[An algorithm for finding the highest number]

As is often the case, there are a number of ways that this could be done, and it will depend a lot on the operations the computer will understand.

For example, we could just say “Take the maximum of 3, 6, 10, …”, however if our computer doesn’t understand this “maximum” function, then there will be other ways. For example, if a computer can understand the binary “greater than” operation, then we can move through each pair. So taking pairs, the greater of 3 and 6 is 6, 10 is greater than 6, 20 is greater than 10 and we could keep on going this way: but that’s not actually what the “greater than” symbol tells us. It actually gives us a logical output of TRUE or FALSE. So what we actually need to say here is, for each pair of numbers moving across, if the first number is greater than the second, record the first, if not, record the second. If we did this for each pair then we’d need to record all of the winners, and then move through the pairs again - and so we’d require a kind of “memory” command. If we do it this way, we’ll need to be able to store all of these values in a list, so another way we can do it with comparisons and one memory space is just to proceed like this. If the first number is bigger than the second, record the first, otherwise, record the second. Now compare the number we recorded to the next number, if it is larger, then keep it recorded, otherwise, overwrite that record with the other number and continue until we reach the end of the list.

Intuitively, this might seem very different to how we would tell with such a small list of numbers, but actually it might be a good technique if we were searching a really long list where we can’t see all the numbers at once, since the computer can very quickly perform this simple check 1000s or millions of times.

The difficulty with programming is matching these different

techniques that we might consciously or unconsciously use with the language of the computer.