Google Sheet Formulas

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

https://youtu.be/2QJtyU17RJ0

Range	**Threshold**	
1, 2, 3, 4, 5, 6,7	3	5
10, 20, 30, 40, 50	25	40
2, 4, 6, 8, 10	5	8
0, 5, 10, 15, 20, 25	200	0
100, 200, 300, 400	200	300

The given code defines an Apps Script function called AVE_Above_Threshold, which takes two parameters: range and threshold. This function calculates the average of the numbers in a given range that are above or equal to the specified threshold. Here's a step-by-step breakdown of what the code does:

1. Initialize two variables, sum and count, to keep track of the sum of numbers above the threshold and the count of numbers above the threshold, respectively.

Convert the range parameter into an array of values by performing the following steps:
a. Remove any leading or trailing whitespace from the range string using the trim() method.
b. Split the resulting string using a comma as the delimiter to separate individual values.

2. c. Store the resulting array in the arr variable.
3. Iterate over each element in the arr array using the forEach method and a callback function that takes a val parameter representing the current element.

Inside the callback function, convert the val into a number by performing the following steps:
a. Remove any leading or trailing whitespace from the val string using the trim() method.

4. b. Parse the resulting string into an integer using the parseInt() function, and store the result in the num variable.

Check if the num value is greater than or equal to the threshold value:
a. If the condition is true, increment the count variable by 1 to keep track of the count of numbers above the threshold.

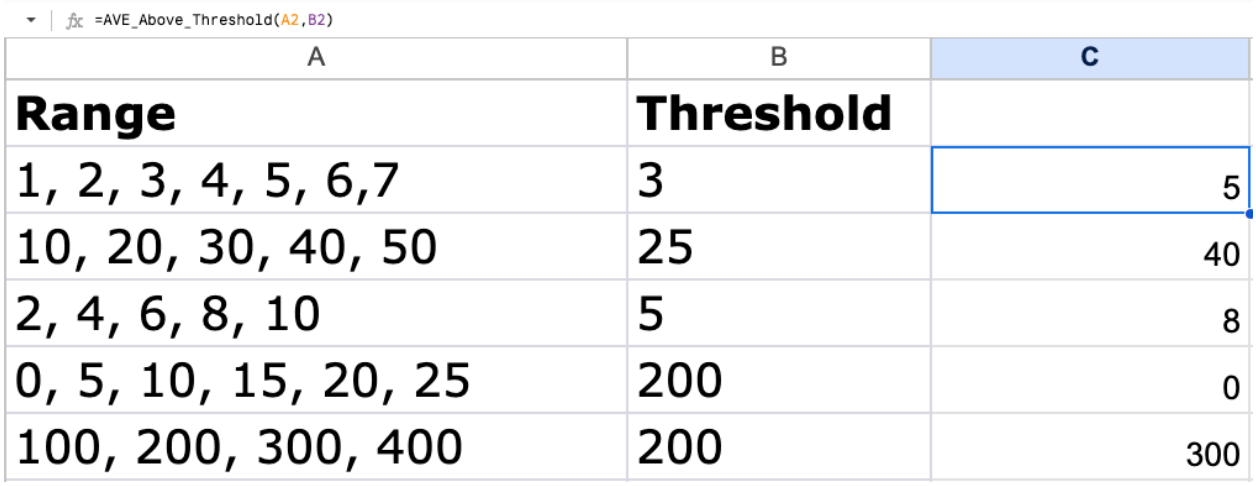
5. b. Add the num value to the sum variable to accumulate the sum of numbers above the threshold.

After the iteration is complete, check if the count variable is equal to zero:

a. If the condition is true, it means there were no numbers above the threshold, so the function returns 0.

6. b. If the condition is false, calculate the average by dividing the sum variable by the count variable, and return the result.

In summary, this function takes a string of comma-separated values (range) and a threshold value (threshold), and calculates the average of the numbers in the range that are above or equal to the threshold. If there are no numbers above the threshold, it returns 0.



