

Google Apps Script Custom Sheet Formulas



| | |
|--|-----------|
| Table of Sample Data | 3 |
| Formula to calculate the average of the values in a given range | 3 |
| Sample table for the code with the given input values: | 4 |
| Concatenate two strings and capitalize the first letter | 5 |
| Table of sample data for the CONCAT_CAPITALIZE_FIRST function: | 6 |
| Calculate the factorial of a given number | 6 |
| Table of sample data for the code: | 7 |
| Count the number of occurrences of a given value in a range | 8 |
| Here is a table of sample data for the COUNT_VALUE function: | 9 |
| Calculate the distance between two sets of latitude and longitude coordinates | 9 |
| Sample data DISTANCE_BETWEEN_COORDINATES: | 11 |
| Check if a given string is a palindrome | 11 |
| IS_PALINDROME data sample | 12 |
| Age of a person based on their birthdate | 13 |
| CALCULATE_AGE sample data | 13 |
| Convert a number from decimal to binary | 14 |

| | |
|---|-----------|
| DECIMAL_TO_BINARY sample data | 15 |
| Random number between two given numbers | 15 |
| Example code RANDOM_NUMBER | 16 |
| Sum of the squares of the numbers in a range | 17 |
| Example table with sample data for this code: | 17 |
| Capitalize the first letter of each word in a given string | 18 |
| Sample data table for the CAPITALIZE_WORDS function: | 19 |
| Calculate the factorial of a given number | 19 |
| Sample data for FACTORIAL function | 20 |
| Extract the domain name from a given URL | 20 |
| Sample data table for GET_DOMAIN_NAME | 21 |
| String to title case | 22 |
| Sample data for TITLE_CASE exercise | 23 |
| Calculate the sum of the even numbers in a range | 23 |
| table of sample data for the SUM_OF_EVEN function: | 24 |
| Count the number of occurrences of a given substring in a string | 25 |
| table of sample data for the COUNT_SUBSTR function: | 25 |
| Convert a number to its Roman numeral equivalent | 26 |
| table of sample data for the TO_ROMAN function: | 27 |
| Return the last number of characters of a string | 28 |
| Code Summary: | 29 |
| Distance between two sets of coordinates | 29 |
| table of sample data for the DISTANCE function: | 30 |
| Calculate the factorial of a number | 31 |
| Sample Code FACTORIAL | 31 |
| Sum of a range of cells | 32 |
| Example dataset SUM_NUMBERS | 32 |
| Concatenate the values in a range of cells | 33 |
| Code explanation JOIN_RANGE | 34 |
| Find the first occurrence of a value in a range of cells | 35 |
| VLOOKUP code example | 36 |
| Extract the nth word from a string | 36 |
| NTH_WORD code explanation | 37 |
| Calculate the percentage change between two numbers | 38 |
| PERCENT_CHANGE code explanation | 38 |
| Distance between two points in 2D space | 39 |

| | |
|---|-----------|
| DISTANCE_2D code explanation | 40 |
| Count the number of cells in a range that meet a certain condition | 41 |
| Table of sample data for the COUNT_IF function: | 42 |
| Number in Roman numeral format to its decimal equivalent | 42 |
| ROMAN_TO_DECIMAL code explanation | 44 |

Table of Sample Data

| Name | Age | Gender | Occupation |
|------|-----|--------|--------------|
| John | 25 | Male | Engineer |
| Jane | 30 | Female | Doctor |
| Jack | 22 | Male | Student |
| Jill | 27 | Female | Lawyer |
| Jim | 35 | Male | Accountant |
| Jess | 28 | Female | Teacher |
| Joe | 24 | Male | Programmer |
| Jen | 33 | Female | Entrepreneur |

Formula to calculate the average of the values in a given range

Example: A formula to calculate the average of the values in a given range, excluding any values that are below a specified threshold.

```

/**
 * Calculates the average of values in a given range,
 * excluding any values that are below a specified
 * threshold.
 * @param {range} range The range of cells to calculate
 * the average for.
 * @param {number} threshold The minimum value that will
 * be included in the average calculation.
 * @return The average of the values in the range that
 * are greater than or equal to the specified threshold.
 */
function AVERAGE_ABOVE_THRESHOLD(range, threshold) {
  var sum = 0;
  var count = 0;

  for (var i = 0; i < range.length; i++) {
    if (range[i][0] >= threshold) {
      sum += range[i][0];
      count++;
    }
  }

  return sum / count;
}

```

Sample table for the code with the given input values:

| Range | Threshold | Expected Output |
|------------------|-----------|-----------------|
| 1, 2, 3, 4, 5, 6 | 3 | 4 |

| | | |
|----------------------|-----|------|
| 10, 20, 30, 40, 50 | 25 | 40 |
| 2, 4, 6, 8, 10 | 5 | 8 |
| 0, 5, 10, 15, 20, 25 | 12 | 17.5 |
| 100, 200, 300, 400 | 500 | 0 |

In this code, the range parameter represents a range of cells in a spreadsheet, while threshold is a numerical value representing the minimum threshold for including a cell's value in the average calculation. The code loops through each cell in the range and calculates the average of all values greater than or equal to the specified threshold, ignoring any values that are below the threshold. The expected output represents the average value of the cells that meet the condition.

