Documentation for L06 Guided Solution  
  
To ensure a smooth installation, I individually added each package to the virtual environment, which streamlined the process. This approach proved effective, as Python 3.13.2 is too recent for the outdated TensorFlow version. Installing Python 3.11 was problematic due to conflicts with existing Python versions on my system, requiring me to manually remove all traces of Python. This involved navigating multiple directories and checking the registry editor.

A screenshot of a computer

AI-generated content may be incorrect.

The installation of Python 3.11 was particularly tricky, as it failed if any remnants of prior installations were detected. After extensive troubleshooting, I opted for Google Colab, which handled TensorFlow installation seamlessly. This experience taught me that Google Colab is an excellent tool for managing Python versions and package compatibility, and I plan to use it more frequently, even with a capable GPU.

the future over running locally.   
A black rectangular object with a white line

AI-generated content may be incorrect.

The process is much more seamless even though I have a good GPU I’ll consider collab more in the future for ease of access.

The Kaggle dataset was not publicly available, so the code given in the guided solution doesn’t work and it needs to be installed manually. Luckily a classmate helped me obtain the csv and I was able to upload it.

A graph showing a wave

AI-generated content may be incorrect.

There was only a very small outlier for a brief time going above 50C.

The date month and year format needed to be rearranged in the dataset.

A black rectangular object with orange and white lines

AI-generated content may be incorrect.API key obtained and predicted temps working  
A screenshot of a computer

AI-generated content may be incorrect.

Had to change the model  
A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

 **Hour Feature**:

* **Reason**: The 'hour' feature captures the **daily cyclical pattern of temperature** (diurnal variation). Temperatures typically rise during the day and drop at night, so knowing the time of day can help the model make more accurate predictions. The hour feature helps account for this regular pattern.

 **Temperature Lag Feature**:

* **Reason**: The 'temp\_lag\_1' feature leverages the **autocorrelation** in temperature data, meaning the temperature at one time is often strongly related to the temperature at the previous time step. By using the previous time step's temperature as a feature, the model can predict the current temperature more accurately. This feature is particularly useful when there is strong temporal dependence, as with weather data.

For phase 4 the predicted versus actual temps of the data was not unsatisfactory  
A screenshot of a computer

AI-generated content may be incorrect.

Which is very strange given the visual representation seems to be fairly accurate. I then decided to check different error scores  
A screen shot of a computer program

AI-generated content may be incorrect.

The r squared is very low, we want above .7

We would also want lower MAE and RMSE based on the visual data and the scores it seems like the model can generally follow the predicted temperature generally over time but not consistently at 1 specific time slot. There’s also a very big outlier in predicted temperature

Red and blue lines on a white background

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Explain the Rolling-Origin Cross-Validation (CV) Concept**

* **Definition**: The rolling-origin CV is a time series validation method where the training data is incrementally expanded, and the model is tested on the next "window" of data, called the test set. It simulates a scenario where the model is trained on historical data and makes forecasts for future time periods.
* **Robust Evaluation**: By using multiple splits (for instance, 5 splits), the model is tested on different time periods, providing insights into how it generalizes over time. This technique is more robust than a single train-test split, as it avoids overfitting and ensures the model performs well across various time windows. It also accounts for potential time-based trends and seasonality, which might not be captured in a single split.

**2. Cross-Validation Loop Summary**

* During each split, the model is trained on data up to a specific point in time (train period), and predictions are made for a subsequent time period (test period).
* This process is repeated for all the splits, ensuring that the model is trained and evaluated on various time periods.

**Metrics Calculation**: After all splits, you can compute metrics such as:

* **MAE (Mean Absolute Error)**: The average absolute difference between predicted and actual values.
* **MSE (Mean Squared Error)**: The average squared difference between predicted and actual values.
* **RMSE (Root Mean Squared Error)**: The square root of MSE, giving you the error in the same units as the data.

Phase 5:  


A screenshot of a computer program

AI-generated content may be incorrect.

After a lot of debugging and time all the epochs went through  
A graph with numbers and a line

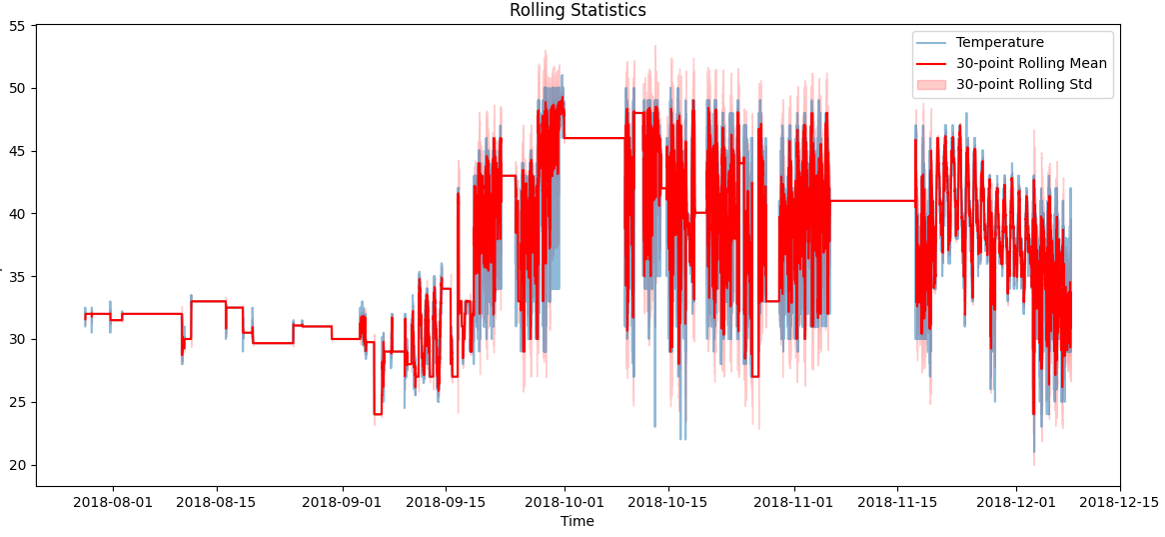
AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Plotting the differences was really weird.

That’s why alignment is failing — the time indexes are from completely different eras, so they can't be merged even with a 5-minute tolerance.  
I’ve pinpointed why the DataFrame is empty: the test set lives in **October 2018**, but the augmented forecast runs out to **December 2025**—so there are no overlapping ds values to join on.



From the results the synthetic data improved scores in MAE, MSE, and RMSE.