Hello,

The game of Nim is a 2-player game. There are 3 piles of coins (initially, there are a random number of coins in each pile). Players make their moves alternatingly. In each turn, a player HAS to make a move, i.e, pick up some number of coins (he cannot pick up zero coins), from any ONE pile only. The player who picks up the last coin (not of a given pile, but the last coin overall) wins the game.

An interesting observation can be made if we consider that we have reached some state where the coins are of the form [0, n, n] for some value of n. For example, [0, 10, 10]. This means that there are 0 coins left in the first pile, and 10 coins left in both the second and third piles. Now, suppose the opponent picks up 6 coins from the third pile. Then, you are supposed to pick up the same number of coins from the other pile (in this case, the second pile). Hence this will lead to [0, 4, 4]. Proceeding with this lockstep synchronization moves, i.e., picking up the same number of coins that your opponent has picked up from the other pile (the one not chosen by your opponent in his current move), we might end up with [0, 1, 1]. Now it is the opponent's turn. Remember, the rules of the game forbid from refraining to make a move. So he HAS to pick up something. Also, he has to pick up from one pile only. Suppose he picks from the third pile, making the configuration [0, 1, 0]. It is now your turn, and hence, you pick up the one remaining coin and win the game.

So, you can observe that if you have reached a configuration of [0, n, n] (or any permutation of this), and it is your opponent's turn to make the next move, then you simply have to mimic whatever move your opponent makes and you will eventually win the game. The question is: How can we go from any arbitrary configuration to some permutation of [0, n, n]? It turns out there is a clever strategy, which can be best demonstrated with an example.

Let us suppose the initial configuration is [15, 10, 8]. It is your turn to make a move. Your sole goal is to ensure that you do something so that you will get to [0, n, n] and this will be the configuration for the opponent (because then, as we saw, in this state, the opponent is going to go downhill). Follow this method, and verify for a few cases that it works:

- Represent each number in the pile of coins in its binary representation. So, in this case, it will be:

15 => 1 1 1 1

10 => 1 0 1 0

8 => 1 0 0 0

- Next, find the sum (the decimal sum, and NOT the binary sum) of this binary-represented triple.

15 => 1 1 1 1

10 => 1 0 1 0

8 => 1 0 0 0

-----------------

3 1 2 1

------------------

- You will need to ensure that this sum contains all even digits. Thus, in your move, you need to change the bits in such a way that each digit becomes even (we start scanning from left to right). Here, the first digit encountered is 3, which is odd, and hence we need to convert this to an even number. This can be done if we change any one of the 1's (that were adding up so as to give a sum of 3) into a 0 (so that the 1's will add up to now give 2). At this point, we have a choice of which 1 to change. Let us say we pick up the third 1 and change that to a zero. Continuing, we see that the next digit in the sum is 1, which is also odd, and hence must be made even. At this point, you might think that we can change the 1 in the first row to 0, thereby making the sum 0. However, you CANNOT DO THIS!! The reason is that you have already changed the 1 of the third row to 0, and hence, you must now continue to toggle the bits of the third row ONLY (think for a moment why this is necessary – imagine if you changed the 1 of the first row to 0, then the first row will end up becoming a new binary number, AND the third row will also end up becoming a new binary number, since we had changed from the third row for the first digit – which essentially means that we have made a move wherein we have picked up some number of coins from the first pile AS WELL AS the third pile; which is against the rules of the game!). Hence we toggle the 0 of the third row to 1. Similarly, continuing, we find that we can toggle the 0 of the last column of the last row to 1 (Verify this). Thus, the new configuration becomes:

1 1 1 1 => 15

1 0 1 0 => 10

0 1 0 1 => 5

- - - - - -

2 2 2 2

- - - - - -

Hence this must be the move that we must make. That is, change the configuration from [15, 10, 8] to [15, 10, 5]. Now, this configuration is not any permutation of [0, n, n], and so we are still not done. It is now the opponent's turn. Let's say the opponent changes this to: [6, 10, 5]. In our turn, we again follow the same algorithm:

6 => 0 1 1 0

10 => 1 0 1 0

5 => 0 1 0 1

-----------

1 2 2 1

--------------

Now, scanning from left to right, we find that the first digit in the sum is 1, which is odd. Hence we need to change this to an even number. You might be tempted to suggest that we can change the 0 of the third row to a 1 for this bit (thereby making the sum 2, which is an even number). However, pause a moment to see the ramifications of this. If we changed that 0 to 1, we are changing the third row to a new number, but, the new number that will result will be greater than 5 (since this is like prepending a 1 to a binary number, which will increase the number). Clearly, the rules of the game do not permit us to INCREASE the number of coins, and hence changing this zero to one is not permitted. The same argument applies to changing the 0 of the first row to a 1. Hence, we have only one choice – to change the 1 of the second row to a 0 (this is unlike the previous case, where we had the freedom to change any of the bits in the first digit – now, we are restricted to changing only the 1 of the second pile!). Thus, the final configuration becomes:

0 1 1 0 => 6

0 0 1 1 => 3

0 1 0 1 => 5

- - - - -

0 2 2 2

- - - - -

Thus, we need to change from the previous configuration to: [6, 3, 5]. Now it is the opponent's turn. Suppose the opponent changes the configuration to: [5, 3, 5]. Please verify that in the next move that you make (following the above algorithm), you will end up with the configuration: [5, 0, 5]. Clearly, you have now landed up in a permutation of [0, n, n], which is a winning streak for you henceforth. It can be shown that if you follow the above algorithm, you can reach a permutation of [0, n, n] starting from any initial configuration.

Now, your assignment is this: Write a Python Program that plays this game against a human. The initial configuration must be chosen as a random sequence of 3 numbers. Assume that the human does not know the strategy mentioned above. After the human player makes his/her move, the computer will follow the strategy outlined above to make its move (in effect, this is like an “Artificial Intelligence” program that seems to be very clever and intelligent in that it knows how to win).

One subtlety to consider: What if when performing the above step of converting each pile to a binary number and adding up the digits, we end up with a sum where all the digits are even? Our strategy was to make a move so as to change all odd digits to even, but what if there are no odd digits? We are not allowed to make no move. Hence, at this point, the best thing that can be done is to make a random move (the computer cannot do any better than this, since getting this configuration would likely imply that the opponent, i.e., human player already knows the strategy for playing the game, and is probably following it against the computer! The best that the computer can do under this circumstance is to make a random move, hoping that the opponent will slip up in a subsequent move).