Concatenate two strings and capitalize the first letter

Example: A formula to concatenate two strings and capitalize the first letter of the resulting string.

```
/**
 * Concatenates two strings and capitalizes the first
 * letter of the resulting string.
 * @param {string} str1 The first string to concatenate.
 * @param {string} str2 The second string to
 * concatenate.
 * @return The concatenated string with the first letter
 * capitalized.
 */
```

```
function CONCAT_CAPITALIZE_FIRST(str1, str2) {  
  var concatenated = str1.concat(str2);  
  var firstLetter =  
concatenated.charAt(0).toUpperCase();  
  var restOfString = concatenated.slice(1);  
  return firstLetter + restOfString;  
}
```

Table of sample data for the CONCAT_CAPITALIZE_FIRST function:

| str1 | str2 | Output |
|-----------|-----------|------------------|
| "hello" | "world" | "HelloWorld" |
| "Goodbye" | "cruelty" | "GoodbyeCruelty" |
| "python" | "script" | "PythonScript" |
| "Java" | "program" | "JavaProgram" |
| "happy" | "feet" | "HappyFeet" |

The CONCAT_CAPITALIZE_FIRST function concatenates two strings str1 and str2 and capitalizes the first letter of the resulting string. The function returns the concatenated string with the first letter capitalized.

Calculate the factorial of a given number

Example: A formula to calculate the factorial of a given number.

```
/**
 * Calculates the factorial of a given number.
 * @param {number} n The number to calculate the
 factorial of.
 * @return The factorial of the given number.
 */
function FACTORIAL(n) {
  if (n == 0) {
    return 1;
  } else {
    return n * FACTORIAL(n - 1);
  }
}
```

Table of sample data for the code:

| Input | Output |
|-------|--------|
| 0 | 1 |
| 1 | 1 |
| 2 | 2 |
| 3 | 6 |
| 4 | 24 |
| 5 | 120 |

Code summary:

This is a recursive function that calculates the factorial of a given number. The factorial of a number is the product of all positive

integers less than or equal to that number. For example, the factorial of 5 (written as 5!) is $5 \times 4 \times 3 \times 2 \times 1 = 120$. If the input is 0, the output is 1. Otherwise, the function multiplies the input by the factorial of (input - 1) and returns the result.

Count the number of occurrences of a given value in a range

Example: A formula to count the number of occurrences of a given value in a range.

```
/**
 * Counts the number of occurrences of a given value in
 * a range.
 * @param {range} range The range of cells to search for
 * the given value.
 * @param {string/number} value The value to search for
 * in the range.
 * @return The number of occurrences of the given value
 * in the range.
 */
function COUNT_VALUE(range, value) {
  var count = 0;

  for (var i = 0; i < range.length; i++) {
    if (range[i][0] == value) {
      count++;
    }
  }

  return count;
}
```


}

Here is a table of sample data for the COUNT_VALUE function:

| range | value | return |
|---|-------|--------|
| 1 2 3 4 5 | 3 | 1 |
| 5 4 3 2 1 | 5 | 1 |
| 2 2 2 2 2 | 2 | 5 |
| 1 1 1 1 1 | 2 | 0 |
| "foo" "bar" "baz" "foo" "foo" | "foo" | 3 |

The COUNT_VALUE function counts the number of occurrences of a given value in a range of cells. In the first example, the value 3 appears once in the range, so the function returns 1. In the second example, the value 5 appears once in the range. In the third example, the value 2 appears five times in the range. In the fourth example, the value 2 does not appear in the range, so the function returns 0. In the fifth example, the string "foo" appears three times in the range.