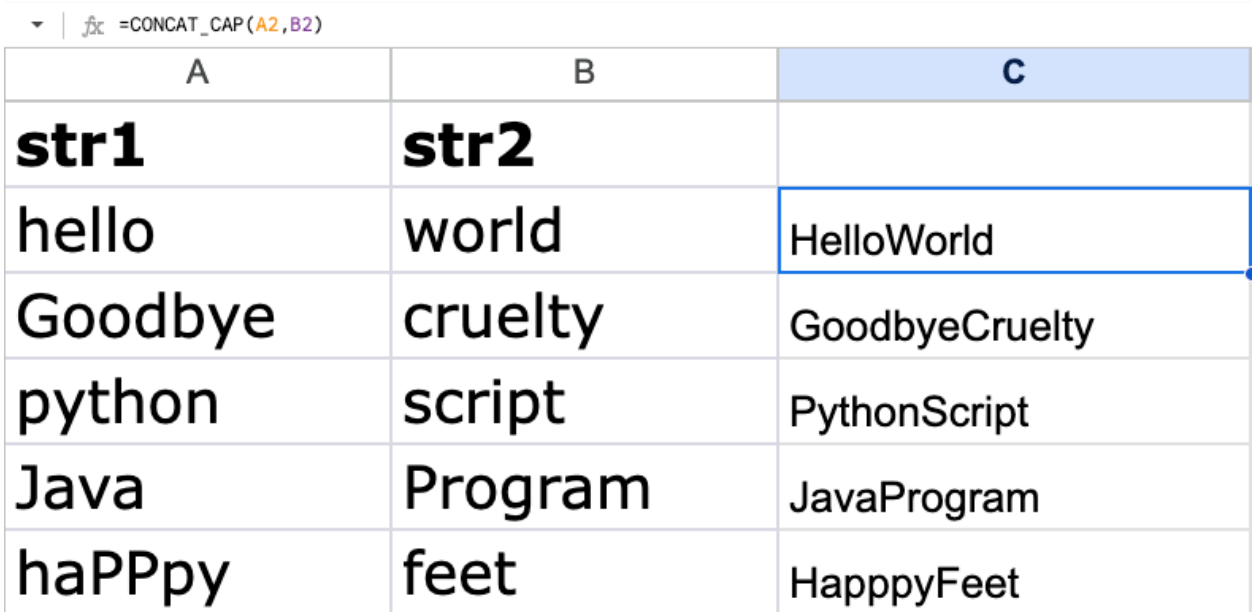
A	B	C
Range	**Threshold**	
1, 2, 3, 4, 5, 6,7	3	5
10, 20, 30, 40, 50	25	40
2, 4, 6, 8, 10	5	8
0, 5, 10, 15, 20, 25	200	0
100, 200, 300, 400	200	300

```
function AVE_Above_Threshold(range,threhold) {
 let sum = 0;
 let count = 0;
 const arr = range.trim().split(',');
 arr.forEach(val =>{
```

```
  let num = parseInt(val.trim());
  if(num >= threhold){
    count++;
    sum += num;
  }
 })
 return count == 0 ? 0 : sum/count;
}
```

Concatenate two strings and capitalize the first letter

https://youtu.be/PuMqOR7ambE




```
function CONCAT_CAP(str1,str2){
 const stra = upperMe(str1);
 const strb = upperMe(str2);
 return stra.concat(strb);
}
function CONCAT_CAP2(str1,str2){
 const stra = str1.chatAt(0).toUpperCase();
 const concatVal = str1.concat(str2);
 let firstLetter = concatVal.charAt(0).toUpperCase();
```

```
 let restVals = concatVal.slice(1).toLowerCase();
 return firstLetter + restVals;
}
function upperMe(val){
 Logger.log(val);
 return val.charAt(0).toUpperCase() + val.slice(1).toLowerCase();
}

function test(){
 const val = 'laurence';
 upperMe(val);
}
```

str1	**str2**	
hello	world	HelloWorld
Goodbye	cruelty	GoodbyeCruelty
python	script	PythonScript
Java	Program	JavaProgram
haPPpy	feet	HapppyFeet

The provided code consists of several functions that manipulate strings in different ways. Let's break down each function and its purpose:

