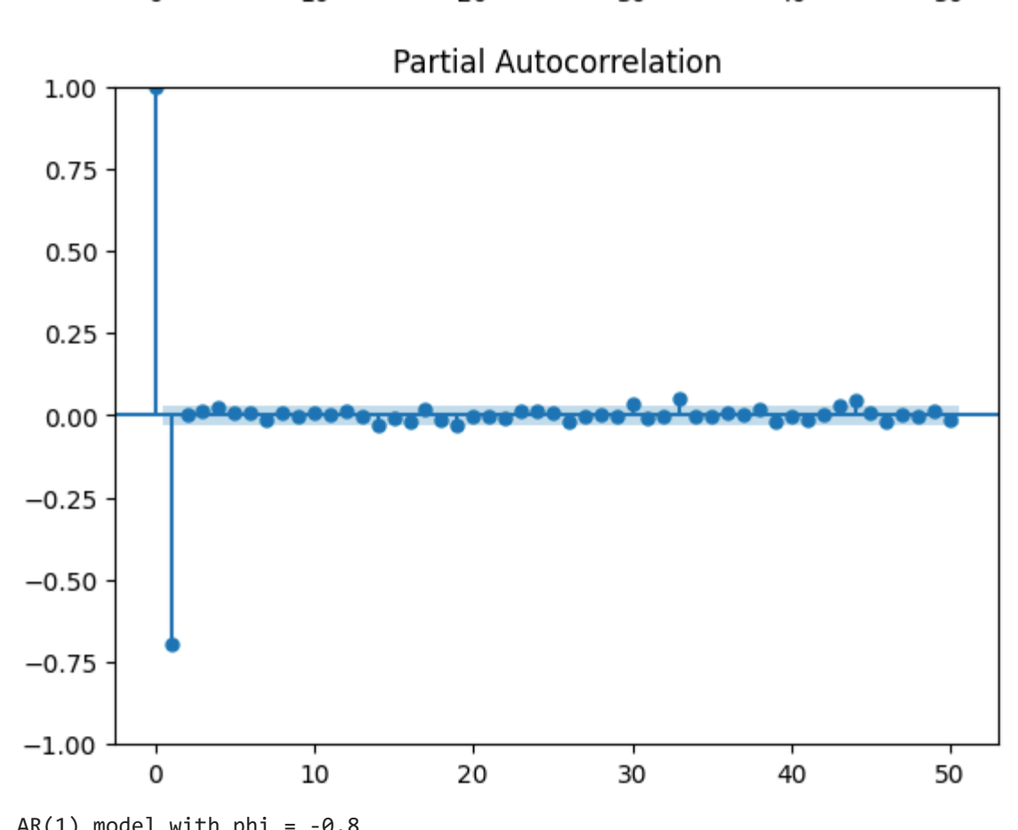
```
AR(1) model with phi = -0.6
```



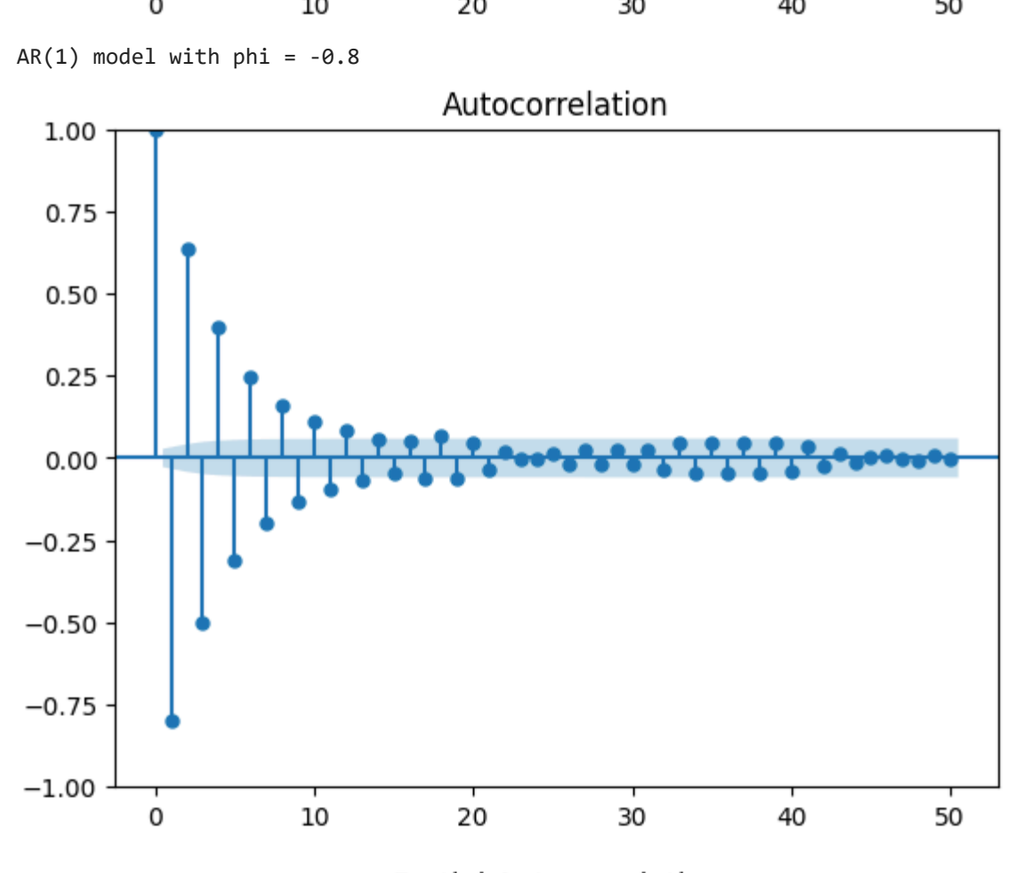


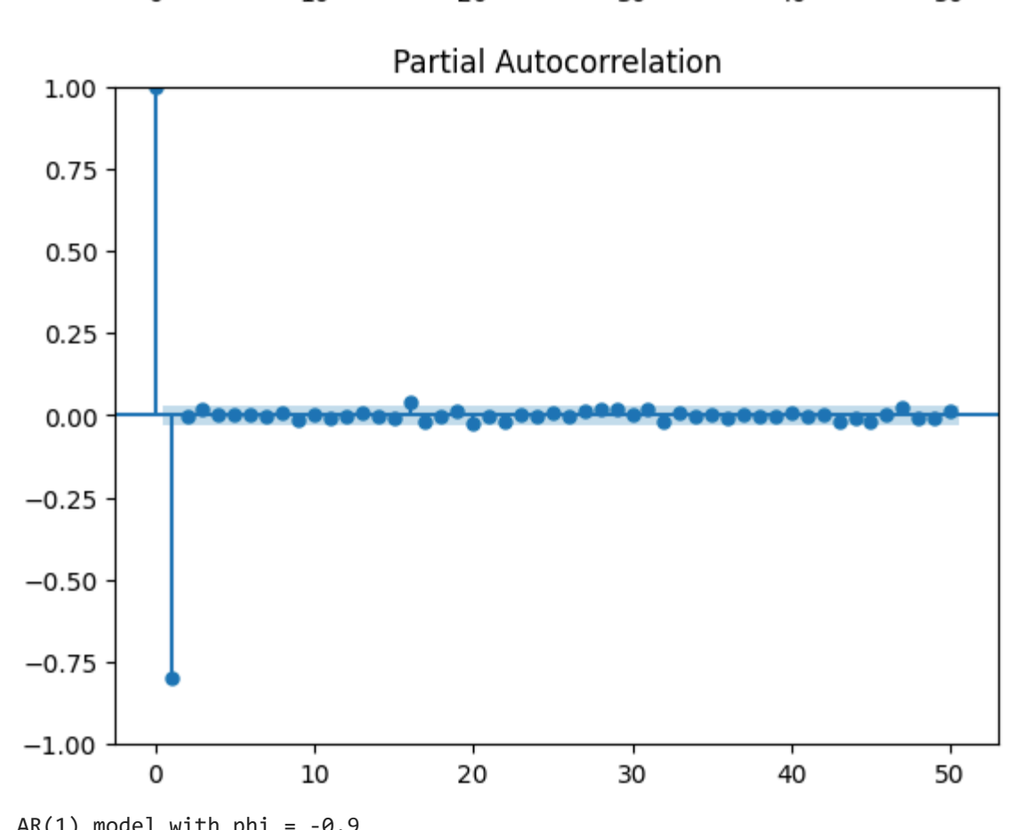
```
AR(1) model with phi = -0.7
```



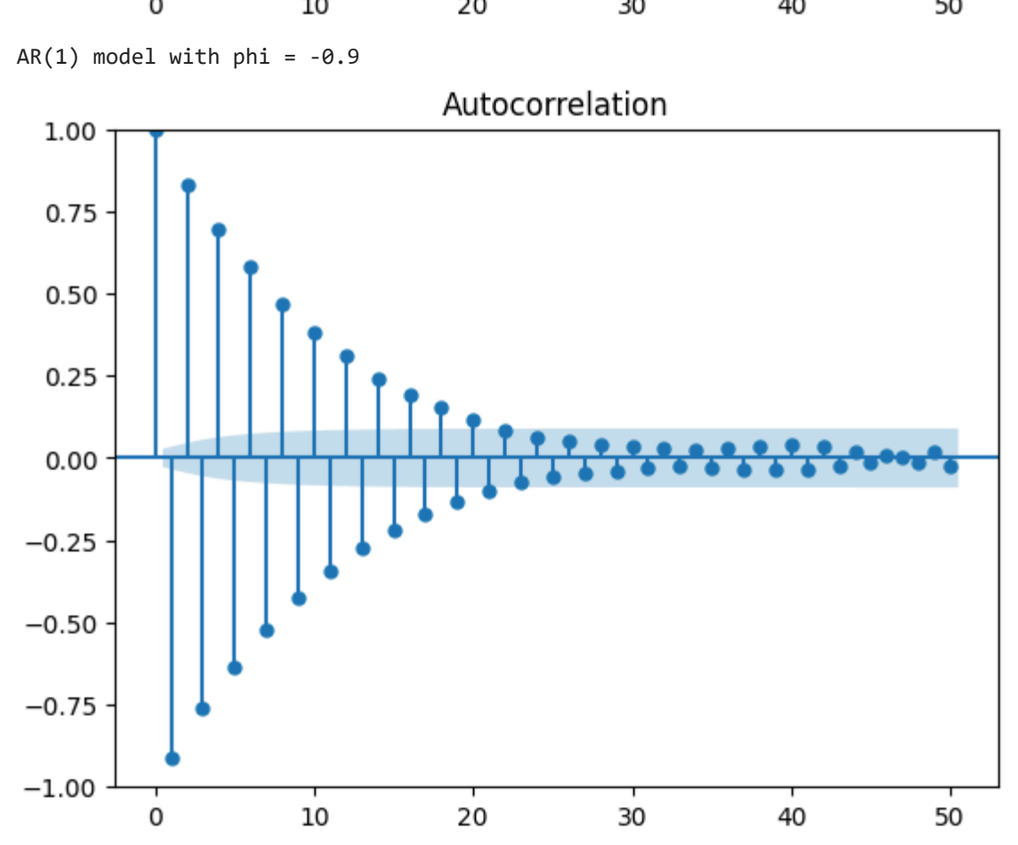


