Learn How to Use the MAKEARRAY Function in Excel: Step-by-Step Examples

Authored by

Mohammed looti

November 12, 2025

The introduction of dynamic array functionality has fundamentally transformed how complex calculations and data manipulations are managed in modern Excel. Standing out among these powerful new tools is the **MAKEARRAY** function. This innovative feature empowers users to dynamically construct an array structure based entirely on specified dimensions (rows and columns) and a custom calculation logic. Unlike traditional methods requiring manual data entry or cumbersome, legacy array formulas, **MAKEARRAY** integrates seamlessly with the LAMBDA function, allowing the content of every cell within the resulting array to be defined programmatically.

This functionality provides unparalleled flexibility for data modeling, advanced simulations, and the generation of complex lookup tables entirely on the fly. By mastering **MAKEARRAY**, developers and analysts can create exceptionally efficient and scalable spreadsheets that automatically adapt to changing input parameters without relying on extensive VBA code or the creation of numerous, distracting helper columns. The underlying principle is elegantly simple: you define the desired output size (the number of rows and columns) and then define the rule (the formula) that precisely determines the value at the intersection of each row and column index.

Deconstructing the MAKEARRAY Syntax and Essential Parameters

The **MAKEARRAY** function adheres to a highly intuitive structure, yet its capabilities for generating sophisticated arrays are extensive. To utilize this function effectively for custom data generation, it is crucial to achieve a deep understanding of its three primary arguments. This function essentially requires two steps: defining the structural boundaries of the array and then providing the calculation engine that will populate every point within those boundaries.

The basic and complete syntax for deployment is:

=MAKEARRAY(rows, cols, lambda(row, col))

Below is a detailed breakdown of each required argument, highlighting their specific roles in the array creation process:

rows: This argument dictates the total vertical extent of the resulting dynamic array. It must be provided as a positive integer, defining the number of rows the output will occupy.
cols: This argument defines the total horizontal extent of the resulting dynamic array. Similar to the rows argument, it must be specified as a positive integer, determining the number of columns in the output structure.
lambda: This is the critical functional component. It requires a valid LAMBDA function that must accept exactly two parameters--typically named **row** (or **r**) and **col** (or **c**)--and must return a single calculated value. The **row** and **col** parameters passed to the nested LAMBDA represent the current index (the physical position) within the array currently being generated. This positional

indexing is what allows the calculation to be based dynamically on the cell's location within the new array structure, making it possible to create highly structured mathematical results.

The following practical examples demonstrate how to leverage this positional indexing feature--the row and column numbers supplied by the LAMBDA function--to generate dynamic arrays for various real-world applications, ranging from simple mathematical tables to complex random data simulations necessary for statistical testing.

Example 1: Programmatically Generating an Array with Formula-Driven Values

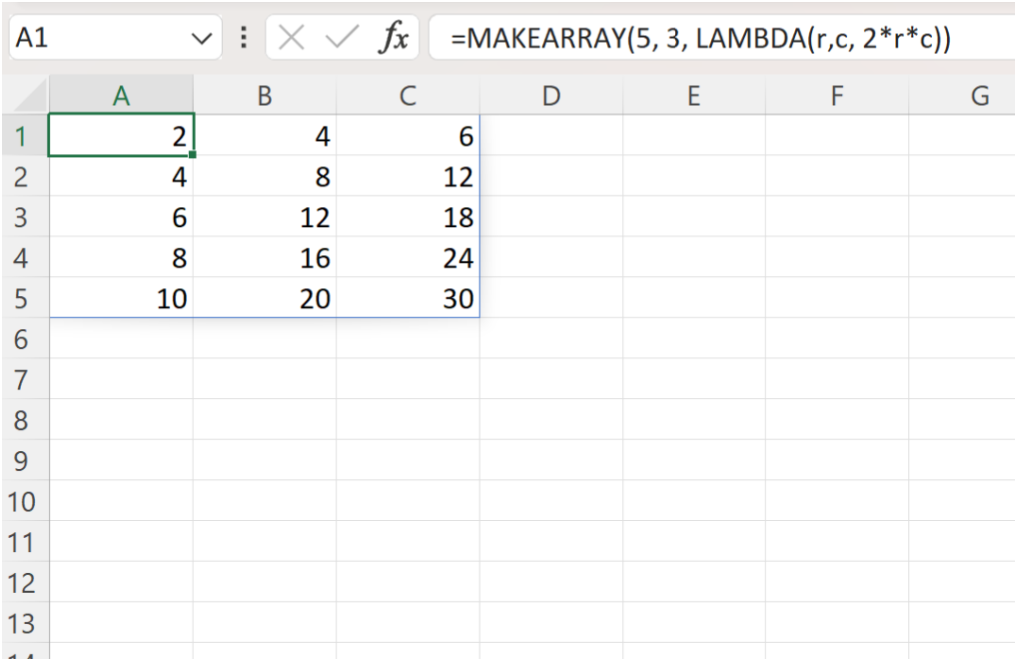
One of the most immediate and powerful applications of the **MAKEARRAY** function is the generation of structured datasets where the content of each individual cell is precisely determined by a formula that incorporates its row and column position. This technique is invaluable for rapidly constructing lookup tables, multiplication tables, or mathematical matrices where values must adhere to a clear, established pattern.

