Answer:

5.  The difference between the two functions is that max2 will take only one iteration to compute, whereas max1 will take two iterations.

I think that both functions will perform the same amount of computation, but max2 will be faster.

I believe that the two functions will take different amounts of time to run—max1 takes two iterations and max2 takes one iteration. I'm not sure why this is happening, but if I were to guess, it might have something to do with how much memory each function needs.

 The two functions take the same amount of time to run, and they perform the same number of operations. The difference between them is that max2 uses a loop to check each value in turn, while max1 just checks the first value.

I think that max2 will be faster than max1 because it does not use a loop at all

The two functions are performing different operations. The first function, max1, is returning a value based on the length of a string, while the second function, max2, is returning a value based on the number of spaces in a string. The two functions will likely perform different amounts of computation and take different amounts of time to run. I think max1 will be faster because it's only comparing one value at a time; this means it can keep running as long as there are more values to compare. On the other hand, max2 needs to compare every character in the string with each other character before it can get an answer—which means it's much less efficient than max1..

Step-by-Step explanation

6.  The first step to performing an experiment is to answer the questions above. After you've answered all of the questions, it's time to run your code.

If you get an error, document what you think should happen and compare it to what actually happens.  Before running the code, we must first answer the following questions:

1. What is the difference between a normal and an extreme value of a function?

-What happens if you increase an input by a very large amount?

2. How would you determine whether or not a function is increasing or decreasing?

3. If a function takes in an input and returns another function, what will happen if you call that function with a high number?

 After running the code, we can see that the red light stays on. This is because we are using a button to turn it on and off, and each time it's pressed, it turns on.

The green light should have turned on when we pressed the button. However, it did not turn on at all. This is because when we press the button, it sets the green LED to output 1V. The LED does not receive any voltage from the Arduino so it does not turn on at all.

7.  The code was run on a large input, so you can tell that it took max1 and max2 about the same amount of time to run. The output also tells you that max1 printed the maximum value 1337.

The code is:

#include <stdio.h>

#include <stdlib.h>

#define MAX\_INPUT 1000000

int main() {

    int max1, max2;

    double num1, num2;

    double sum1, sum2;

   }

#include <stdio.h>

#include <stdlib.h>

#define MAX\_INPUT 1000000

int max1, max2;

double num1, num2;

double sum1, sum2;

// run a couple of times to convince yourself that the runtimes are consistent: for(int i = 0; i< 1000000; ++i) { num1 = sqrt(1e6 \* i);//integer numbers only! num2 = sqrt(1e6 \* (i+100000));//double numbers only! sum1 = 1.0/(double)(num1\*\*2 + 1e-6\*num2\*\*2);//integer and double only! sum2 = 1.0/(double)(sqrt(num1\*\*2 + 1e-6\*num

8.

 As expected, the functions run at their expected speeds. The functions are compiled into machine code and executed on the GPU, which is significantly faster than a CPU. This is because the GPU has much more RAM than a CPU and can thus store more data in each instruction.

The reason that the functions run so slowly is because they don't have access to as much RAM as they would if they were compiled into machine code. There was no way to predict how much data would be required by each function in advance, so it was assumed that each function would need only enough RAM for its own variables and parameters (the size of those variables and parameters was not known). However, because of how quickly these variables' values change during an execution, it's possible for a line of code to access more space than it actually needs.

The only way to solve this problem would be to rewrite all of our functions in C++ instead of Python because it allows us to use pointers instead of having to explicitly declare every variable and point them at where they should go (which is what happens when we're using Python).

However, this wouldn't be ideal because then we'd no longer be able to use NumPy's high-performance matrix operations or Sci.

 The functions run at their expected speeds. This is because of my knowledge on how to optimize a function and how to write code that runs quickly.

I first wrote the code to speed up the function I was using. I did this by changing the loop condition, adding more variables, and using an index variable to store the value of each letter in the string. Then I optimized it by writing it with a for loop instead of a while loop. This made it so that we could get through more letters in less time than before which allowed us to run faster.