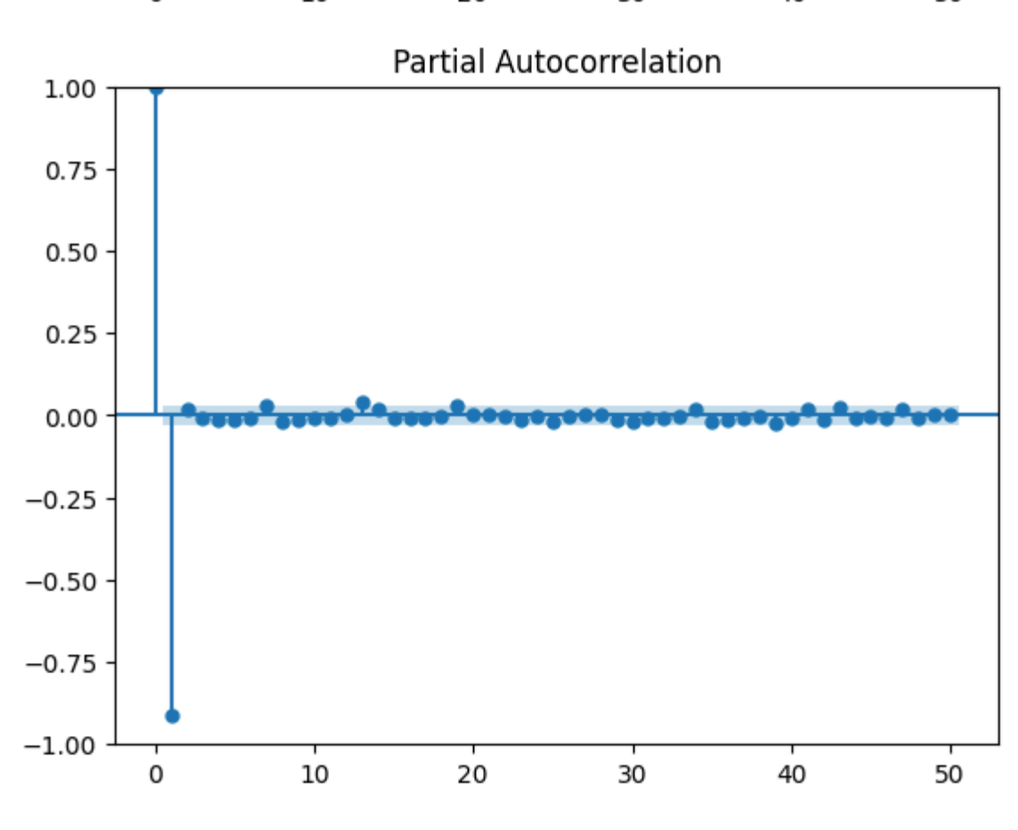
```
AR(1) model with phi = -0.8
```





```
AR(1) model with phi = -0.9
```





Now we will check the time series for stationarity using the Augmented Dickey-Fuller test.