For this initial illustration, our objective is to construct an array consisting of **5** rows and **3** columns. The value placed in every cell of this output array will be derived using the formula: 2 multiplied by the row number (r) multiplied by the column number (c). We initiate this process by typing the following formula directly into cell **A1**:

=MAKEARRAY(5, 3, LAMBDA(r,c, 2*r*c))

Upon execution, **MAKEARRAY** systematically builds the defined 5x3 structure. It iterates through each position, applying the inner LAMBDA function to calculate the corresponding value based on the current row and column indices. The resulting array immediately spills across the required range, presenting a dynamically calculated matrix of results.

The following screenshot displays the output generated by this formula in a standard Excel workbook:



The resulting array successfully contains 5 rows and 3 columns. To confirm the calculation logic, we can manually check the values against the simple formula 2 * row number * column number:

For the cell at row 1 and column 1 (r=1, c=1): The calculation is 2 * 1 * 1, which results in the value **2**.
For the cell at row 2 and column 1 (r=2, c=1): The calculation is 2 * 2 * 1, which results in the value **4**.
For the cell at row 3 and column 1 (r=3, c=1): The calculation is 2 * 3 * 1, which results in the value **6**.
For the cell at row 5 and column 3 (r=5, c=3): The calculation is 2 * 5 * 3, which results in the value **30**.

This powerful methodology proves exceptionally valuable when there is a need to quickly generate and visualize complex relationships between two variables (represented by the rows and columns) without the tedious requirement of manually copying, pasting, and adjusting formulas across a large grid.

Example 2: Generating Dynamic Arrays of Random Categorical Text Data

The versatility of **MAKEARRAY** extends significantly beyond basic numerical calculations; it is equally proficient at generating arrays populated with non-numeric, randomized data. This capability is essential for simulation modeling, efficient testing of pivot tables, or rapidly generating extensive sample lists for demonstrations. By thoughtfully nesting functions such as **CHOOSE** and **RANDBETWEEN** within the central LAMBDA function, we can stipulate that the content of each

cell is selected randomly from a predefined, finite set of text options.

