Advantages of Functions

How to reduce redundancy using Functions?

To illustrate this, let us take a very basic human habit of drinking water.

If we look at the action of drinking water, it involves 4 main steps only:

Take a glass of water.
Sip the water from the glass.
Gulp the water down the throat.
Put down the glass when we are no longer thirsty.

Although the action “Drink water” constitutes 4 steps, while describing our day-to-day life routine to someone else, we use phrases like “Drink water”. This action of “Drink water” is easy for the other person to understand and covers up the above 4 steps in a single action. Thus, every time we include “Drink water” in our routine, it will automatically cover the above 4 steps.

Similarly, the main idea behind using a function in your code is to keep the code **DRY** (**D**on’t **R**epeat **Y**ourself). Cutting out repeated commands helps to minimize errors, keeps code short, and saves programming time.

Advantages of using Functions

A few of the advantages of using functions are:

Increases readability makes code organized and easy to understand.

Reduces code length: redundant code is removed and replaced by functions.
Reusability: Code reusability increases.

Can Function return a value?

Till now we have only used functions in a way wherein once a function is called, execution of all the logic and display of the output was done inside the function.

However, as discussed before, the main purpose behind using functions is to get rid of repetitive chunks of code. **Thus, the usefulness of using a function in a program comes to the forefront, when an operation performed inside a function gives back a value, which can be used later in the program to generate meaningful results.**

As an example, consider a scenario wherein we calculate the square of a number using a function, calculate the cube of another number using another function and then add the results generated from these two functions and print them.

```
square_of_number (input1)
{
  result1 = input1 * input1
  return result1
}

cube_of_number (input2)
{
  result2 = input2* input2*input2
  return result2
}

main function ()
{
  val1 = square_of_number (2)
```

```
  val2 = square_of_number (3)
  val3 = val1 + val2
}
```