In [61]:
```python
from statsmodels.tsa.stattools import adfuller
for phi in [-0.6, -0.7, -0.8, -0.9]:
    y = ar_model(1, [phi], 18, 5000)
    print(f"AR(1) model with phi = {phi}")
    print(" ")
    print(adfuller(y))
    print(" ")
```

```
AR(1) model with phi = -0.6

(-143.74081438649802, 0.0, 0, 4999, {'1%': -3.431658793968827, '5%': -2.862118345383404, '10%': -2.567077853953267}, 14014.671388204191)

AR(1) model with phi = -0.7

(-46.292348147266736, 0.0, 2, 4997, {'1%': -3.431659318070175, '5%': -2.862118576928539, '10%': -2.567077977213785}, 14008.55459356905)

AR(1) model with phi = -0.8

(-205.8553401203077, 0.0, 0, 4999, {'1%': -3.431658793968827, '5%': -2.862118345383404, '10%': -2.567077853953267}, 14168.137917689623)

AR(1) model with phi = -0.9

(-307.97577737231967, 0.0, 0, 4999, {'1%': -3.431658793968827, '5%': -2.862118345383404, '10%': -2.567077853953267}, 14159.9281657359)
```

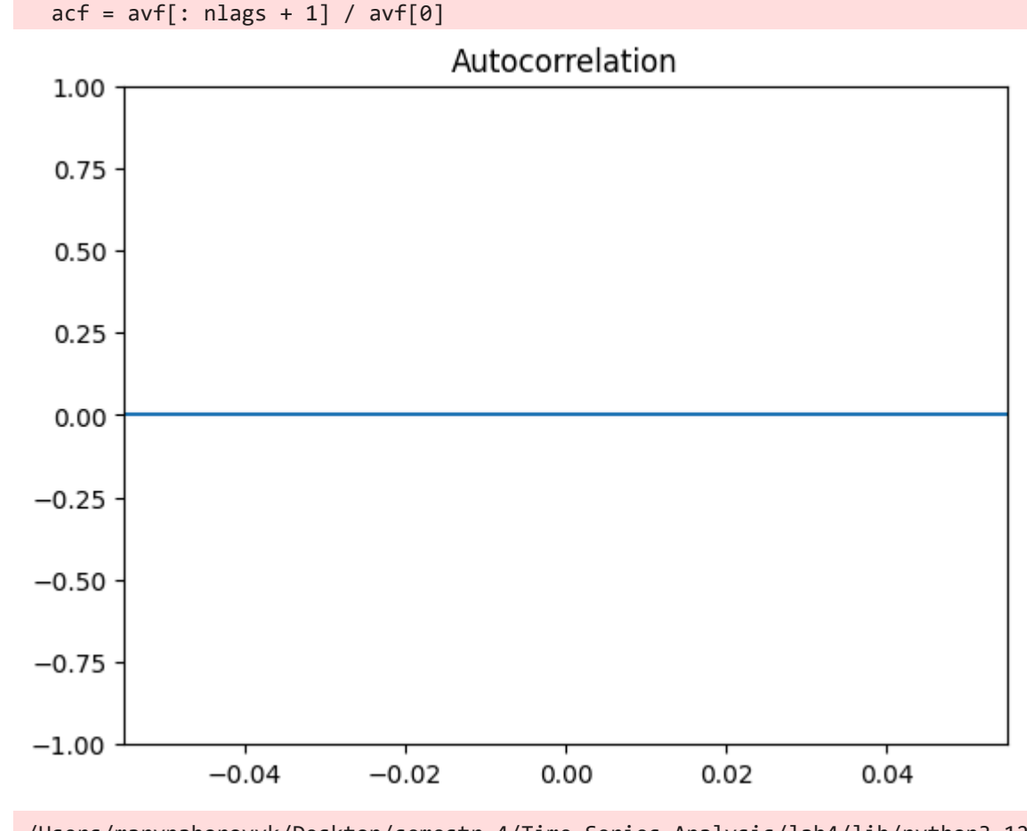
Analysis of the results: p-value > 0.05 - This implies that time-series is non-stationary. p-value <=0.05 - This implies that time-series is stationary

The results of the test show that the time series is stationary.

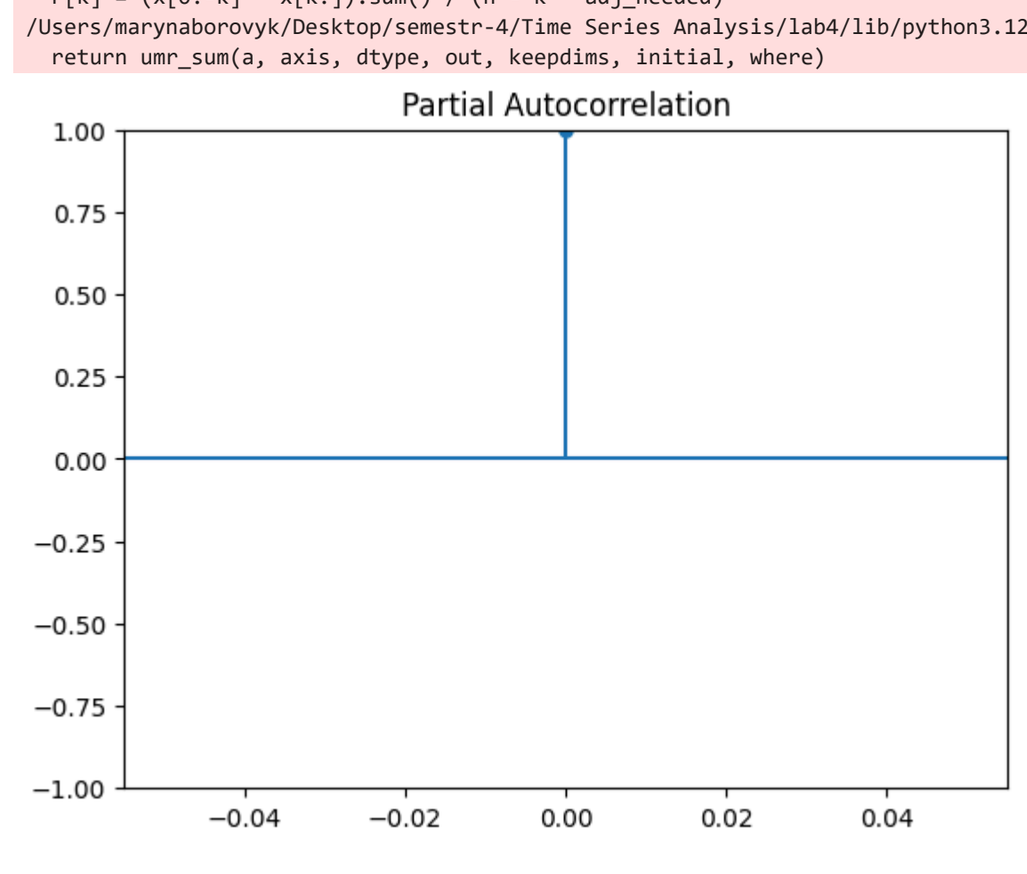
What happens when |ϕ1| > 1? We can check it by substituting the value of phi = 1.1 in our function.