Calculate the distance between two sets of latitude and longitude coordinates

Example: A formula to calculate the distance between two sets of latitude and longitude coordinates.

```

/**
 * Calculates the distance between two sets of latitude
 and longitude coordinates.
 * @param {number} lat1 The latitude of the first
 coordinate.
 * @param {number} lon1 The longitude of the first
 coordinate.
 * @param {number} lat2 The latitude of the second
 coordinate.
 * @param {number} lon2 The longitude of the second
 coordinate.
 * @return The distance between the two coordinates in
 kilometers.
 */
function DISTANCE_BETWEEN_COORDINATES(lat1, lon1, lat2,
lon2) {
  var earthRadiusKm = 6371;
  var dLat = degreesToRadians(lat2 - lat1);
  var dLon = degreesToRadians(lon2 - lon1);
  var a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
          Math.cos(degreesToRadians(lat1)) *
Math.cos(degreesToRadians(lat2)) *
          Math.sin(dLon / 2) * Math.sin(dLon / 2);
  var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 -
a));
  var distance = earthRadiusKm * c;
  return distance;
}

function degreesToRadians(degrees) {
  return degrees * Math.PI / 180;
}

```

}

Sample data DISTANCE_BETWEEN_COORDINATES:

| latitude 1 | longitude 1 | latitude 2 | longitude 2 | distance |
|------------|-------------|------------|-------------|----------|
| 37.7749 | -122.4194 | 40.7128 | -74.0060 | 4136.93 |
| 51.5074 | -0.1278 | 48.8566 | 2.3522 | 343.93 |
| -33.8688 | 151.2093 | 22.3193 | 114.1694 | 7286.57 |
| 35.6895 | 139.6917 | 37.5665 | 126.9780 | 1166.32 |

Note: The distance is in kilometers, and the coordinates are in decimal degrees format.

This function uses the Haversine formula to calculate the great-circle distance between two points on a sphere (in this case, the Earth) given their latitude and longitude coordinates. The function takes four parameters: lat1 and lon1 are the latitude and longitude of the first point, and lat2 and lon2 are the latitude and longitude of the second point. The function returns the distance between the two points in kilometers.

The degreesToRadians function is a helper function that converts degrees to radians, which is necessary for use in the Haversine formula.

Check if a given string is a palindrome

Example: A formula to check if a given string is a palindrome.

```
/**
```

```

* Checks if a given string is a palindrome.
* @param {string} str The string to check.
* @return True if the string is a palindrome, false
otherwise.
*/
function IS_PALINDROME(str) {
  str = str.toLowerCase().replace(/[^a-z0-9]/g, '');
  for (var i = 0; i < str.length/2; i++) {
    if (str[i] !== str[str.length - 1 - i]) {
      return false;
    }
  }
  return true;
}

```

IS_PALINDROME data sample

| Input | Output |
|--------------------------------|--------|
| "racecar" | true |
| "Level" | true |
| "A man a plan a canal Panama" | true |
| "hello" | false |
| "12321" | true |
| "Was it a car or a cat I saw?" | true |

| | |
|-------------------|-------|
| "No 'x' in Nixon" | true |
| "Palindrome" | false |

Age of a person based on their birthdate

Example: A formula to calculate the age of a person based on their birthdate.

provide a table of sample data for this code in sheets with code summary

```
/**
 * Calculates the age of a person based on their
 * birthdate.
 * @param {date} birthdate The person's birthdate.
 * @return The person's age in years.
 */
function CALCULATE_AGE(birthdate) {
  var ageDifMs = Date.now() - birthdate.getTime();
  var ageDate = new Date(ageDifMs);
  return Math.abs(ageDate.getUTCFullYear() - 1970);
}
```

CALCULATE_AGE sample data

Sample data table:

| birthdate | age |
|------------|-----|
| 1990-01-01 | 32 |
| 1985-03-15 | 36 |

| | |
|------------|----|
| 1975-12-31 | 46 |
| 2005-07-10 | 16 |
| 1999-11-22 | 22 |

Code summary:

The `CALCULATE_AGE` function calculates the age of a person based on their birthdate. It takes a birthdate parameter which is a `Date` object representing the person's birthdate, and returns the person's age in years as an integer. The function first calculates the difference between the current date and the birthdate in milliseconds using the `Date.now()` and `getTime()` methods. It then creates a new `Date` object from the age difference in milliseconds and uses the `getUTCFullYear()` method to extract the person's age in years. The `Math.abs()` function is used to ensure that the age is always positive.

Convert a number from decimal to binary

Example: A formula to convert a number from decimal to binary.

```
/**
 * Converts a number from decimal to binary.
 * @param {number} decimal The decimal number to
 * convert.
 * @return The binary representation of the decimal
 * number.
 */
function DECIMAL_TO_BINARY(decimal) {
  return decimal.toString(2);
}
```

DECIMAL_TO_BINARY sample data

| Sample Input | Expected Output |
|--------------|-----------------|
| 0 | 0 |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |

Note: The expected output is the binary representation of the input decimal number.

