# A Beginner’s Guide to Finding Digits in SAS Strings Using the ANYDIGIT Function

Authored by

**Mohammed looti**

November 14, 2025

The SAS ANYDIGIT function is recognized as an indispensable utility for advanced character string manipulation within comprehensive data processing workflows. This highly optimized function is specifically engineered to quickly pinpoint the very first occurrence of any numerical digit embedded within a designated string, subsequently returning its precise, 1-based positional index. Such capability is foundational for critical data management tasks, encompassing rigorous data cleaning protocols, comprehensive validation procedures, and the essential process of parsing complex mixed alphanumeric identifiers where accurately isolating numerical components is the primary analytical objective.

Real-world datasets frequently present significant challenges because source information rarely adheres to perfectly standardized or structured formats. Data analysts routinely encounter records--such as unique employee IDs, detailed product codes, or complex addresses--that seamlessly integrate both alphabetical characters and numerical sequences. When tasked with analyzing these mixed data types, the process of accurately extracting, isolating, or validating the numerical components can become unnecessarily time-consuming and error-prone without specialized tools. The ANYDIGIT function dramatically simplifies this complexity by providing a direct, unambiguous, and highly efficient method to pinpoint the exact starting location of any numerical sequence within a given string, thereby ensuring superior efficiency and reliability throughout your coding efforts.

This expert guide is meticulously designed to walk you through the core syntax and demonstrate the practical, real-world applications of the ANYDIGIT function in SAS programming. We will commence by exploring its fundamental usage through clear, illustrative examples, demonstrating precisely how to determine the position of the first numerical digit across a diverse range of identifiers. Furthermore, we will delve into its optional parameters, specifically showcasing how these arguments can be utilized to significantly refine your search criteria, enabling highly specific data extraction and fulfilling advanced analytical requirements. By the conclusion of this tutorial, you will possess a robust and practical understanding of how to effectively leverage ANYDIGIT to enhance both your overall SAS programming efficiency and data handling precision.

## Understanding the Syntax of ANYDIGIT

The ANYDIGIT function in SAS is characterized by a remarkably concise and intuitive syntax, which facilitates its seamless integration into virtually any data manipulation routine you may encounter. Gaining a complete mastery of its structure and parameters is paramount for harnessing its full capabilities within complex data transformation processes. The fundamental structure of the function requires the specification of the primary target string, and it optionally permits an integer argument to precisely control the starting point from which the search for a digit should commence.

The function is formally defined using the following standard structure, highlighting its required and

optional components:

**ANYDIGIT(expression, )**

We can now systematically dissect each required and optional component that constitutes this essential syntax to ensure clarity:

**expression**: This component is **mandatory** and serves as the primary character string that the ANYDIGIT function will meticulously examine. The expression can be supplied as a direct string literal (enclosed in quotes), a variable that holds a string value, or even a complex operation that ultimately evaluates to a string. The function conducts a thorough scan of this input from left to right to isolate the position of the first numerical character it encounters.
**start (optional)**: This is an optional integer argument allowing programmers to specify the precise starting position for the search operation within the string expression. If this argument is entirely omitted, the function defaults to beginning the search from the very first character (position 1). Providing a value here means the search will begin at that specific character position. This is an incredibly useful feature when the initial segment of a string is known not to contain digits, or when the analysis specifically requires finding digits only after a certain defined prefix. The output is an integer representing the 1-based index position of the first numerical character found. If the function completes its scan without finding any digits in the specified range, it returns the value **0**.

The value returned by the ANYDIGIT function is consistently an integer. A non-zero, positive integer result signifies the 1-based index position where the very first numerical digit was successfully located. Conversely, a return value of **0** clearly indicates that no digit was detected within the entire searched character string or within the constrained search range defined by the optional start argument. This unambiguous distinction between a found position and an absent digit is vital, as it allows for simple and effective integration of the function's output into conditional logic statements and subsequent data transformation steps, greatly enhancing code robustness.

## Practical Application: Basic Usage of ANYDIGIT

To concretely demonstrate the powerful utility of the ANYDIGIT function, we will walk through a highly relevant and common data management scenario involving employee information records. Imagine a situation where employee identifiers consist of a heterogeneous mix of alphanumeric characters. Our specific objective is to accurately determine and record the position of the first numerical digit within each individual employee identifier, a critical step often required for standardized data parsing or ID standardization processes.