In [62]:
```python
y = ar_model(1, 1.1, 18, 5000)
print(f"AR(1) model with phi = 1.1")
plot_acf(y, lags=50) # b
plt.show()
plot_pacf(y, lags=50) # b
plt.show()
```

```
AR(1) model with phi = 1.1
/var/folders/z6/r8xtfhp56gd9rzqzptq9vv880000gn/T/ipykernel_51865/1885854207.py:9: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you extract a single element from your array before performing this operation. (Deprecated NumPy 1.25.)
  y[t] = c + np.dot(phi, y[t-p:t][::-1]) + np.random.normal()
/Users/marynaborovyk/Desktop/semestr-4/Time Series Analysis/lab4/lib/python3.12/site-packages/statsmodels/tsa/stattools.py:693: RuntimeWarning: invalid value encountered in divide
  acf = avf[: nlags + 1] / avf[0]
```



```
/Users/marynaborovyk/Desktop/semestr-4/Time Series Analysis/lab4/lib/python3.12/site-packages/statsmodels/regression/linear_model.py:1481: RuntimeWarning: overflow encountered in square
  r[0] = (x ** 2).sum() / n
/Users/marynaborovyk/Desktop/semestr-4/Time Series Analysis/lab4/lib/python3.12/site-packages/statsmodels/regression/linear_model.py:1483: RuntimeWarning: overflow encountered in multiply
  r[k] = (x[0:-k] * x[k:]).sum() / (n - k * adj_needed)
/Users/marynaborovyk/Desktop/semestr-4/Time Series Analysis/lab4/lib/python3.12/site-packages/numpy/core/_methods.py:49: RuntimeWarning: invalid value encountered in reduce
  return umr_sum(a, axis, dtype, out, keepdims, initial, where)
```