Random number between two given numbers

Example: A formula to generate a random number between two given numbers.

```
/**
 * Generates a random number between two given numbers.
 * @param {number} min The minimum possible value of the
 random number.
 * @param {number} max The maximum possible value of the
 random number.
 * @return A random number between min and max.
 */
function RANDOM_NUMBER(min, max) {
  return Math.floor(Math.random() * (max - min + 1)) +
min;
}
```

Example code RANDOM_NUMBER

| min | max | Result |
|-----|-----|--------|
| 0 | 10 | 3 |
| 1 | 6 | 4 |
| 10 | 20 | 12 |
| -5 | 5 | -1 |
| 50 | 100 | 72 |

Sum of the squares of the numbers in a range

Example: A formula to calculate the sum of the squares of the numbers in a range.

provide a table of sample data for this code in sheets with code summary

```
/**
 * Calculates the sum of the squares of the numbers in a
 * range.
 * @param {range} range The range of numbers to square
 * and sum.
 * @return The sum of the squares of the numbers in the
 * range.
 */
function SUM_OF_SQUARES(range) {
  var sum = 0;
  for (var i = 0; i < range.length; i++) {
    sum += range[i][0] * range[i][0];
  }
  return sum;
}
```

Example table with sample data for this code:

| Range |
|-------|
| 1 |
| 2 |
| 3 |
| 4 |

5

SUM_OF_SQUARES(Range)

55

Code summary:

The function takes in a range of numbers and calculates the sum of the squares of those numbers. It uses a for loop to iterate over the range, and adds the square of each number to a running sum. Finally, it returns the total sum of squares.

Capitalize the first letter of each word in a given string

Example: A formula to capitalize the first letter of each word in a given string.

```
/**
 * Capitalizes the first letter of each word in a given
 * string.
 * @param {string} str The string to capitalize.
 * @return The capitalized string.
 */
function CAPITALIZE_WORDS(str) {
  var words = str.split(' ');
  for (var i = 0; i < words.length; i++) {
    words[i] = words[i].charAt(0).toUpperCase() +
words[i].slice(1);
  }
  return words.join(' ');
}
```

}

Sample data table for the CAPITALIZE_WORDS function:

| Input | Output |
|-------------------------|-------------------------|
| "hello world" | "Hello World" |
| "how are you" | "How Are You" |
| "this is a test" | "This Is A Test" |
| "JavaScript is awesome" | "JavaScript Is Awesome" |

Summary:

This function takes a string as input and capitalizes the first letter of each word in the string. It achieves this by first splitting the string into an array of words, then looping through each word and using the `charAt` and `toUpperCase` methods to capitalize the first letter. Finally, the function joins the words back together into a single string and returns the result.

Calculate the factorial of a given number

Example: A formula to calculate the factorial of a given number.

```
/**
 * Calculates the factorial of a given number.
 * @param {number} n The number to calculate the
 * factorial of.
 * @return The factorial of n.
 */
```

```
function FACTORIAL(n) {  
  if (n === 0 || n === 1) {  
    return 1;  
  }  
  return n * FACTORIAL(n - 1);  
}
```

Sample data for FACTORIAL function

| Input | Output |
|--------------|--------|
| FACTORIAL(0) | 1 |
| FACTORIAL(1) | 1 |
| FACTORIAL(2) | 2 |
| FACTORIAL(3) | 6 |
| FACTORIAL(4) | 24 |
| FACTORIAL(5) | 120 |
| FACTORIAL(6) | 720 |

Extract the domain name from a given URL

Example: A formula to extract the domain name from a given URL.

```
/**  
 * Extracts the domain name from a given URL.
```

```

* @param {string} url The URL to extract the domain
name from.
* @return The domain name.
*/
function GET_DOMAIN_NAME(url) {
  var domain = '';
  var matches =
url.match(/^https?:\/\/\:\/\/([\^\/?#]+)(?:[\/?#]|$)/i);
  if (matches && matches[1]) {
    domain = matches[1];
  }
  return domain;
}

```

Sample data table for GET_DOMAIN_NAME

| Input (url) | Expected Output (domain name) |
|---|---|
| https://www.example.com | www.example.com |
| http://example.com/index.html | example.com |
| https://subdomain.example.org/path?query=string | subdomain.example.org |
| http://localhost:3000/ | localhost |

| | |
|---|--|
| https://www.example.co.uk/ | www.example.co.uk |
| ftp://ftp.example.net/files/ | ftp.example.net |
| https://www.example.com/about-us#team | www.example.com |
| https://www.google.com/search?q=google&oq=google&aqs=chrome..69i57j46i131i433j0i131i433j69i61l3.3434j1j7&sourceid=chrome&ie=UTF-8 | www.google.com |

String to title case

Example: A formula to convert a string to title case.

provide a table of sample data for this code in sheets with code summary

```
/**
 * Converts a string to title case.
 * @param {string} str The string to convert.
 * @return The title-cased string.
 */
function TITLE_CASE(str) {
  var words = str.toLowerCase().split(' ');
  for (var i = 0; i < words.length; i++) {
    words[i] = words[i].charAt(0).toUpperCase() +
words[i].slice(1);
  }
  return words.join(' ');
}
```

