Question 1.1: What information is stored in the ZORI data? What does each row represent?

Each row in the Zillow Observed Rent Index (ZORI) dataset represents a region in the US (or the US as a whole). Every row contains data of a smoothed measure of the typical observed market rate rent across the given region (or the US) for every month from March 2015 to September 2022.

Question 2.2: Let's find some better parameters using trial and error. Try some combinations of p, d and q and see what is the result of the predictions. We will manually grade this question, and in order to recieve full credit, you must achieve an RMSE of less than 3.5 and a plot of your predictions vs the actual values must be shown.

Hint: Think about what does p, d, q mean in the ARIMA model.

When not worrying about overfitting, I can set up a simple grid search in a range of 0 through 6 (i.e. dependent on up to the last six months and stationary in up to the sixth difference) for each parameter to minimize the RMSE and found the optimal parameters to be `(p, d, q) = (0, 6, 6)` resulting in $RMSE = 0.375056$.

In [23]:
```
# define function for the model to iterate over
def ARIMA_test(train, test, order=(1, 1, 1),log=False):
    if log:
        train = np.log(train)
    ARIMAmodel = ARIMA(train, order = order)
    ARIMAmodel = ARIMAmodel.fit()
    y_pred_arima = ARIMAmodel.get_forecast(len(test.index)) # generate predictions
    y_pred_arima_df = y_pred_arima.conf_int(alpha = 0.05)
    y_pred_arima_df["Predictions"] = ARIMAmodel.predict(start = y_pred_df.index[0], end = y_pr
    y_pred_arima_df.index = test.index
    if log:
        y_pred_arima_df["Predictions"] = np.exp(y_pred_arima_df["Predictions"])
    y_pred_arima_out = y_pred_arima_df["Predictions"]

    arima_rmse = np.sqrt(mean_squared_error(test["CPI"].values, y_pred_arima_out))
    return arima_rmse, y_pred_arima_out
```

In [24]:
```
# create a grid for brute forcing
grid = [(p, d, q) for p in range(7) for d in range(7) for q in range(7)]

# brute force to find the best ARIMA model
results = []
for p, d, q in grid:
    try:
        rmse = ARIMA_test(train, test, order=(p, d, q))[0]
        results.append((p, d, q, rmse))
    except:
        continue

results_df = pd.DataFrame(results, columns = ['p', 'd', 'q', 'RMSE']).sort_values('RMSE')
results_df.head(5)
```

Out[24]:

	p	d	q	RMSE
48	0	6	6	0.375056
97	1	6	6	0.431818
145	2	6	5	0.435312
290	5	6	5	0.462546
339	6	6	5	0.529545

If the solution should be slightly more pragmatic and worry about overfitting, I can start looking at the meaning of the parameter for the integrated part. The `d` parameter is the number of times the data is differenced. As we are working with CPI and we assume the FED in the long run are successful with their target inflation of around $\sim 2\%$ YoY (though they usually look at PCE), the log-first difference would be stationary with a constant level around $1.02^{1/12} - 1 \approx 0.00165$ MoM. To remove this constant part, I set `d=2`.