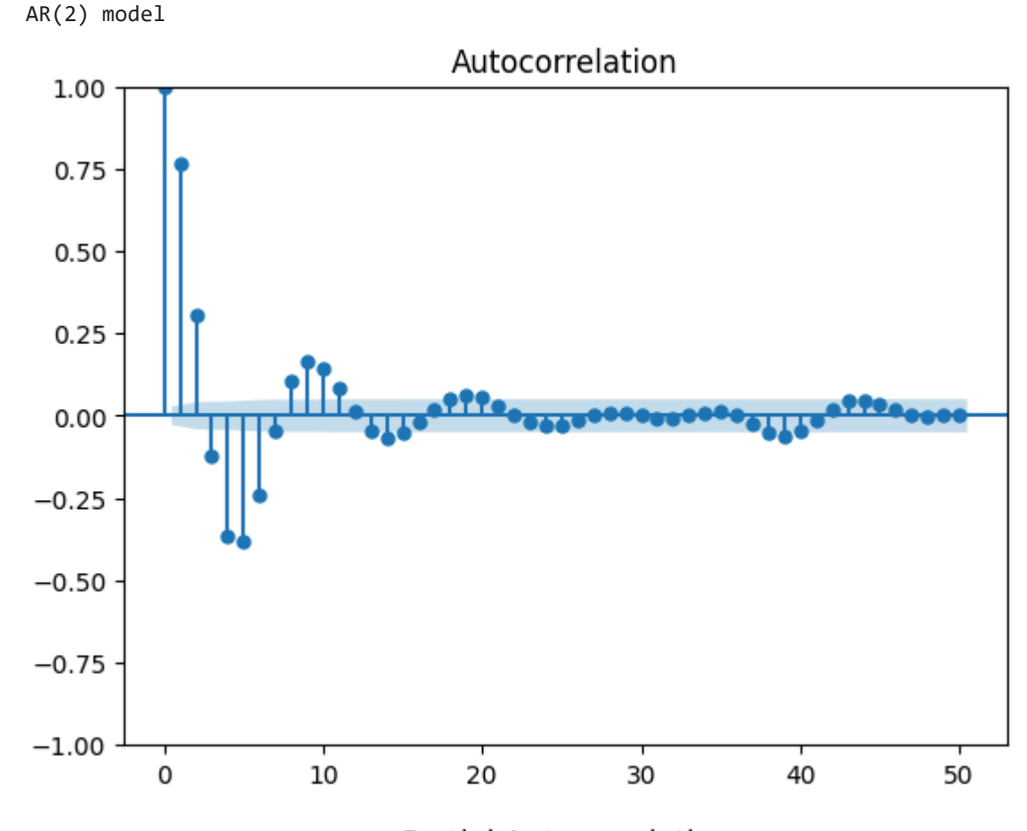


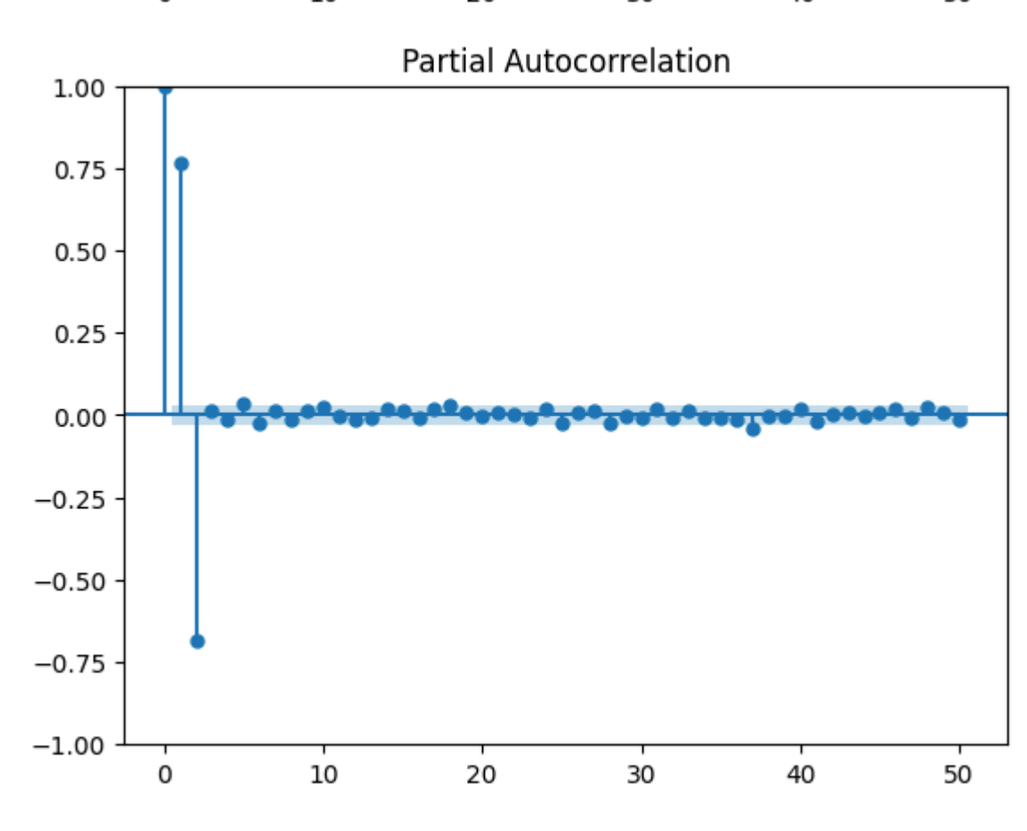
The result shows that we can't calculate the ACF and PACF for the AR(1) model with phi = 1.1.

Calculate 5000 values of the AR(2) model with coefficients: 1.3, -0.7

In [63]:
```python
y = ar_model(2, [1.3, -0.7], 8, 5000)
print("AR(2) model")
plot_acf(y, lags=50)
plt.show()
plot_pacf(y, lags=50)
plt.show()
```

```
AR(2) model
```

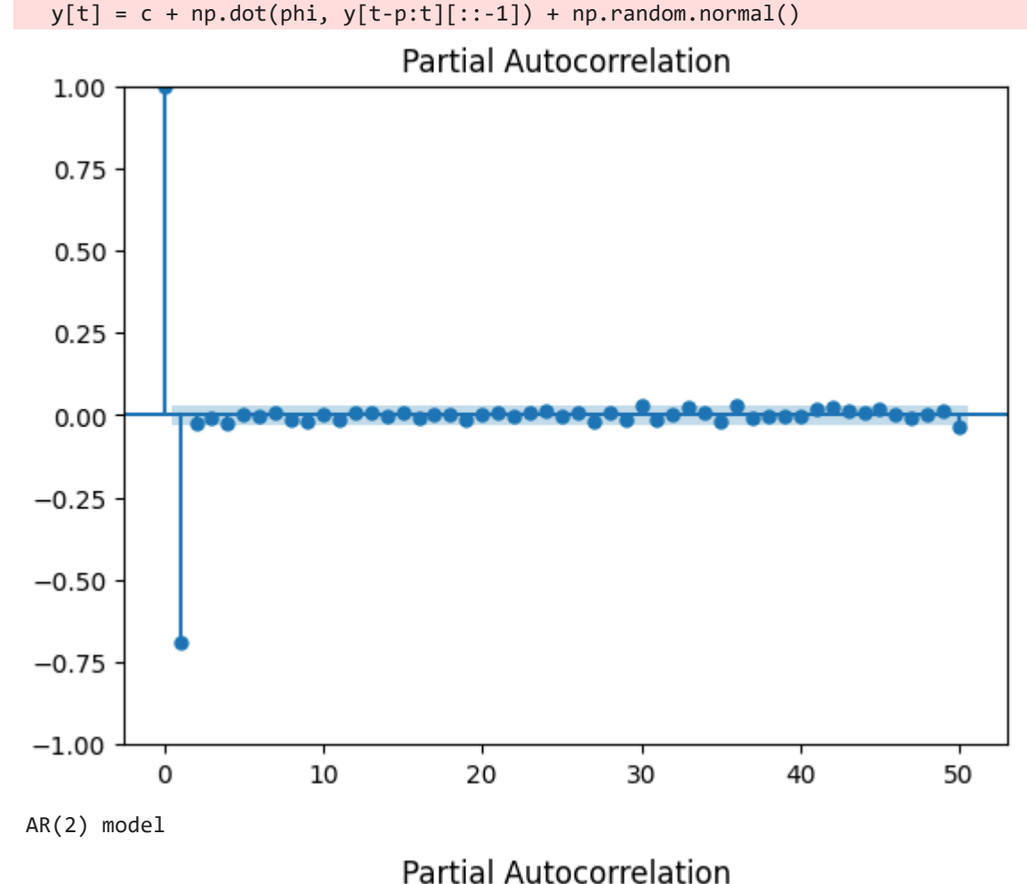




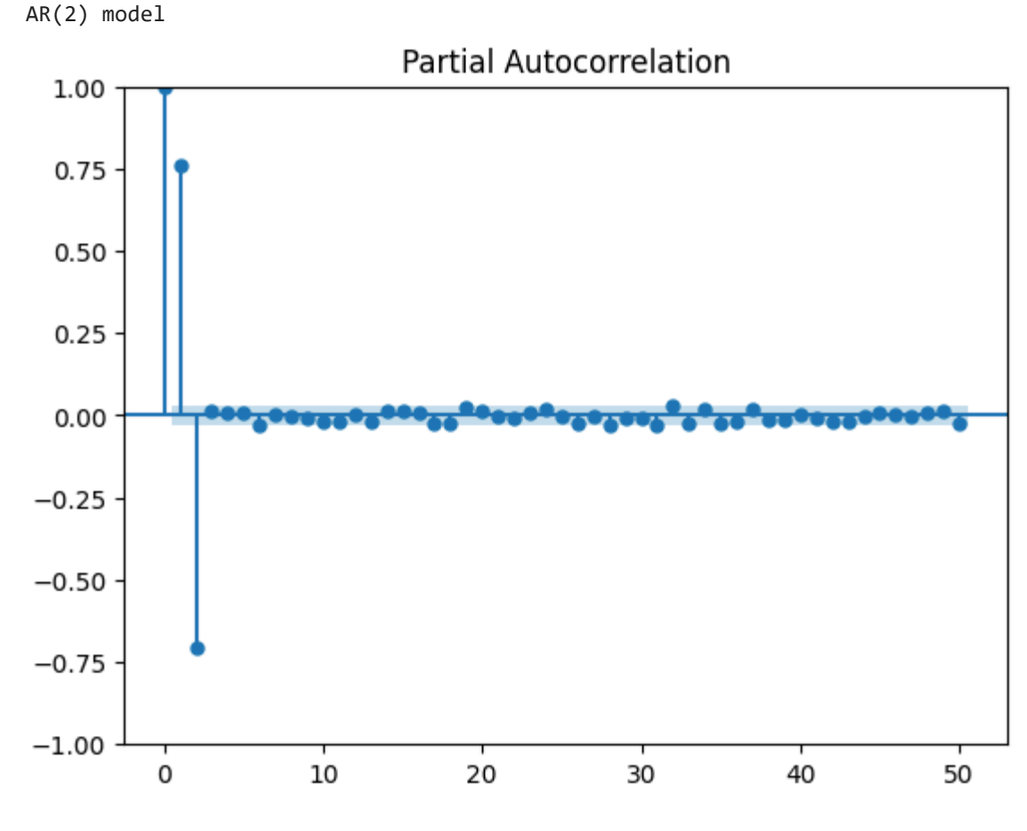
Comparing the structure of PACFs for AR(1) and AR(2) models:

In [64]:
```python
y = ar_model(1, -0.7, 18, 5000)
print("AR(1) model")
plot_pacf(y, lags=50)
plt.show()

y = ar_model(2, [1.3, -0.7], 8, 5000)
print("AR(2) model")
plot_pacf(y, lags=50)
plt.show()
```

```
AR(1) model
/var/folders/z6/r8xtfhp56gd9rzqzptq9vv880000gn/T/ipykernel_51865/1885854207.py:9: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you extract a single element from your array before performing this operation. (Deprecated NumPy 1.25.)
  y[t] = c + np.dot(phi, y[t-p:t][::-1]) + np.random.normal()
```



```
AR(2) model
```



Task 2. Correlations in MA models

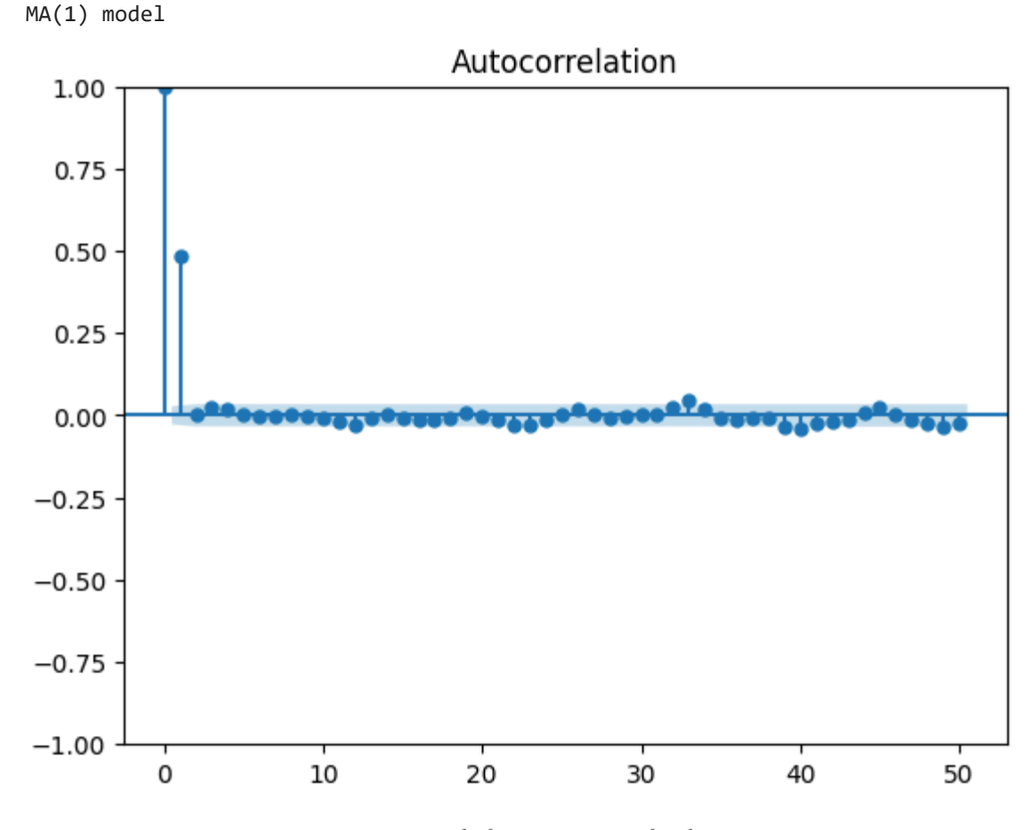
Writing a function that calculates the values of MA(q) model(parameter burnin determines how many initial values are discarded)

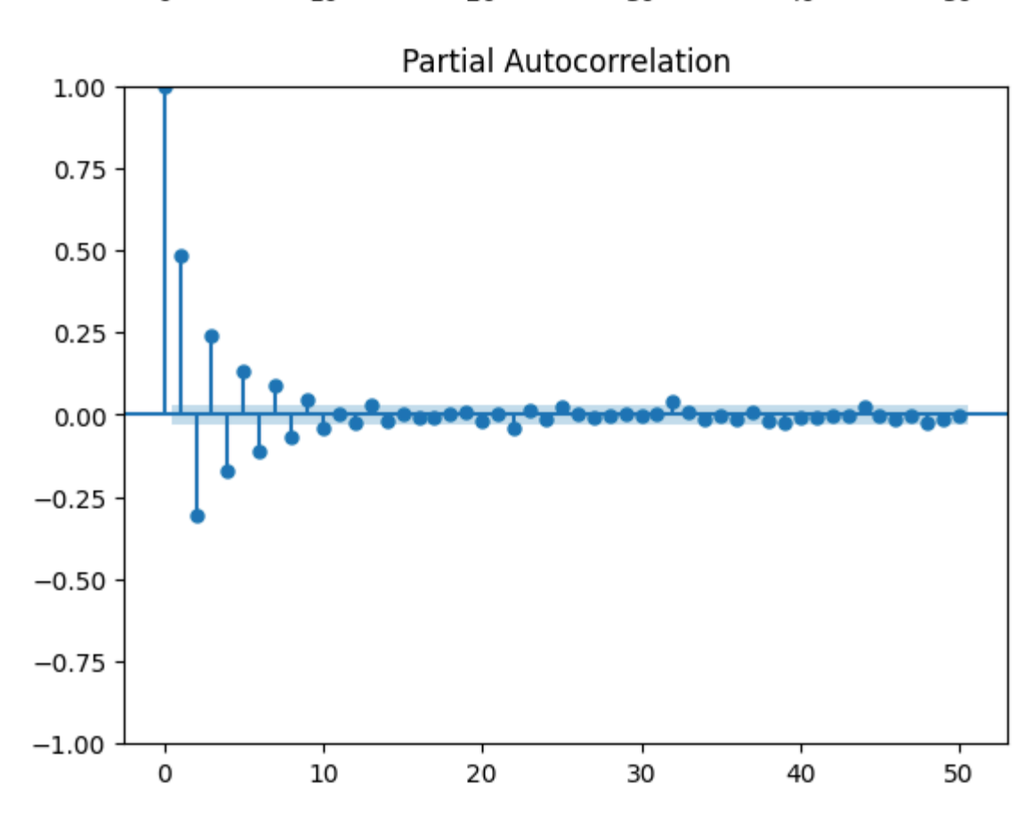
In [65]:
```python
def ma_model(q, theta, c, n, burnin=100):
    y = np.zeros(n + burnin)
    e = np.random.normal(size=n + burnin)
    for t in range(q, n + burnin):
        y[t] = c + e[t] + np.dot(theta, e[t-q:t][::-1])
    return y[burnin:]
```

a) Calculate 5000 values of MA(1) model b) Calculate the autocorrelation (ACF) and partial autocorrelation (PACF) function for this time series.