Sample data for TITLE_CASE exercise

Sample Data Table:

| Input | Output |
|-------------------------|-------------------------|
| "hello world" | "Hello World" |
| "THE QUICK BROWN FOX" | "The Quick Brown Fox" |
| "cAn YoU cOnVeRt ThIs?" | "Can You Convert This?" |
| "tEStIng tiTle caSE" | "Testing Title Case" |
| "ALL CAPS" | "All Caps" |

Code Summary: The TITLE_CASE function takes a string input and converts it to title case format where the first letter of each word in the string is capitalized. The function uses `toLowerCase()` to convert the input string to lowercase and `split()` to create an array of words from the string. It then iterates through each word in the array, capitalizes the first letter using `charAt()` and `toUpperCase()`, and then concatenates the rest of the word using `slice()`. Finally, the function returns the title-cased string using `join()` to combine the array of words back into a single string with spaces in between.

Calculate the sum of the even numbers in a range

Example: A formula to calculate the sum of the even numbers in a range.

```
/**
```

```

* Calculates the sum of the even numbers in a range.
* @param {range} range The range of numbers to sum.
* @return The sum of the even numbers in the range.
*/
function SUM_OF_EVEN(range) {
  var sum = 0;
  for (var i = 0; i < range.length; i++) {
    if (range[i][0] % 2 === 0) {
      sum += range[i][0];
    }
  }
  return sum;
}

```

table of sample data for the SUM_OF_EVEN function:

| Range | Expected Output |
|------------|-----------------|
| [[1], [2]] | 2 |
| [[3], [4]] | 4 |
| [[1], [3]] | 0 |
| [[2], [4]] | 6 |
| [[], []] | 0 |

Code Summary:

The SUM_OF_EVEN function takes a range of numbers as input and calculates the sum of the even numbers in that range. The input range is expected to be a two-dimensional array, where each element is itself an array containing a single number. The

function iterates through the range and checks each number to see if it is even (i.e., divisible by 2). If the number is even, it is added to the running total sum. Finally, the function returns the sum of the even numbers in the range.

Count the number of occurrences of a given substring in a string

Example: A formula to count the number of occurrences of a given substring in a string.

```
/**
 * Counts the number of occurrences of a given substring
 * in a string.
 * @param {string} str The string to search.
 * @param {string} substr The substring to count.
 * @return The number of occurrences of substr in str.
 */
function COUNT_SUBSTR(str, substr) {
  var count = 0;
  var pos = str.indexOf(substr);
  while (pos !== -1) {
    count++;
    pos = str.indexOf(substr, pos + 1);
  }
  return count;
}
```

table of sample data for the COUNT_SUBSTR function:

| str | substr | Result |
|-----|--------|--------|
|-----|--------|--------|

| | | |
|---------------|----------|---|
| "hello world" | "l" | 3 |
| "banana" | "a" | 3 |
| "mississippi" | "ss" | 2 |
| "javascript" | "script" | 1 |
| "apple" | "banana" | 0 |

Convert a number to its Roman numeral equivalent

Example: A formula to convert a number to its Roman numeral equivalent.

```
/**
 * Converts a number to its Roman numeral equivalent.
 * @param {number} num The number to convert.
 * @return The Roman numeral equivalent of num.
 */
function TO_ROMAN(num) {
  if (typeof num !== 'number') return NaN;
  var roman = '';
  var romanValues = {
    M: 1000,
    CM: 900,
    D: 500,
    CD: 400,
    C: 100,
    XC: 90,
    L: 50,
```

```
    XL: 40,  
    X: 10,  
    IX: 9,  
    V: 5,  
    IV: 4,  
    I: 1  
};  
for (var key in romanValues) {  
    while (num >= romanValues[key]) {  
        roman += key;  
        num -= romanValues[key];  
    }  
}  
return roman;  
}
```

table of sample data for the TO_ROMAN function:

| Input (num) | Output |
|-------------|--------|
| 1 | I |
| 3 | III |
| 4 | IV |
| 9 | IX |
| 10 | X |
| 14 | XIV |

| | |
|------|------|
| 50 | L |
| 99 | XCIX |
| 100 | C |
| 500 | D |
| 1000 | M |

Code summary: The TO_ROMAN function converts a given number to its equivalent Roman numeral. It does this by first checking if the input is a number and then using a dictionary of Roman numeral values to build up the Roman numeral string character by character.

