Covering here different answers from everywhere inside of the web:

> I want to build a real number s in $[0, 1]$ that I know isn't on this list. This means I have a bunch of **requirements** to meet: I need $s \in [0, 1]$, and I need
>
> $$s \neq r_1, \quad s \neq r_2, \quad s \neq r_3, \quad \ldots.$$
>
> The idea is that I'll define my real number so that each binary digit takes care of some requirement.
>
> First, to make $s = [0, 1]$, let's begin with "$0.----$". There, that was easy. Now what about the *other* requirements?
>
> I'm going to rewrite our array above, but with certain digits suggestively highlighted:
>
> - 0.01101000...
> - 0.10101100...
> - 0.11111111...
> - 0.01100001...
>
> And remember that two real numbers are different if their binary expansions are *ever* different. *(As remarked above, this isn't quite true on the nose, but ignore it for now; it's easy to fix after you get the general idea.)* So e.g. to make sure $s \neq r_1$ I just need s and r_1 to have one different digit, and so forth.

So here's how we do that:

- Take all the red digits and put them together - this is the **diagonal sequence**. In this case it's {\color{red} 0010...}. The nth number of the diagonal sequence is the nth digit of the nth real on our list (*check this!*)
- Reverse them! Change each 0 to a 1, and each 1 to a 0. This is the **antidiagonal sequence**, and in our case is 1101...
- Put a "0." in front of the antidiagonal sequence to get a real in $[0, 1]$ (here, "0.1101..."); this is our s.

Now clearly $s \in [0, 1]$ so it's enough to check that s isn't on our list.

- $s \neq r_1$, since s and r_1 differ on the first binary digit: s has a 1 but r_1 has a 0.
- $s \neq r_2$, since s and r_2 differ on the second binary digit: s has a 1 but r_1 has a 0.
- In general, $s \neq r_n$ for any n, since s and r_n will always have different nth binary digits.

So s isn't on the list! And what we've shown, in fact, is that **no list contains every real number.**

Cantor's diagonalization method is a powerful proof technique in mathematics and computer science, particularly in the areas of set theory and computability theory. It was introduced by the mathematician Georg Cantor to prove that the set of real numbers is uncountable, meaning it has a higher cardinality than the set of natural numbers. This method has since found applications in various fields, including the study of computable functions and the existence of non-computable functions.

The practical meaning of Cantor's diagonalization method lies in its ability to demonstrate the existence of objects or entities that possess certain properties or behaviors that are different from a given countable collection of objects or entities. It allows us to prove the existence of such objects without explicitly constructing them.

In the context of computability theory, Cantor's diagonalization is used to prove the existence of non-computable functions. It shows that there are more functions from natural numbers to natural numbers (denoted as N → N) than there are computable functions. This means that not every function can be computed by an algorithm or a Turing machine.

The difference is that we *can* write down an (infinite) list of all natural numbers. Here, I'll start it right now:

1
2
3
4
5
...

With that list, I've listed *every single* natural number. I'm not missing any of them. The point of Cantor's diagonalization argument is that *any* list of real numbers you write down will be incomplete, because for any list, I can find some real number that is not on your list.

The diagonalization argument proceeds as follows:

1. Assume, for the sake of contradiction, that the set of functions from N to N is countable.
2. Enumerate all the functions in this set as f_0, f_1, f_2, ...
3. Construct a table where the rows represent the functions and the columns represent the arguments (natural numbers).
4. Define a new function f that differs from each function f_i at the i-th position. This is done by adding 1 to the value of $f_i(i)$ or by using any other suitable modification.
5. The newly constructed function f is different from every function in the enumeration, leading to a contradiction.
6. Therefore, the assumption that the set of functions from N to N is countable must be false.

Applications:

1. Existence of Non-Computable Functions:
    - Cantor's diagonalization is used to prove the existence of non-computable functions.
    - It shows that there are functions that cannot be computed by any algorithm or Turing machine.
    - This has implications in the study of computability theory and the limits of computation.
2. Cardinality of Sets:
    - Cantor's diagonalization is used to compare the cardinality of different sets.
    - It can be used to prove that the power set of a set A (denoted as 2^A) has a higher cardinality than the set A itself.
    - This has applications in set theory and the study of infinite sets.
3. Undecidability and Incompleteness:
    - Cantor's diagonalization is used to prove the undecidability of certain problems in computability theory.
    - It is also used in Gödel's incompleteness theorems to show the limitations of formal systems.