In [66]:
```python
y = ma_model(1, [0.8], 20, 5000) # a
print("MA(1) model")
plot_acf(y, lags=50) # b
plt.show()
plot_pacf(y, lags=50) # b
plt.show()
```

```
MA(1) model
```

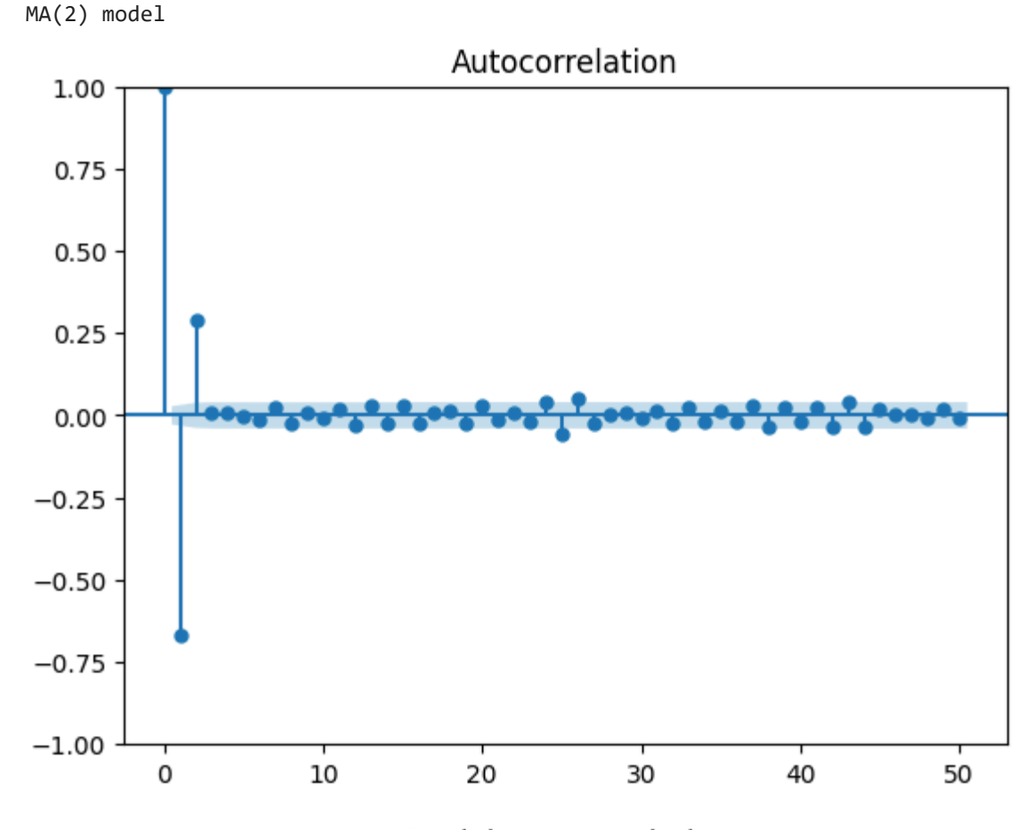


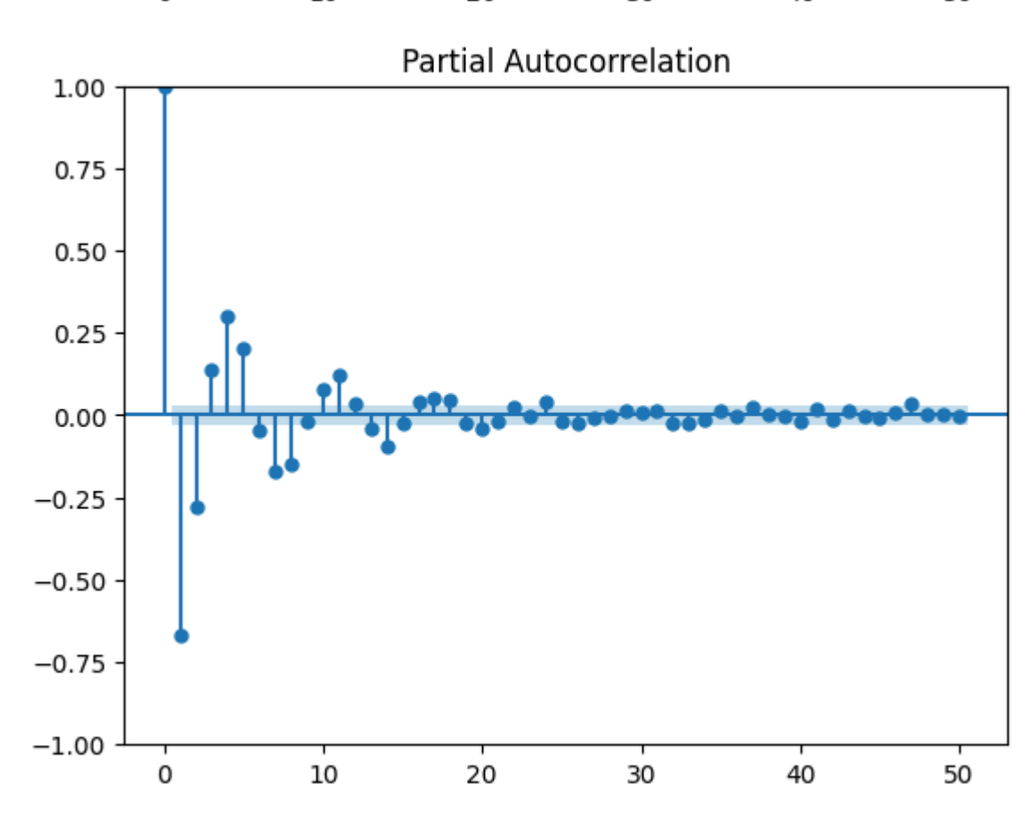


Repeat the calculations for the MA(2) model with coefficients: -1, 0.8

In [67]:
```python
y = ma_model(2, [-1, 0.8], 0, 5000) # a
print("MA(2) model")
plot_acf(y, lags=50)
plt.show()
plot_pacf(y, lags=50)
plt.show()
```

```
MA(2) model
```