Return the last number of characters of a string

Example: A formula to return the last N characters of a string.

```
/**
 * Returns the last N characters of a string.
 * @param {string} str The string to slice.
 * @param {number} n The number of characters to return.
 * @return The last N characters of str.
 */
function RIGHT(str, n) {
  if (n >= str.length) {
    return str;
  }
  return str.slice(str.length - n);
}
```

Code Summary:

The code defines a function called "RIGHT", which takes two parameters, a string "str" and a number "n". It returns the last "n" characters of the input string.

Sample Data:

| str | n | Output |
|-----------|----|---------|
| "hello" | 3 | "llo" |
| "world" | 5 | "world" |
| "foo bar" | 2 | "ar" |
| "test" | 10 | "test" |

Distance between two sets of coordinates

Example: A formula to calculate the distance between two sets of coordinates.

```
/**
 * Calculates the distance between two sets of
 * coordinates.
 * @param {number} lat1 The latitude of the first point.
 * @param {number} lon1 The longitude of the first
 * point.
 * @param {number} lat2 The latitude of the second
 * point.
 * @param {number} lon2 The longitude of the second
 * point.
```

```

* @param {string} [unit='K'] The unit of distance to
use (K for kilometers, M for miles).
* @return The distance between the two points.
*/
function DISTANCE(lat1, lon1, lat2, lon2, unit) {
  if (typeof unit === 'undefined') {
    unit = 'K';
  }
  var radLat1 = Math.PI * lat1 / 180;
  var radLat2 = Math.PI * lat2 / 180;
  var theta = lon1 - lon2;
  var radTheta = Math.PI * theta / 180;
  var dist = Math.sin(radLat1) * Math.sin(radLat2) +
Math.cos(radLat1) * Math.cos(radLat2) *
Math.cos(radTheta);
  dist = Math.acos(dist);
  dist = dist * 180 / Math.PI;
  dist = dist * 60 * 1.1515;
  if (unit === 'K') {
    dist = dist * 1.609344;
  }
  return dist;
}

```

table of sample data for the DISTANCE function:

| lat1 | lon1 | lat2 | lon2 | unit |
|-------|---------|-------|--------|------|
| 40.71 | -74.01 | 51.51 | -0.08 | K |
| 37.77 | -122.41 | 35.68 | 139.69 | K |

| | | | | |
|-------|---------|-------|--------|---|
| 51.50 | -0.12 | 48.86 | 2.35 | M |
| 48.86 | 2.35 | 51.50 | -0.12 | M |
| 34.05 | -118.24 | 40.71 | -74.01 | K |

It is a function that calculates the distance between two sets of coordinates. The input values (lat1, lon1, lat2, lon2, and unit) should be provided when calling the function.

Calculate the factorial of a number

Example: A formula to calculate the factorial of a number.

```
/**
 * Calculates the factorial of a number.
 * @param {number} n The number to calculate the
 * factorial of.
 * @return The factorial of n.
 */
function FACTORIAL(n) {
  if (n < 0 || n % 1 !== 0) return NaN;
  var result = 1;
  for (var i = 2; i <= n; i++) {
    result *= i;
  }
  return result;
}
```

Sample Code FACTORIAL

This code only takes in a single numeric argument and calculates its factorial using a loop.

Sum of a range of cells

Example: A formula to calculate the sum of a range of cells, ignoring any cells that contain non-numeric values.

```
/**
 * Calculates the sum of a range of cells, ignoring any
 * cells that contain non-numeric values.
 * @param {range} range The range of cells to sum.
 * @return The sum of the numeric values in the range.
 */
function SUM_NUMBERS(range) {
  var sum = 0;
  var values = range.getValues();
  for (var i = 0; i < values.length; i++) {
    for (var j = 0; j < values[i].length; j++) {
      if (typeof values[i][j] === 'number') {
        sum += values[i][j];
      }
    }
  }
  return sum;
}
```

Example dataset SUM_NUMBERS

Here's an example dataset you can use with this code:

| | A | B | C | D | E |
|---|-----|-----|-----|-----|-----|
| 1 | 10 | 20 | x | 30 | 40 |
| 2 | abc | def | ghi | 123 | 456 |

| | A | B | C | D | E |
|---|-----|-----|-----|-----|-----|
| 3 | 5.5 | 7.5 | 9.5 | 2.5 | 4.5 |
| 4 | 1 | | 2 | | 3 |
| 5 | 5 | 10 | 15 | 20 | 25 |

You can use the formula `=SUM_NUMBERS(A1:E5)` to calculate the sum of all the numeric values in the range A1:E5, which should result in 98.5.