We initiate this demonstration by constructing a sample SAS data structure named **my_data** using the DATA step. This structure will contain two primary variables: **employeeID**, designated as a character variable storing unique alphanumeric identifiers, and **sales**, a numeric variable

representing fictional sales figures. We utilize the **datalines** statement to embed the sample data directly into our program, ensuring that the example is immediately reproducible and perfectly clear for tutorial purposes.

The following SAS code block demonstrates the creation and immediate viewing of this initial dataset, which provides the foundation for our subsequent analysis:

```
/*create dataset*/
data my_data;
input employeeID $ sales;
datalines;
54AAF 23
0009A 38
BC18B 40
09H30 12
04429 65
B1300 90
B1700 75
RRHHJ 35
0Y009 40
C6500 23
;
run;

/*view dataset*/
proc print data=my_data;
```

| Obs | employeeID | sales |
|---|---|---|
| 1 | 54AAF | 23 |
| 2 | 0009A | 38 |
| 3 | BC18B | 40 |
| 4 | 09H30 | 12 |
| 5 | 04429 | 65 |
| 6 | B1300 | 90 |
| 7 | B1700 | 75 |
| 8 | RRHHJ | 35 |
| 9 | 0Y009 | 40 |
| 10 | C6500 | 23 |

With our essential sample data successfully established, the next logical step is the direct application of the ANYDIGIT function. We integrate this function within a new DATA step to generate a subsequent dataset, which we appropriately name **new_data**. Within this new structure, we introduce a calculated column named **firstDigit**. This column is dynamically populated by the result of applying the ANYDIGIT function directly to the **employeeID** column. This simple yet powerful methodology ensures that we can clearly and immediately observe the calculated 1-based position of the first digit for every single employee identifier present in our sample data, providing immediate insight into the data structure.

```
/*create new dataset*/
data new_data;
set my_data;
firstDigit = anydigit(employeeID);
run;

/*view new dataset*/
proc print data=new_data;
```

| Obs | employeeID | sales | firstDigit |
|---|---|---|---|
| 1 | 54AAF | 23 | 1 |
| 2 | 0009A | 38 | 1 |
| 3 | BC18B | 40 | 3 |
| 4 | 09H30 | 12 | 1 |
| 5 | 04429 | 65 | 1 |
| 6 | B1300 | 90 | 2 |
| 7 | B1700 | 75 | 2 |
| 8 | RRHHJ | 35 | 0 |
| 9 | 0Y009 | 40 | 1 |
| 10 | C6500 | 23 | 2 |

## Interpreting the Results from Basic ANYDIGIT Usage

Following the successful execution of the SAS code, the resulting **new_data** structure now incorporates a crucial new column designated **firstDigit**. This variable diligently documents the 1-based index position of the first numerical character encountered within each corresponding **employeeID** character string. The output table clearly demonstrates the speed and efficiency with which the ANYDIGIT function processes and analyzes the diverse alphanumeric patterns present in the source data.

To truly appreciate the function's precise behavior and its commitment to the 1-based index, let us analyze several specific rows from our output to understand how the **firstDigit** column is populated based on various input strings:

For the **employeeID** "54AAF", the leading digit is '5', which is located precisely at position **1**. Consequently, the value of **firstDigit** is 1.
Similarly, when processing "0009A", the first digit is '0', also occupying position **1**. Therefore, **firstDigit** is recorded as 1.
Consider the identifier "BC18B". The initial letters 'B' and 'C' hold positions 1 and 2. The first numerical character, '1', is thus discovered at position **3**. Accordingly, **firstDigit** is 3.
In the distinct case of "RRHHJ", the input character string contains no numerical digits whatsoever. As mandated by its definition, the ANYDIGIT function returns **0**, explicitly signaling that no digit was successfully located within the entire string.

This detailed analysis underscores the function's exceptional ability to accurately and reliably pinpoint the initial numerical character, irrespective of its position or the surrounding alphabetical

characters within the string. Critically, the explicit return of **0** for strings lacking any numerical content serves as an invaluable feature for data quality assurance and validation routines. This zero return provides a clear, actionable flag for identifying and managing records that deviate from expected numerical inclusion patterns, dramatically streamlining data validation processes.

## Advanced Usage: Employing the Start Argument

Although the fundamental application of the ANYDIGIT function provides considerable utility, its optional **start** argument introduces a level of flexibility and precision that is essential for advanced data processing tasks. This argument grants the user the ability to define a specific starting point for the search operation, effectively allowing the programmer to bypass or ignore initial characters of a string and concentrate exclusively on a more relevant segment of the data. This targeted searching capability proves particularly indispensable when dealing with structured data where a known prefix must be systematically excluded from the search scope.