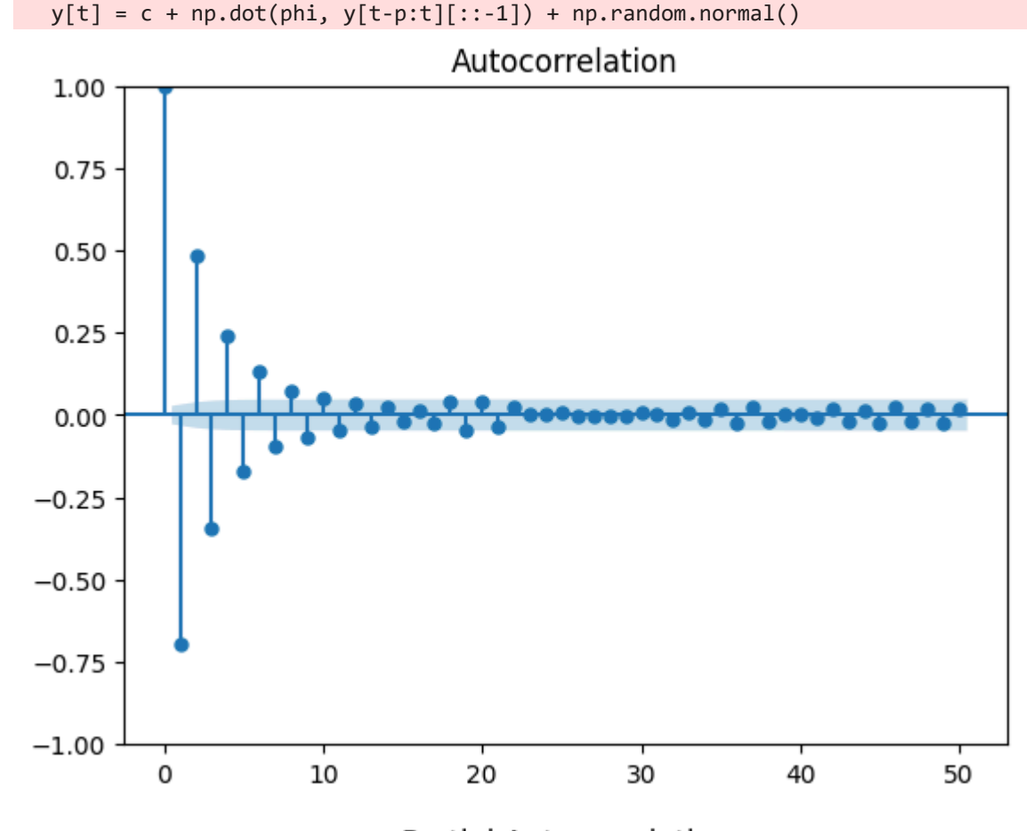


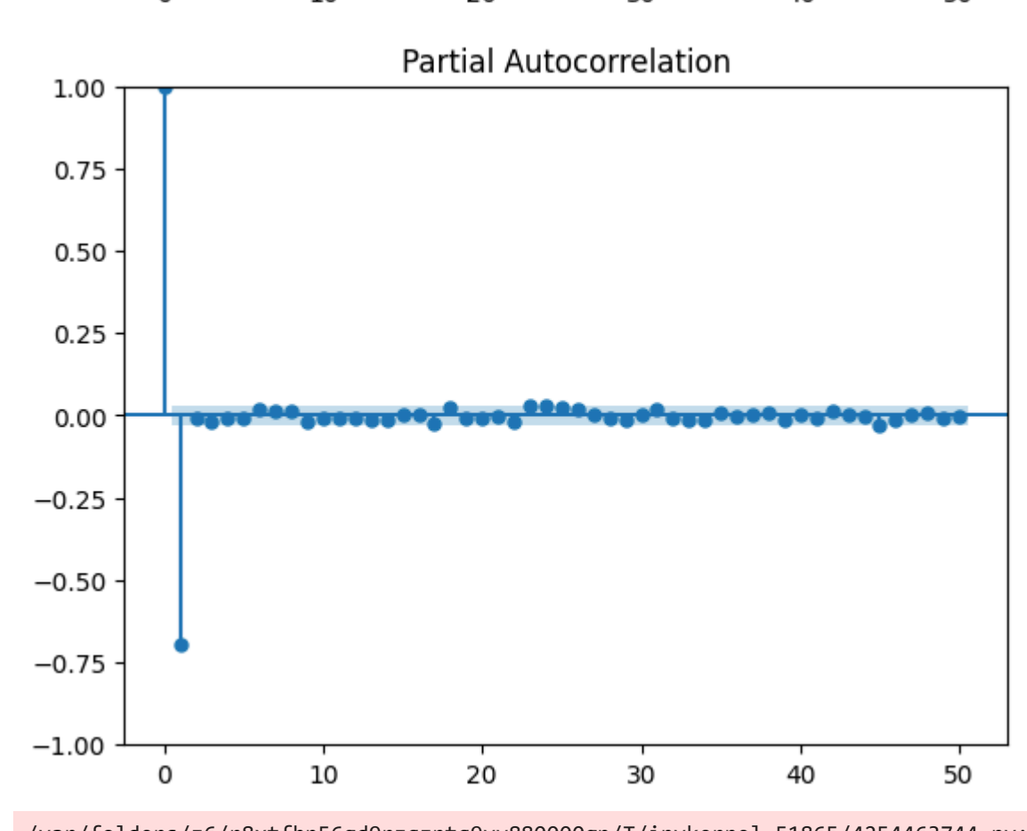


Comparing the structure of ACF/PACFs for AR and MA models(i will take the same values for AR and MA models for better understanding the differences)

In [68]:
```python
y = ar_model(1, -0.7, 18, 5000)
print("AR(1) model")
plot_acf(y, lags=50)
plt.show()
plot_pacf(y, lags=50)
plt.show()

y = ma_model(1, -0.7, 18, 5000)
print("MA(1) model")
plot_acf(y, lags=50)
plt.show()
plot_pacf(y, lags=50)
plt.show()
```

```
AR(1) model
/var/folders/z6/r8xtfhp56gd9rzqzptq9vv880000gn/T/ipykernel_51865/1885854207.py:9: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you extract a single element from your array before performing this operation. (Deprecated NumPy 1.25.)
  y[t] = c + np.dot(phi, y[t-p:t][::-1]) + np.random.normal()
```





```
/var/folders/z6/r8xtfhp56gd9rzqzptq9vv880000gn/T/ipykernel_51865/4254463744.py:5: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you extract a single element from your array before performing this operation. (Deprecated NumPy 1.25.)
  y[t] = c + e[t] + np.dot(theta, e[t-q:t][::-1])
MA(1) model
```