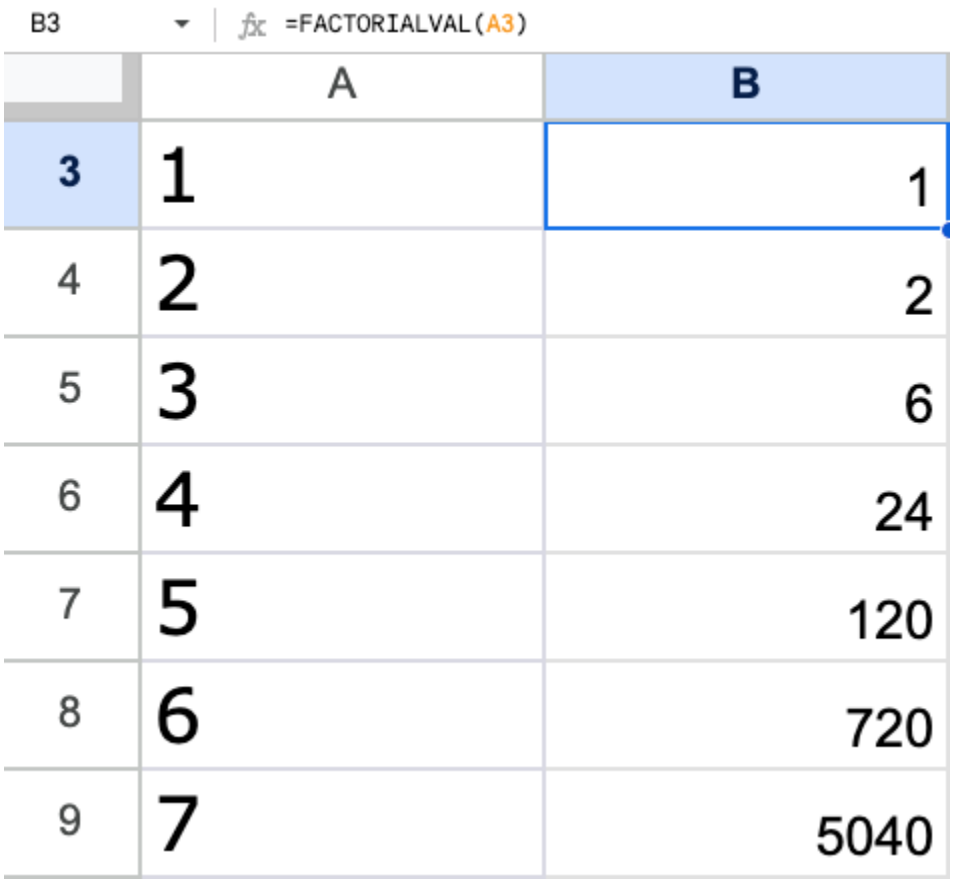
1. CONCAT_CAP(str1, str2):
    - This function takes two string parameters, str1 and str2.
    - It calls the upperMe function with str1 as an argument and stores the result in the stra variable.
    - It calls the upperMe function with str2 as an argument and stores the result in the strb variable.
    - It returns the concatenation of stra and strb using the concat method.
2. Overall, this function converts both input strings to uppercase and then concatenates them.
3. CONCAT_CAP2(str1, str2):
    - This function takes two string parameters, str1 and str2.
    - It retrieves the first character of str1 using the charAt method, converts it to uppercase, and assigns it to the firstLetter variable.
    - It concatenates str1 and str2 using the concat method and assigns the result to the concatVal variable.
    - It retrieves the first character of concatVal, converts it to uppercase, and assigns it back to the firstLetter variable.
    - It converts the remaining characters of concatVal to lowercase and assigns them to the restVals variable using the slice and toLowerCase methods.
    - It returns the concatenation of firstLetter and restVals.

4. This function capitalizes the first letter of the resulting concatenation of str1 and str2 and converts the remaining characters to lowercase.
5. upperMe(val):
    - This function takes a string parameter, val.
    - It logs the val to the Logger for debugging purposes.
    - It retrieves the first character of val using the charAt method, converts it to uppercase, and assigns it to the firstLetter variable.
    - It converts the remaining characters of val to lowercase and assigns them to the restVals variable using the slice and toLowerCase methods.
    - It returns the concatenation of firstLetter and restVals.
6. This function capitalizes the first letter of val and converts the remaining characters to lowercase.
7. test():
    - This function is not used within the provided code and has no impact on the execution of the other functions. It logs the value of the val variable (which is set to 'laurence') when called, but it does not return any value.

In summary, the code provides three functions (CONCAT_CAP, CONCAT_CAP2, and upperMe) that manipulate strings in various ways such as converting characters to uppercase or lowercase, concatenating strings, and capitalizing the first letter. The test() function is not directly related to the other functions and serves as a standalone test function.

Calculate the factorial of a given number

https://youtu.be/w_vIDg7OMoc




```
function FACTORIALVAL(val){
 if(val == 0){
   return 1;
 }else{
   Logger.log(val);
   return val * FACTORIALVAL(val-1);
 }
```

Laurence Svekis https://basescripts.com/

```
}


function test1(){
 let v =4;
 Logger.log(FACTORIALVAL(v));
}
```

The given code consists of two functions: FACTORIALVAL and test1. Let's break down each function and its purpose:

1. FACTORIALVAL(val):
    - This function calculates the factorial value of a given number val.
    - It first checks if the val is equal to 0. If it is, the function immediately returns 1. This is the base case of the factorial calculation, as the factorial of 0 is defined as 1.
    - If val is not equal to 0, the function logs the value of val to the Logger for debugging purposes.
    - It then recursively calls the FACTORIALVAL function with val-1 as the argument and multiplies the result with val.
    - The recursion continues until the base case is reached (when val becomes 0), at which point the recursion stops and the function returns the final factorial value.
2. In summary, the FACTORIALVAL function recursively calculates the factorial of a given number using the formula n! = n * (n-1)!.
3. test1():

- This function is not directly related to the factorial calculation. It serves as a test function to showcase the usage of the FACTORIALVAL function.
- It initializes a variable v with the value 4.
- It logs the result of calling the FACTORIALVAL function with v as the argument to the Logger.

4. When executed, the test1 function will log the intermediate values of val during the recursive calculation of the factorial of 4 to the Logger, as well as the final factorial value.