Consider a practical scenario: if your **employeeID** column consistently begins with two fixed alphabetic characters--perhaps representing a regional or departmental code--and your analytical task requires finding the first numerical character **only** after this prefix, the **start** argument becomes truly non-negotiable. By meticulously setting the start position parameter to **3**, you explicitly instruct the SAS environment to commence the digit search from the third character onward, thereby efficiently skipping the first two prefix positions. This targeted search mechanism is vital for preventing misinterpretation of results and ensuring that the returned positional index is highly pertinent to your specific analytical or data extraction requirements.

We will now proceed to modify our prior SAS code example to fully incorporate and demonstrate the utility of the **start** argument. In this updated procedure, we explicitly instruct the ANYDIGIT function to search for the position of the first digit within the **employeeID** column, but this time, the entire search operation will be forced to commence strictly from position **3**. This deliberate modification will vividly illustrate how controlling the search's starting point fundamentally alters the resultant positional data, especially for identifiers that originally contained digits within the first two character slots.

```
/*create new dataset*/
data new_data;
set my_data;
firstDigit = anydigit(employeeID, 3);
run;

/*view new dataset*/
proc print data=new_data;
```

| Obs | employeeID | sales | firstDigit |
|---|---|---|---|
| 1 | 54AAF | 23 | 0 |
| 2 | 0009A | 38 | 3 |
| 3 | BC18B | 40 | 3 |
| 4 | 09H30 | 12 | 4 |
| 5 | 04429 | 65 | 3 |
| 6 | B1300 | 90 | 3 |
| 7 | B1700 | 75 | 3 |
| 8 | RRHHJ | 35 | 0 |
| 9 | 0Y009 | 40 | 3 |
| 10 | C6500 | 23 | 3 |

## Analyzing Output with the Start Argument

A careful examination of the output generated by our revised SAS code--specifically where the ANYDIGIT function utilizes the **start** argument set to **3**--reveals substantial and expected differences when compared to the initial results. This new output powerfully illustrates the control gained by restricting the search range. The **firstDigit** column now exclusively reflects the position of the first numerical character found **from the third character position onwards**.

It is critically important to observe that **employeeID** values which previously contained a digit in the first or second position (such as "54AAF" or "0009A") now uniformly display a value of **0** in the **firstDigit** column. This outcome is the direct consequence of the search being explicitly initiated at position **3**, which causes the function to deliberately ignore any digits that occurred prior to this specified starting point. Since no subsequent numerical characters are found from position 3 onwards in these particular strings, the function correctly and logically returns **0**, indicating no digit was found within the defined search scope.

Conversely, for character strings such as "BC18B", where the first numerical character ('1') already resides exactly at position 3, the resulting value remains **3**. This confirms that the function successfully identifies the **first** digit encountered within the **specified search range**. For the identifier "0Y009", the absolute first digit is at position 1, but because the search begins at 3, the function instead finds the next '0' at position 3, returning **3**. This precise behavior is fundamental for scenarios that demand highly granular control over pattern recognition and data extraction within specific segments of character strings, significantly enhancing data parsing and validation capabilities.

## Conclusion and Further Exploration

The ANYDIGIT function in SAS remains a fundamental and exceptionally effective tool for accurately pinpointing the position of the first numerical character within any given character string. As clearly demonstrated through our step-by-step examples, its streamlined syntax, coupled with the optional **start** argument, provides robust capabilities essential for advanced data cleaning, stringent validation, and complex string parsing tasks. Whether the requirement is to find the absolute earliest digit or to meticulously constrain the search to a specific segment of a string, ANYDIGIT consistently delivers a precise and highly efficient solution.

Achieving mastery over critical functions such as ANYDIGIT is absolutely vital for any SAS programmer who regularly interfaces with real-world data, which is frequently characterized by inconsistencies and mixed data formats. By enabling the accurate identification of numerical character positions, you empower yourself to construct far more reliable data transformation logic, precisely extract key identifiers, and ultimately elevate the overall quality and usability of your analytical data.

To continue advancing your SAS string manipulation expertise, we highly recommend exploring related character functions that build upon similar positional logic. Consider dedicating time to learning functions such as **ANYALPHA** (designed to locate the first alphabetic character), **ANYSPACE** (used to find the first blank space), or the versatile **FIND** function (used to locate specific substrings). By combining and integrating these powerful functions, you can unlock significantly more complex data processing capabilities, allowing you to effectively address and resolve a much broader spectrum of data challenges.

## Additional Resources

The following curated tutorials provide comprehensive explanations on how to effectively utilize other common and powerful character functions available in SAS programming: