- # Friend Function in C++

- Data hiding is a fundamental concept of object-oriented programming. It restricts the access of private members from outside of the class.
- Similarly, protected members can only be accessed by derived classes and are inaccessible from outside.

  For example,

```
class MyClass {
   private:
      int member1;
}

int main() {
   MyClass obj;

   // Error! Cannot access private members from here.
   obj.member1 = 5;
}
```

However, there is a feature in C++ called friend functions that break this rule and allow us to access member functions from outside the class.

If a function is defined as a friend function in C++, then the protected and private data of a class can be accessed using the function.
By using the keyword friend compiler knows the given function is a friend function.
For accessing the data, the declaration of a friend function should be done inside the body of a class starting with the keyword friend.

Declaration of friend function in C++

```
class class_name
{
   friend data_type function_name(argument/s);      // syntax of friend functio
n.
};
```

In the above declaration, the friend function is preceded by the keyword friend. The function can be defined anywhere in the program like a normal C++ function. The function definition does not use either the keyword **friend or scope resolution operator**.

Characteristics of a Friend function:

- The function is not in the scope of the class to which it has been declared as a friend.
- It cannot be called using the object as it is not in the scope of that class.
- It can be invoked like a normal function without using the object.
- It cannot access the member names directly and has to use an object name and dot membership operator with the member name.
- It can be declared either in the private or the public part.

Example 1: Working of friend Function

```
// C++ program to demonstrate the working of friend function

#include <iostream>
using namespace std;

class Distance {
   private:
      int meter;

      // friend function
      friend int addFive(Distance);

   public:
      Distance() : meter(0) {}

};

// friend function definition
int addFive(Distance d) {
```

```
    //accessing private members from the friend function
    d.meter += 5;
    return d.meter;
}

int main() {
    Distance D;
    cout << "Distance: " << addFive(D);
    return 0;
}
```

• friend class –

We can also use a friend Class in C++ using the friend keyword. For example,

```
class ClassB;

class ClassA {
  // ClassB is a friend class of ClassA
  friend class ClassB;
  ... .. ...
}

class ClassB {
  ... .. ...
}
```

When a class is declared a friend class, all the member functions of the friend class become friend functions. Since ClassB is a friend class, we can access all members of ClassA from inside ClassB. However, we cannot access members of ClassB from inside ClassA. It is because friend relation in C++ is only granted, not taken.

A friend class can access both private and protected members of the class in which it has been declared as friend.

Let's see a simple example of a friend class.

```
#include <iostream>

using namespace std;
  class A
{
    int x =5;
    friend class B;          // friend class.
};
class B
{
  public:
    void display(A &a)
    {
        cout<<"value of x is : "<<a.x;
    }
};
int main()
{
    A a;
    B b;
    b.display(a);
    return 0;
}
```

Output:

```
value of x is : 5
```

In the above example, class B is declared as a friend inside the class A. Therefore, B is a friend of class A. Class B can access the private members of class A.