Concatenate the values in a range of cells

Example: A formula to concatenate the values in a range of cells, separated by a given delimiter.

```
/**
 * Concatenates the values in a range of cells,
 * separated by a given delimiter.
 * @param {range} range The range of cells to
 * concatenate.
 * @param {string} [delimiter=','] The delimiter to use
 * between values.
 * @return The concatenated string.
 */
function JOIN_RANGE(range, delimiter) {
  if (typeof delimiter === 'undefined') {
    delimiter = ',';
  }
  var values = range.getValues();
  var result = '';
```

```

for (var i = 0; i < values.length; i++) {
  for (var j = 0; j < values[i].length; j++) {
    if (j > 0) {
      result += delimiter;
    }
    result += values[i][j];
  }
}
return result;
}

```

Code explanation JOIN_RANGE

To use the JOIN_RANGE function in Google Sheets, you can simply enter the function with the appropriate parameters in any cell:

```
=JOIN_RANGE(A1:A5, "|")
```

This will concatenate the values in cells A1 through A5 with the delimiter "|" and return the concatenated string.

If you do not specify a delimiter, the default delimiter of "," will be used:

```
=JOIN_RANGE(A1:A5)
```

This will concatenate the values in cells A1 through A5 with the default delimiter "," and return the concatenated string.

Note that you will need to have the script containing the JOIN_RANGE function added to your Sheets document for this to

work. You can do this by opening the script editor (Tools > Script editor), pasting in the code, and saving the script. Once the script is saved, you can use the JOIN_RANGE function in any cell in your Sheets document.

Find the first occurrence of a value in a range of cells

Example: A formula to find the first occurrence of a value in a range of cells, and return the corresponding value from another range.

```
/**
 * Finds the first occurrence of a value in a range of
 * cells, and returns the corresponding value from another
 * range.
 * @param {string} searchValue The value to search for.
 * @param {range} searchRange The range of cells to
 * search in.
 * @param {range} returnRange The range of cells to
 * return values from.
 * @return The first value in returnRange that
 * corresponds to searchValue in searchRange.
 */
function VLOOKUP(searchValue, searchRange, returnRange)
{
  var searchValues = searchRange.getValues().flat();
  var returnValues = returnRange.getValues().flat();
  var index = searchValues.indexOf(searchValue);
  if (index === -1) {
    return '';
  }
}
```

```
}  
  return returnValues[index];  
}
```

VLOOKUP code example

To use this function in Google Sheets, you can simply enter the function with the appropriate parameters in any cell:

```
=VLOOKUP("apple", A1:A10, B1:B10)
```

This will search the range A1:A10 for the value "apple" and return the corresponding value from the range B1:B10.

You can also reference cell ranges as parameters:

```
=VLOOKUP(A1, B1:B10, C1:C10)
```

This will search the range B1:B10 for the value in cell A1 and return the corresponding value from the range C1:C10.

Note that you will need to have the script containing the VLOOKUP function added to your Sheets document for this to work. You can do this by opening the script editor (Tools > Script editor), pasting in the code, and saving the script. Once the script is saved, you can use the VLOOKUP function in any cell in your Sheets document.

Extract the nth word from a string

Example: A formula to extract the nth word from a string.

```

/**
 * Extracts the nth word from a string.
 * @param {string} str The input string.
 * @param {number} n The index of the word to extract
 (1-based).
 * @return The nth word in the input string, or an empty
 string if n is out of range.
 */
function NTH_WORD(str, n) {
  var words = str.split(/\s+/);
  if (n < 1 || n > words.length) {
    return '';
  }
  return words[n - 1];
}

```

NTH_WORD code explanation

To use this function in Google Sheets, you can simply enter the function with the appropriate parameters in any cell:

```
=NTH_WORD("This is a sample sentence", 3)
```

This will return the third word in the sentence, which is "a".

You can also reference cell values as parameters:

```
=NTH_WORD(A1, B1)
```

This will return the nth word in the string specified in cell A1, where n is the value in cell B1.

Note that you will need to have the script containing the NTH_WORD function added to your Sheets document for this to work. You can do this by opening the script editor (Tools > Script editor), pasting in the code, and saving the script. Once the script is saved, you can use the NTH_WORD function in any cell in your Sheets document.

Calculate the percentage change between two numbers

Example: A formula to calculate the percentage change between two numbers.

```
/**
 * Calculates the percentage change between two numbers.
 * @param {number} oldVal The original value.
 * @param {number} newVal The new value.
 * @return The percentage change between oldVal and
 * newVal, expressed as a decimal.
 */
function PERCENT_CHANGE(oldVal, newVal) {
  if (oldVal === 0) {
    return newVal === 0 ? 0 : 1;
  }
  return (newVal - oldVal) / oldVal;
}
```

PERCENT_CHANGE code explanation

To use this function in Google Sheets, you can simply enter the function with the appropriate parameters in any cell:

```
=PERCENT_CHANGE(50, 75)
```

This will return the percentage change between the old value of 50 and the new value of 75, which is 0.5 or 50%.