In summary, the code provides a FACTORIALVAL function that calculates the factorial value of a given number using recursion, and a test1 function that demonstrates the usage of the FACTORIALVAL function by calculating the factorial of 4 and logging the result.

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

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

https://youtu.be/36zBPR6349A

C2 =COUNT_VAL(A2,B2)

	A	B	C
1	range	value	
2	1,2,3,4,5,81,2,3,4,5,8	3	2
3	5,4,3,2,1	5	1
4	2,2,2,2,2	2	5
5	1,1,1,1,1	2	0
6	foo,bar,baz,foo,foo	foo	3

The provided code defines an Apps Script function called COUNT_VAL that counts the occurrences of a specific value within a given range. Let's break down the code step by step:

1. Initialize a variable count to keep track of the count of occurrences of the specified value within the range. Set it to 0 initially.
2. Split the range string into an array of values using the split method. The delimiter used for splitting is a comma (','), as specified by range.split(',').
3. Iterate over each element (v) in the arr array using the forEach method and a callback function.
4. Inside the callback function, the code checks if the current element (v) is equal to the specified value (val) after

removing any leading or trailing whitespace from v using the trim method.

5. If the equality condition (val == v.trim()) is true, it means the current element matches the specified value. In this case, increment the count variable by 1.
6. After iterating over all the elements in the array, the function returns the final value of the count variable, representing the total count of occurrences of the specified value within the given range.

In summary, the COUNT_VAL function takes a range (a comma-separated string of values) and a target value (val). It iterates through the values in the range, counts the number of occurrences of the target value, and returns the count.

```
function COUNT_VAL(range,val){
 let count = 0;
 const arr = range.split(',');
 arr.forEach(v =>{
   if(val == v.trim()){
     count++;
   }
 })
 return count;
}
```

range	value	

1,2,3,4,5,81,2,3,4,5,8	3	2
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

Calculate the distance between two sets of latitude and longitude coordinates

https://youtu.be/cD6pUr_5RtE

E3 =DISTANCE_BETWEEN(A3,B3,C3,D3)

	A	B	C	D	E
1	latitude 1	longitude 1	latitude 2	longitude 2	distance
2	37.7749	-122.4194	40.7128	-74.006	4129.086165
3	51.5074	-0.1278	48.8566	2.3522	343.5560603
4	-33.8688	151.2093	22.3193	114.1694	7375.505614
5	35.6895	139.6917	37.5665	126.978	1152.618258

The code defines a function called DISTANCE_BETWEEN that calculates the distance between two geographical coordinates on the Earth's surface using the Haversine formula. Here's a step-by-step breakdown of the code:

1. Define a constant earthRadius with a value of 6371, representing the Earth's radius in kilometers.
2. Calculate the difference in latitude (dLat) and longitude (dLon) between the two coordinates. These differences are obtained by subtracting the respective initial coordinates (lat1 and lon1) from the final coordinates (lat2 and lon2).
3. Convert the differences in latitude and longitude from degrees to radians by calling the degreesToRadians function with the respective differences as arguments. The degreesToRadians function converts degrees to radians using the formula deg * Math.PI / 180.
4. Use the Haversine formula to calculate the distance between the two coordinates. The Haversine formula involves several trigonometric calculations. Here's the breakdown of the

formula:

a. Calculate a using the following equation:

5. arduino
6. Copy code
7. 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);

   This equation represents the squared value of the Haversine function applied to the angular differences. It involves computing the squares of sine and cosine functions.

   b. Calculate c using the following equation:
8. arduino
9. Copy code
10. c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

    The atan2 function calculates the arctangent of the ratio of two numbers. In this case, it computes the arctangent of the square root of a divided by the square root of 1 - a. The result represents the angular distance in radians.

    c. Calculate distance by multiplying earthRadius with c. This step converts the angular distance to a physical distance in kilometers.
11. Finally, return the calculated distance value as the result of the function.

Additionally, the code includes a helper function called degreesToRadians, which converts degrees to radians by multiplying the input value (deg) by Math.PI / 180. This function is used to convert the latitude and longitude differences from degrees to radians.

In summary, the DISTANCE_BETWEEN function calculates the distance (in kilometers) between two sets of latitude and

longitude coordinates using the Haversine formula. It leverages the degreesToRadians helper function to convert degrees to radians and performs a series of trigonometric calculations to determine the distance.

```
function DISTANCE_BETWEEN(lat1,lon1,lat2,lon2){
 const earthRadius = 6371;
 const dLat = degreesToRadians(lat2-lat1);
 const dLon = degreesToRadians(lon2-lon1);
 const 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);
 const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
 const distance = earthRadius * c;
 return distance;
}

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

latitude 1	longitude 1	latitude 2	longitude 2	distance
37.7749	-122.4194	40.7128	-74.006	4129.086165

51.5074	-0.1278	48.8566	2.3522	343.5560603
-33.8688	151.2093	22.3193	114.1694	7375.505614
35.6895	139.6917	37.5665	126.978	1152.618258

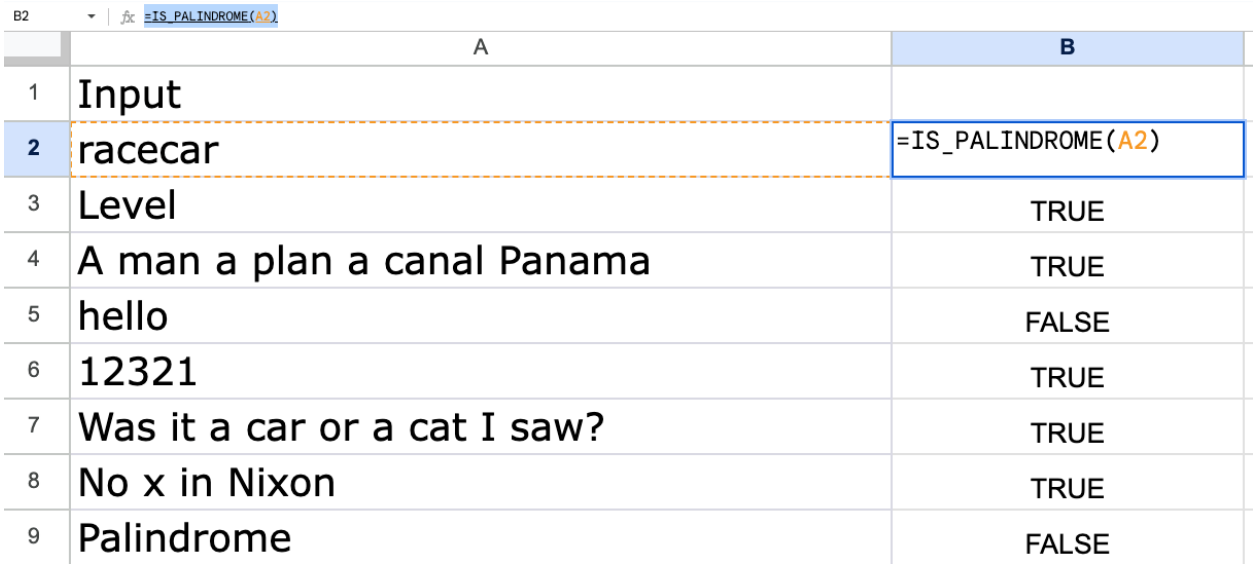
Check if a given string is a palindrome

https://youtu.be/8oalWIwdNMc

A palindrome is a word, phrase, number, or sequence of characters that reads the same forward and backward, ignoring spaces, punctuation, and capitalization (in the case of letters). In other words, a palindrome remains unchanged when its order is reversed.
For example, here are some examples of palindromes:

- "level"
- "madam"
- "racecar"
- "A man, a plan, a canal: Panama"
- "12321"
- "deed"

In each of these examples, the sequence of characters reads the same from left to right as it does from right to left. Palindromes can be formed by individual letters, entire words, or even complete sentences.



	A	B
1	Input	
2	racecar	=IS_PALINDROME(A2)
3	Level	TRUE
4	A man a plan a canal Panama	TRUE
5	hello	FALSE
6	12321	TRUE
7	Was it a car or a cat I saw?	TRUE
8	No x in Nixon	TRUE
9	Palindrome	FALSE

```
function IS_PALINDROME(str){
 str = str.toString();
 str = str.toLowerCase().replace(/[^a-z0-9]/g,'');
 for(let i=0;i<str.length/2;i++){
   if(str[i] !== str[str.length-1-i]){
     return false;
   }
 }
 return true;
}
```

The above code defines a function called IS_PALINDROME that checks whether a given input string is a palindrome. Here's a step-by-step breakdown of the code:

1. Convert the input str to a string explicitly using the toString() method. This step ensures that the input is treated as a string, even if it was originally a different data type.
2. Convert the string to lowercase using the toLowerCase() method. This step ensures that the comparison is case-insensitive.
3. Remove any non-alphanumeric characters from the string using the replace() method with a regular expression /[^a-z0-9]/g and replacing them with an empty string. This step eliminates any characters that are not letters or digits from the string.
4. Iterate over the characters in the string using a for loop. The loop variable i starts at 0 and increments until it reaches half of the string length (str.length/2).
5. Inside the loop, compare the character at index i with the character at the corresponding index from the end of the string (str.length-1-i).
6. If the characters being compared are not equal, it means the string is not a palindrome. In this case, the function immediately returns false to indicate that the input string is not a palindrome.
7. If the loop completes without finding any unequal characters, it means the string is a palindrome. The function returns true to indicate that the input string is a palindrome.

In summary, the IS_PALINDROME function checks whether a given input string is a palindrome by converting the string to lowercase, removing non-alphanumeric characters, and then comparing characters from both ends of the string towards the middle. If all the characters match, the function returns true, indicating that the string is a palindrome. If any characters don't

match, the function returns false, indicating that the string is not a palindrome.

Input	
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

https://youtu.be/7FO7QONKKuQ

B2 =CAL_AGE(A2)

	A	B
1	birthdate	Age
2	1990-01-01	33
3	1985-03-15	38
4	1975-12-31	47
5	2015-07-10	7
6	1999-11-25	23

```
function CAL_AGE(birthdate){
 const mils = Date.now() - birthdate.getTime();
 const ageDate = new Date(mils);
 const age = Math.abs(ageDate.getUTCFullYear()-1970);
 return age;
}
```

The above code defines a function called CAL_AGE that calculates the age in years based on a given birthdate. Here's a step-by-step breakdown of the code:

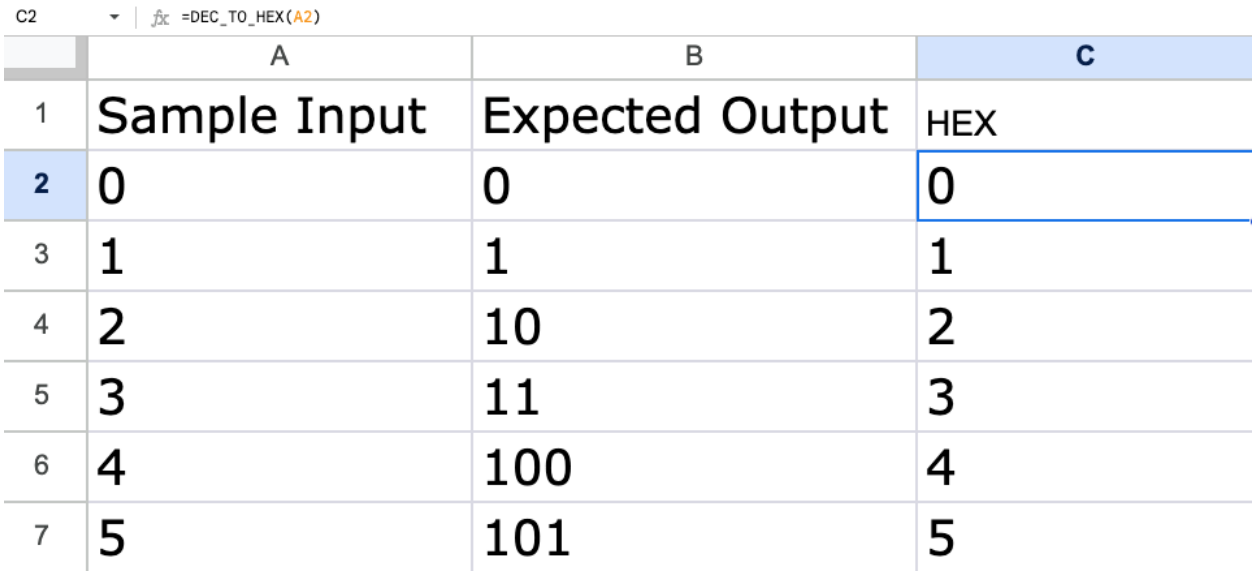
1. Calculate the difference in milliseconds between the current date and time (obtained using Date.now()) and the birthdate. This is done by subtracting the birthdate's time in milliseconds (obtained using birthdate.getTime()) from the current time. The result is stored in the variable mils.
2. Create a new Date object called ageDate using the calculated mils value. This object represents a date that is the calculated difference in milliseconds from the reference date (January 1, 1970, 00:00:00 UTC).
3. Calculate the age in years by subtracting 1970 (the reference year) from the full year (getUTCFullYear()) of the ageDate object. The getUTCFullYear() method retrieves the year in UTC (Coordinated Universal Time) format.
4. Use the Math.abs() function to ensure that the age is always a positive value. This is necessary because the subtraction of 1970 may result in a negative value if the birthdate is in the future.
5. Return the calculated age as the result of the function.

In summary, the CAL_AGE function takes a birthdate as input and calculates the age in years based on the current date and time. It uses the difference in milliseconds between the birthdate and the current date to determine the age and returns it as an integer.

birthdate	Age
1990-01-01	33
1985-03-15	38
1975-12-31	47
2015-07-10	7
1999-11-25	23

Convert a number from decimal to binary

https://youtu.be/3yKhEN_hxhg



	A	B	C
1	Sample Input	Expected Output	HEX
2	0	0	0
3	1	1	1
4	2	10	2
5	3	11	3
6	4	100	4
7	5	101	5

```
function DEC_TO_BINARY(dec){
 return dec.toString(2);
}
function DEC_TO_HEX(dec){
 return dec.toString(16);
}
```

The above code consists of two functions: DEC_TO_BINARY and DEC_TO_HEX. Let's break down each function and its purpose:

1. DEC_TO_BINARY(dec):
    - This function takes a decimal number dec as input and converts it to its binary representation.

- The toString(2) method is called on the dec number to convert it to a string representation in base 2 (binary).
- The function returns the binary representation of the decimal number as a string.

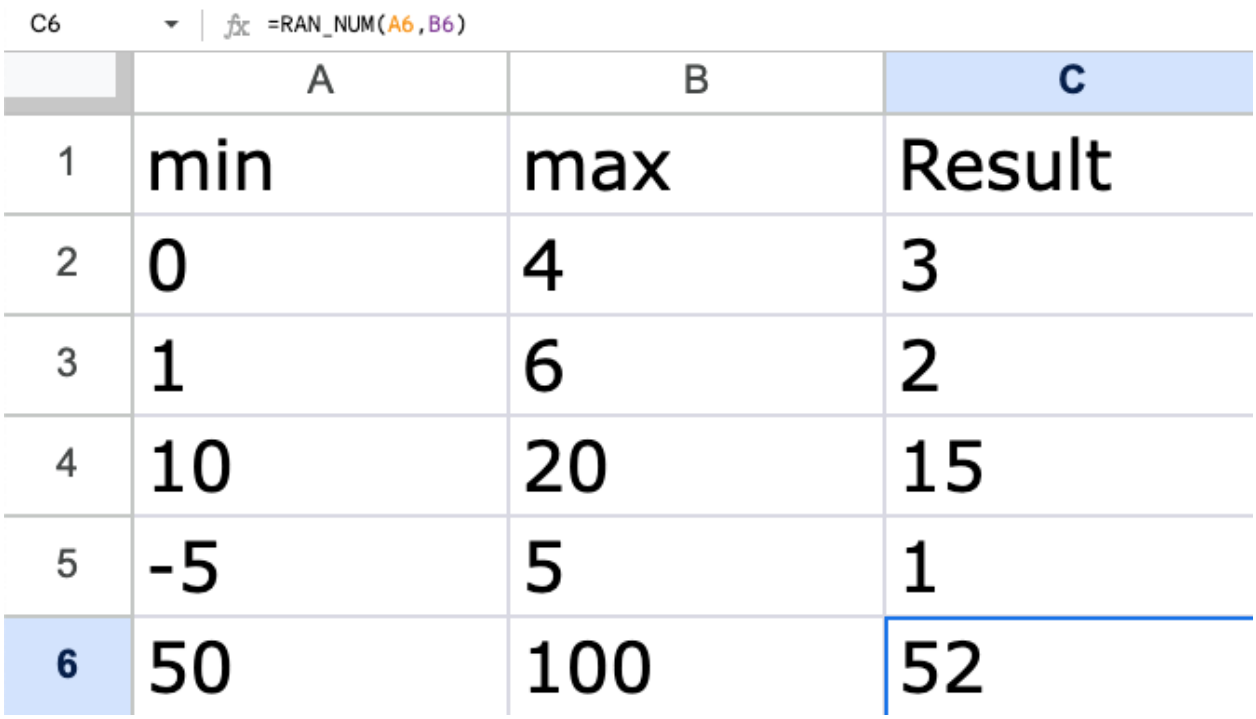
2. In summary, the DEC_TO_BINARY function converts a decimal number to its binary representation.
3. DEC_TO_HEX(dec):
    - This function takes a decimal number dec as input and converts it to its hexadecimal representation.
    - The toString(16) method is called on the dec number to convert it to a string representation in base 16 (hexadecimal).
    - The function returns the hexadecimal representation of the decimal number as a string.
4. In summary, the DEC_TO_HEX function converts a decimal number to its hexadecimal representation.

These functions provide a convenient way to convert decimal numbers to binary and hexadecimal representations using the built-in toString method in JavaScript.

Sample Input	Expected Output	HEX
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5

Random number between two given numbers

https://youtu.be/tWhZGNPCOlQ



	A	B	C
1	min	max	Result
2	0	4	3
3	1	6	2
4	10	20	15
5	-5	5	1
6	50	100	52

