### **Task 5: Applying Custom NumPy Universal Functions**

**Objective & Rationale** NumPy's universal functions (ufuncs) are a core component of its performance. They allow for fast, element-wise operations on entire arrays without the overhead of Python loops. This application demonstrates how to define a custom ufunc and apply it in a user-friendly way.

**Our Approach** The application functions as a simple, powerful calculator.

1. **Function Definition**: A standard Python function, my\_ufunc\_func, is defined to perform a simple calculation (squaring a value and adding a constant).
2. **Vectorization**: np.vectorize() is used to convert this standard Python function into a ufunc. This allows the operation to be applied to every element of the array in a highly optimized manner.
3. **Interactive Display**: The UI provides two input fields for the array and a constant. Upon calculation, it displays both the input and output arrays in tables, along with the mathematical formula that was used.

**Key Takeaway** This project is an excellent demonstration of how to harness NumPy's powerful low-level optimizations. It provides a tangible example of a ufunc and its ability to perform fast, element-wise operations on entire datasets, a crucial skill for scientific and mathematical computing.

**Github-Repo:**<https://github.com/Kuldeep-Tapodhan/numpy-ufunc-django>

**Sample Output** For an input array of [1, 2, 3, 4, 5] and a constant of 10, the output would be:

**Output ScreenShots:**  