You can also reference cell values as parameters:

```
=PERCENT_CHANGE(A1, B1)
```

This will return the percentage change between the old value in cell A1 and the new value in cell B1.

Note that you will need to have the script containing the PERCENT_CHANGE function added to your Sheets document for this to work. You can do this by opening the script editor (Tools > Script editor), pasting in the code, and saving the script. Once the script is saved, you can use the PERCENT_CHANGE function in any cell in your Sheets document.

Distance between two points in 2D space

Example: A formula to calculate the distance between two points in 2D space.

```
/**
```

```
* Calculates the distance between two points in 2D space.
```

```
* @param {number} x1 The x-coordinate of the first point.
```

```
* @param {number} y1 The y-coordinate of the first point.
```

```
* @param {number} x2 The x-coordinate of the second point.
```

```
* @param {number} y2 The y-coordinate of the second
point.
* @return The distance between the two points.
*/
function DISTANCE_2D(x1, y1, x2, y2) {
  var dx = x2 - x1;
  var dy = y2 - y1;
  return Math.sqrt(dx * dx + dy * dy);
}
```

DISTANCE_2D code explanation

To use this function in Google Sheets, you can simply enter the function with the appropriate parameters in any cell:

```
=DISTANCE_2D(1, 2, 4, 6)
```

This will return the distance between the points (1, 2) and (4, 6), which is approximately 4.472.

You can also reference cell values as parameters:

```
=DISTANCE_2D(A1, B1, C1, D1)
```

This will return the distance between the points specified by the values in cells A1, B1, C1, and D1.

Note that you will need to have the script containing the DISTANCE_2D function added to your Sheets document for this to work. You can do this by opening the script editor (Tools > Script editor), pasting in the code, and saving the script. Once the script

is saved, you can use the DISTANCE_2D function in any cell in your Sheets document.

Count the number of cells in a range that meet a certain condition

Example: A formula to count the number of cells in a range that meet a certain condition.

provide a table of sample data for this code in sheets with code summary

```
/**
 * Counts the number of cells in a range that meet a
 * certain condition.
 * @param {range} range The range of cells to count.
 * @param {string} condition The condition to test (e.g.
 * '>0').
 * @return The number of cells in the range that meet
 * the condition.
 */
function COUNT_IF(range, condition) {
  var values = range.getValues();
  var count = 0;
  for (var i = 0; i < values.length; i++) {
    for (var j = 0; j < values[i].length; j++) {
      if (eval(values[i][j] + condition)) {
        count++;
      }
    }
  }
  return count;
}
```

Table of sample data for the COUNT_IF function:

| Range | Condition | Result |
|-----------------------|------------|--------|
| 1, 2, 3, 4, 5 | '>3' | 2 |
| 1, 3, 5, 7, 9 | '>0' | 5 |
| 0, 0, 0, 0, 0 | '>0' | 0 |
| 'apple', 'banana' | ""banana"" | 1 |
| 'cat', 'dog', 'mouse' | '="cat"' | 1 |

The COUNT_IF function counts the number of cells in a range that meet a certain condition. The range parameter represents the range of cells to count, while the condition parameter specifies the condition to test. The function evaluates each cell in the range against the condition using eval(), and increments a counter if the condition is met. The function returns the total count of cells that meet the condition.

Number in Roman numeral format to its decimal equivalent

A formula to convert a number in Roman numeral format to its decimal equivalent.

```
/**
 * Converts a number in Roman numeral format to its
 * decimal equivalent.
 * @param {string} str The Roman numeral to convert.
 * @return The decimal equivalent of the Roman numeral,
 * or NaN if str is not a valid Roman numeral.
```

```
*/  
function ROMAN_TO_DECIMAL(str) {  
  var values = {  
    'M': 1000,  
    'D': 500,  
    'C': 100,  
    'L': 50,  
    'X': 10,  
    'V': 5,  
    'I': 1  
  };  
  var total = 0;  
  var lastValue = 0;  
  for (var i = str.length - 1; i >= 0; i--) {  
    var currentValue = values[str.charAt(i)];  
    if (currentValue >= lastValue) {  
      total += currentValue;  
    } else {  
      total -= currentValue;  
    }  
    lastValue = currentValue;  
  }  
  if (isNaN(total)) {  
    return NaN;  
  }  
  return total;  
}
```

ROMAN_TO_DECIMAL code explanation

This function takes a string `str` representing a number in Roman numeral format, and returns its decimal equivalent. The function first creates a lookup table values mapping each Roman numeral to its decimal value. It then loops through each character in `str` from right to left, adding or subtracting the corresponding decimal value based on whether the current value is greater than or less than the previous value. Finally, the function checks whether the result is a valid number, and returns it if so, or `NaN` if not.