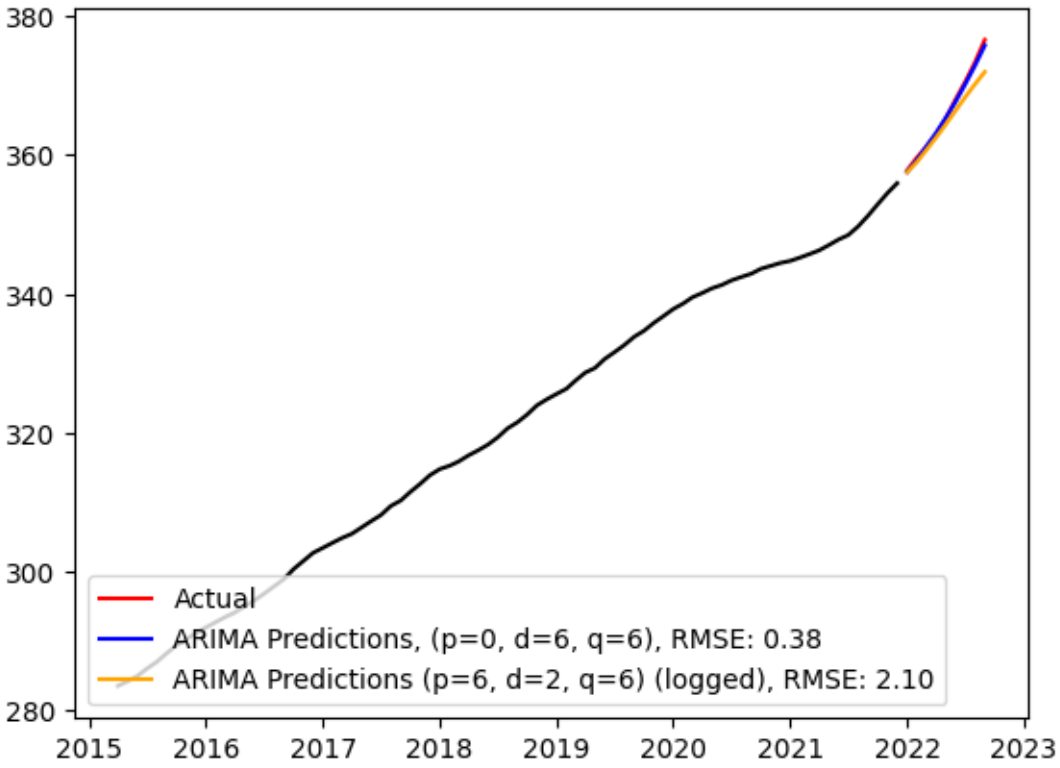
For the moving average part (shock persistency) and the autoregressive part (momentum), I will again seek to minimize the RMSE with up to six months of persistency. I found the optimal parameters to be `(p, d, q) = (6, 2, 6)`, resulting in $RMSE = 2.104195$.

In [25]:
```
# create a grid for brute forcing
grid = [(p, 2, q) for p in range(7) for q in range(7)]

# brute force to find the best ARIMA model
results = []
for p, d, q in grid:
    try:
        rmse = ARIMA_test(train, test, order=(p, d, q), log=True)[0]
        results.append((p, d, q, rmse))
    except:
        continue

results_df_ln = pd.DataFrame(results, columns = ['p', 'd', 'q', 'RMSE']).sort_values('RMSE')
results_df_ln.head(5)
```

Out[25]:

	p	d	q	RMSE
48	6	2	6	2.104195
5	0	2	5	3.252532
12	1	2	5	3.358089
19	2	2	5	3.359002
35	5	2	0	3.360853

In [26]:
```
# rerun model with best parameters and plot
opt = results_df.iloc[0, :3]
opt_ln = results_df_ln.iloc[0, :3]
ARIMA_opt = ARIMA_test(train, test, order=opt)
ARIMA_ln = ARIMA_test(train, test, order = opt_ln, log=True)

plt.plot(train, color = "black") # plot
plt.plot(test, color = "red", label = 'Actual')
plt.plot(ARIMA_opt[1], color='blue', label = f'ARIMA Predictions, (p={opt[0]:.0f}, d={opt[1]:.
plt.plot(ARIMA_ln[1], color='orange', label = f'ARIMA Predictions (p={opt_ln[0]:.0f}, d={opt_ln
plt.legend(loc='lower left');
```



The first model found solely by minimizing the RMSE closely resembles the actual data – however, it is likely overfitting. While not as close to the actual data, the second model is likely more robust and generalizes better. One could also consider an ARIMA(1,2,1) on the logged series would be best to avoid overfitting – this would yield $RMSE = 3.45$.

Question 3.2: Which time series is more volatile? What can be one possible explanation?

The Zillow Observed Rent Index is the most volatile of the time series. This is explained by the fact that the ZORI only considers rent and is thus more sensitive than a broader index like the CPI. This is especially true for interest rates, which affect the housing market more than the general economy.

Question 3.3: How can we interpret the findings above?

Hint: Read the docstring for the function `grangers_causation_matrix`.

Granger causality does not necessarily imply causality; rather, it is a statistical concept that explains whether one time series is useful in forecasting another. From the matrix above, we can see that both ZORI_x CPI_y and CPI_x and ZORI_y have values below the significance level, and thus, granger causes each other. The effect, however, is stronger from ZORI to CPI than the other way around.