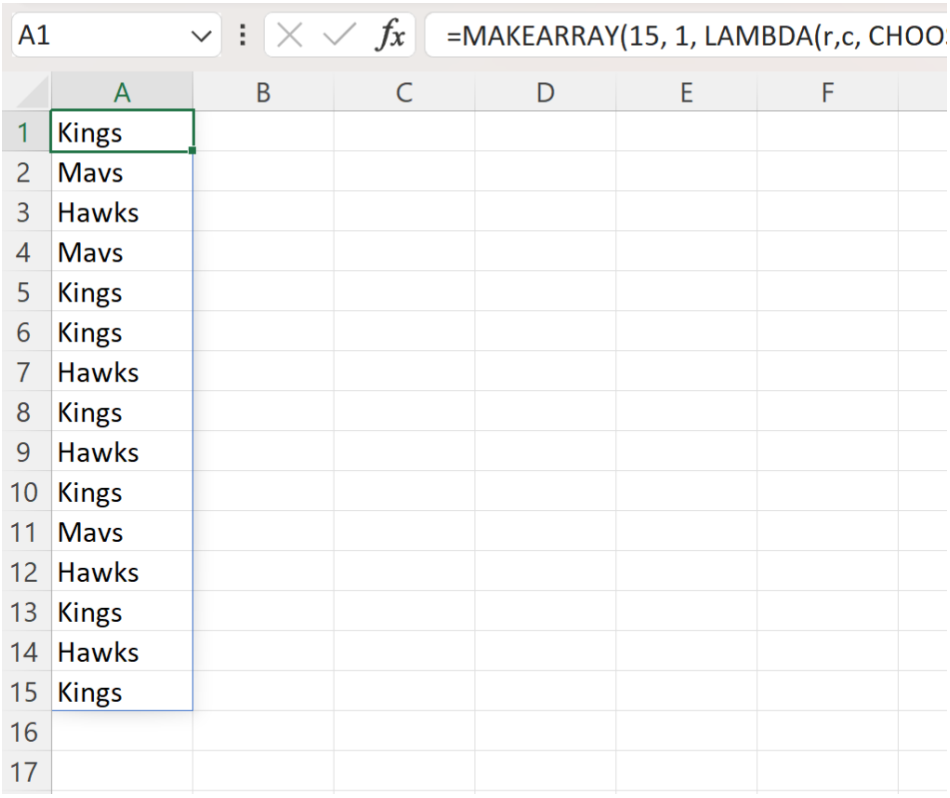
In this specific scenario, we will generate a single-column array composed of 15 rows. The rule is that each cell must randomly display one of three predefined team names: "Mavs," "Hawks," or "Kings." The core of the logic relies on **RANDBETWEEN(1,3)** to randomly select an index (1, 2, or 3). Subsequently, the **CHOOSE** function utilizes this randomly generated index to retrieve the corresponding text value from the defined list.

The complete formula required to achieve this dynamic text generation is entered into cell **A1**:

=MAKEARRAY(15,1,LAMBDA(r,c,CHOOSE(RANDBETWEEN(1,3),"Mavs","Hawks","Kings")))

It is important to note that while the LAMBDA function structurally requires the row (r) and column (c) parameters, these positional indices are not used in the calculation logic in this particular instance. The primary purpose of the LAMBDA here is simply to ensure that the inner calculation (the random selection process) is executed exactly 15 times, once for each cell in the array, thereby guaranteeing the generation of 15 independent random text values.

The following screenshot vividly illustrates the practical result of deploying this formula:



The resultant array accurately contains 15 rows and 1 column, precisely as specified by the

dimensions. Crucially, the value in each cell is independently and randomly selected from the three permissible text strings defined within the CHOOSE function. This powerful technique proves indispensable for quickly populating data models with controlled variability, making it effortless to generate realistic and manageable test data.

Example 3: Populating Arrays for Statistical Modeling with Random Numeric Integers

