Week 3

Q1.

	Mean execution time	Standard deviation
Base	3.01 ms	733 µs
Iterrows Haversine	1.79 ms	258 µs
Apply	5.09 ms	205 ms
Vectorized (Pandas series)	5.98 ms	137 µs
Vectorized (Numpy arrays)	2.37 ms	72.6 µs
Cythonized loop (unaltered)	1.14 ms	223 µs
Redefined with C	676 µs	108 µs

Because of the small dataset, we can see that the base function using only looping is more effective than some other ways. The Apply method takes the longest to use because Pandas

The use of.apply() is not recommended for complex numerical calculations. Pandas is still slower than NumPy for tasks involving a lot of computation, although Vectorized (Pandas series) is quicker than the apply() approach. Cython's ability to efficiently compile Python loops into C makes the Cythonized loop significantly faster. C libraries, which had the lowest mean execution time, were finally used for increased efficiency.

Q2.

	Minimum execution time	Mean execution time	Maximum execution time
Base	3.09 s	3.63 s	3.97 s
Vectorized using matrix operations	24.4 µs	41.85 µs	449.5 µs
Vectorized using Apply	335.9 µs	552.9 µs	1293.6 µs

Before I used vectorized approaches, the brute-force methods worked fine in Python. Because R's structure is built for massive data analysis, execution speeds shown that it processed vectorized commands more quickly than any other R operation. Even Cythonic-optimized Python code for speed execution was surpassed by matrix operations in R.
R has the innate capacity to run high-performance calculations on enormous amounts of data, particularly when performing mathematical and statistical procedures.

Q3.

My experience with R has shown me that employing mathematical functions that are well-optimized can result in speedier execution times. However, because Python and Python perform calculations differently internally, this might not always be the case. The biggest issue I had with R was that it took a lot longer to write and debug the code since I kept running into bugs. Python, on the other hand, made coding considerably simpler because built-in functions handled a lot of the little nuances, enabling quicker development. In my view, Python is better for personal research or projects because of its flexibility and ease of use, while R is better for professional applications needing high-performance computations.

Q4.

The choice between Python and R as programming languages is influenced by a number of variables, such as run-time performance, coding simplicity, and user environments.

Python is a top technology because it offers a wide range of libraries needed for machine learning in addition to general-purpose programming tools.industrial norm. R's robust capabilities in statistical analysis and visualization make it ideal for research settings where these activities must be completed. R still dominates statistical data analysis, especially in medical research, while Python is widely used in the industrial sector. The needs of my data analysis jobs and my intended field of employment will determine which of Python and R I use. Although I work to improve my proficiency in both languages, Python is my preferred language because of its ease of use.