```
function RAN_NUM(min,max){
 return Math.floor(Math.random()*(max-min+1))+min;
}
```

The above code defines a function called RAN_NUM that generates a random number within a given range. Here's a step-by-step breakdown of the code:

1. The function takes two parameters, min and max, which represent the minimum and maximum values of the desired range.

2. The Math.random() function generates a random decimal number between 0 (inclusive) and 1 (exclusive). This function is a built-in JavaScript method that returns a random floating-point number.
3. The expression Math.random() * (max - min + 1) calculates a random number within the range (max - min + 1). This expression scales the random number to fit the desired range.
4. The Math.floor() function is called on the result of the previous expression to round down the random number to the nearest integer. This ensures that the generated number is an integer within the range (max - min + 1).
5. The rounded-down random number is then added to the min value using the +min operation. This step shifts the range from (max - min + 1) to the desired range from min to max.
6. The final result is returned as the output of the function.

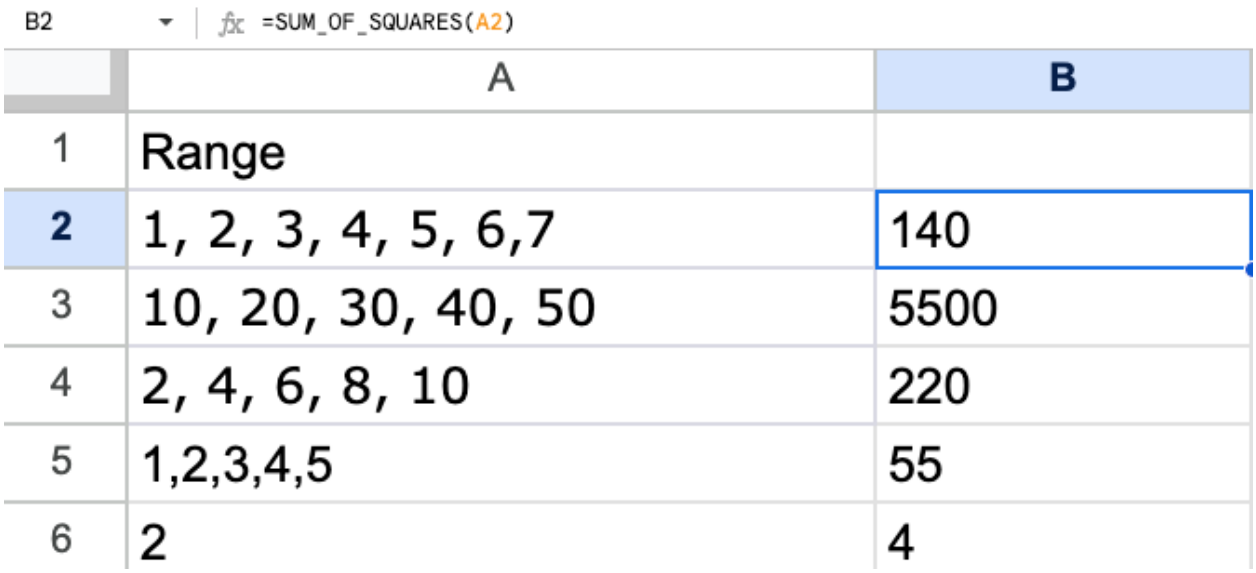
In summary, the RAN_NUM function generates a random integer within a specified range defined by the min and max parameters. It uses the Math.random() function to generate a random decimal number, scales it to the desired range, rounds it down to the nearest integer, and then adjusts the range by adding the min value.

min	max	Result
0	4	3
1	6	2
10	20	15
-5	5	1

50	100	52

Sum of the squares of the numbers in a range

https://youtu.be/sTiZFFD-bmc



	A	B
1	Range	
2	1, 2, 3, 4, 5, 6,7	140
3	10, 20, 30, 40, 50	5500
4	2, 4, 6, 8, 10	220
5	1,2,3,4,5	55
6	2	4

The provided code defines a function called SUM_OF_SQUARES that calculates the sum of the squares of numbers within a given range. Here's a step-by-step breakdown of the code:

1. Initialize a variable sum with a value of 0. This variable will store the cumulative sum of the squares.
2. Convert the range input to a string using the toString() method. This step ensures that the range is treated as a string, even if it was originally a different data type.
3. Remove any whitespace characters from the string using the replace() method with a regular expression /\s+/g and replacing them with an empty string. This step eliminates any spaces within the string.
4. Trim any leading or trailing whitespace from the string using the trim() method. This step ensures that any extraneous whitespace at the beginning or end of the string is removed.

5. Split the modified range string into an array of individual values using the split() method. The values are split using commas as the delimiter.
6. Iterate over each value in the arr array using the forEach() method.
7. Within the loop, convert each value to a number by implicitly multiplying it with itself (val * val). This step ensures that the value is treated as a number for the subsequent addition.
8. Add the squared value to the sum variable.
9. After iterating through all the values, return the calculated sum as the result of the function.

In summary, the SUM_OF_SQUARES function takes a range of numbers as input, calculates the square of each number within the range, and returns the sum of the squared values. The code ensures that the input range is converted to a string, any whitespace is removed, and each value is squared and added to the cumulative sum.

```
function SUM_OF_SQUARES(range){
 let sum = 0;
 range = range.toString();
 range = range.replace(/\s+/g,'').trim();
 const arr = range.split(',');
   arr.forEach(val =>{
   sum += val * val;
 })
```

```
 return sum;
 //return JSON.stringify(arr);
}
```

Range	
1, 2, 3, 4, 5, 6,7	140
10, 20, 30, 40, 50	5500
2, 4, 6, 8, 10	220
1,2,3,4,5	55
2	4