# What is an Algorithm?

In this tutorial, we will learn what algorithms are with the help of examples.

In computer programming terms, an algorithm is a set of well-defined instructions to solve a particular problem. It takes a set of input(s) and produces the desired output. For example,

An algorithm to add two numbers:

1. Take two number inputs
2. Add numbers using the + operator
3. Display the result

---

## Qualities of a Good Algorithm

- Input and output should be defined precisely.
- Each step in the algorithm should be clear and unambiguous.
- Algorithms should be most effective among many different ways to solve a problem.
- An algorithm shouldn't include computer code. Instead, the algorithm should be written in such a way that it can be used in different programming languages.

---

## Algorithm Examples

Algorithm to add two numbers

Algorithm to find the largest among three numbers

Algorithm to find all the roots of the quadratic equation

Algorithm to find the factorial

Algorithm to check prime number

Algorithm of Fibonacci series

## Algorithm 1: Add two numbers entered by the user

```
Step 1: Start
Step 2: Declare variables num1, num2 and sum.
Step 3: Read values num1 and num2.
Step 4: Add num1 and num2 and assign the result to sum.
        sum←num1+num2
Step 5: Display sum
Step 6: Stop
```

## Algorithm 2: Find the largest number among three numbers

```
Step 1: Start
Step 2: Declare variables a,b and c.
Step 3: Read variables a,b and c.
Step 4: If a > b
           If a > c
              Display a is the largest number.
           Else
              Display c is the largest number.
        Else
           If b > c
              Display b is the largest number.
           Else
              Display c is the greatest number.
Step 5: Stop
```

## Algorithm 3: Find Roots of a Quadratic Equation $ax^2 + bx + c = 0$

```
Step 1: Start
Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;
Step 3: Calculate discriminant
         D ← b2-4ac
Step 4: If D ≥ 0
              r1 ← (-b+√D)/2a
              r2 ← (-b-√D)/2a
              Display r1 and r2 as roots.
        Else
              Calculate real part and imaginary part
              rp ← -b/2a
              ip ← √(-D)/2a
              Display rp+j(ip) and rp-j(ip) as roots
Step 5: Stop
```

## Algorithm 4: Find the factorial of a number

```
Step 1: Start
Step 2: Declare variables n, factorial and i.
Step 3: Initialize variables
          factorial ← 1
          i ← 1
Step 4: Read value of n
Step 5: Repeat the steps until i = n
     5.1: factorial ← factorial*i
     5.2: i ← i+1
Step 6: Display factorial
Step 7: Stop
```

## Algorithm 5: Check whether a number is prime or not

```
Step 1: Start
Step 2: Declare variables n, i, flag.
Step 3: Initialize variables
        flag ← 1
        i ← 2
Step 4: Read n from the user.
Step 5: Repeat the steps until i=(n/2)
     5.1 If remainder of n÷i equals 0
            flag ← 0
            Go to step 6
     5.2 i ← i+1
Step 6: If flag = 0
           Display n is not prime
        else
           Display n is prime
Step 7: Stop
```

---

## Algorithm 6: Find the Fibonacci series till the term less than 1000

```
Step 1: Start
Step 2: Declare variables first_term,second_term and temp.
Step 3: Initialize variables first_term ← 0 second_term ← 1
Step 4: Display first_term and second_term
Step 5: Repeat the steps until second_term ≤ 1000
     5.1: temp ← second_term
     5.2: second_term ← second_term + first_term
     5.3: first_term ← temp
     5.4: Display second_term
Step 6: Stop
```