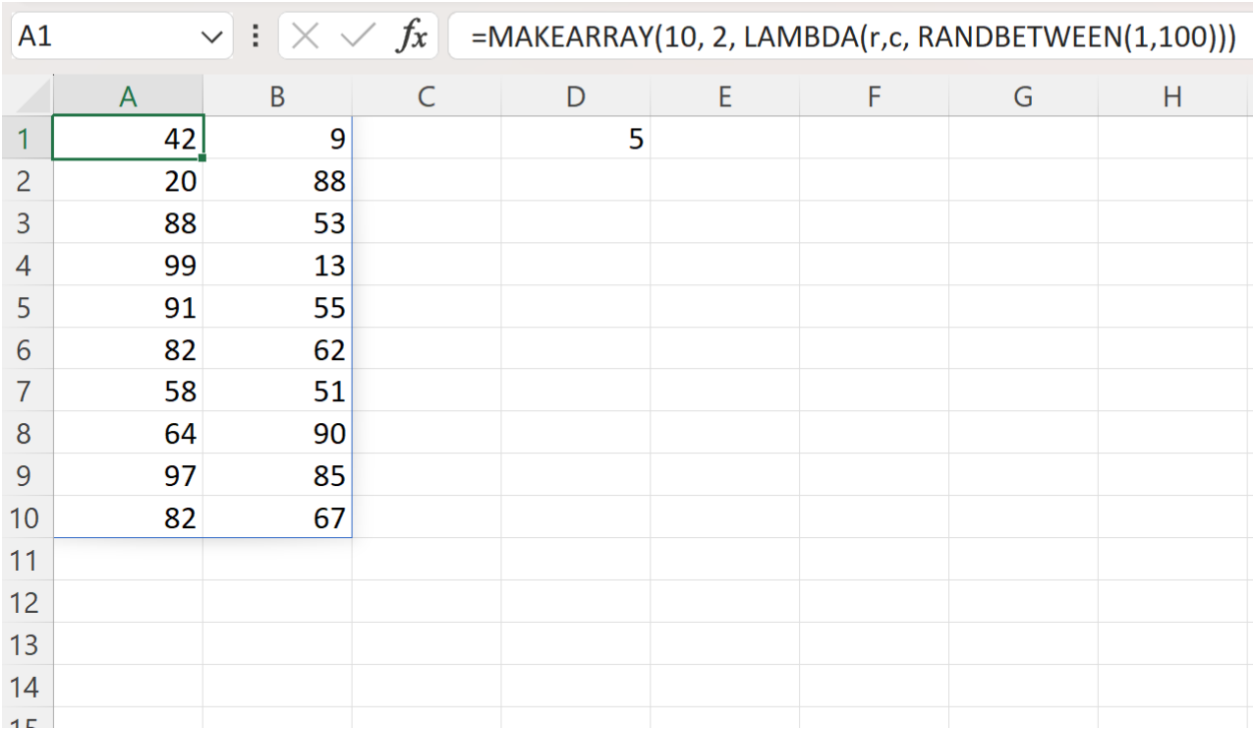
Generating large arrays of random numerical values is a fundamental requirement in statistical modeling, forecasting, and various simulation tasks within spreadsheets. Whether the goal is to execute Monte Carlo simulations or simply to obtain a large, randomized dataset for testing calculation speed and stress-testing models, **MAKEARRAY** offers the most efficient way to accomplish this using native Excel functions. This final example focuses on generating a two-dimensional array populated exclusively by random integers within a specified range.

We will configure the function to generate an array comprising **10** rows and **2** columns. The value for each of the 20 cells in this structure will be a randomly chosen integer between 1 and 100, inclusive. We achieve this purely random population by utilizing the **RANDBETWEEN** function directly within the nested LAMBDA. We enter the following concise formula into cell **A1**:

=MAKEARRAY(10, 2, LAMBDA(r,c, RANDBETWEEN(1,100)))

As observed in the previous example, the row and column indices (r and c) are structurally required by the LAMBDA definition but are functionally irrelevant to the calculation itself, as we are generating purely random numbers independent of their cell position. The core mechanism is that **MAKEARRAY** ensures the **RANDBETWEEN(1,100)** formula is executed exactly 20 times (10 rows multiplied by 2 columns), generating a unique random value for every slot.

The following screenshot clearly demonstrates the instantaneous output of this dynamic array generation:



This resulting array instantly populates 10 rows and 2 columns. Every value within the structure is an independently and randomly chosen integer between 1 and 100. It is essential to bear in mind that because **RANDBETWEEN** is classified as a volatile function, these values will automatically recalculate every time the workbook undergoes a change or recalculation event. This makes the function perfect for applications requiring a fresh dataset for ongoing simulations or iterative analysis.

Conclusion and Advanced Practical Considerations

The **MAKEARRAY** function stands as a highly versatile and absolutely essential utility for any user seeking to fully leverage the power of dynamic arrays in Microsoft Excel. By expertly combining control over array dimensions with the custom calculation logic enabled by the LAMBDA function, users gain the ability to generate highly structured, formula-driven, or randomly populated datasets instantly and efficiently.

Whether your requirements involve constructing a complex mathematical matrix based on positional indices (as shown in Example 1), generating robust simulation data using non-numeric categories (Example 2), or rapidly populating a statistical model with bounded random numbers (Example 3), **MAKEARRAY** dramatically reduces both the time investment and the inherent complexity traditionally associated with large-scale data generation in spreadsheets. Mastering the architecture and deployment of this function represents a significant leap forward toward achieving advanced proficiency in modern Excel data manipulation and modeling.

Additional Resources

For users interested in further exploring the expansive capabilities of dynamic arrays and integrating advanced Excel functions into complex data workflows, the following tutorials provide detailed explanations on how to perform other common operations and integrate these powerful new tools: