Tutorial 26 - Friend Functions in C++

1. Definition

- Friend functions can access private and protected members of a class even though they are not members of the class.
- Declared using the keyword `friend` in the class.
- Writing the friend function prototype in the class does **not** make it a member of the class.

---

2. Example Code

Class with Friend Function

```
class Complex {
    int a, b;
    friend Complex sumComplex(Complex o1, Complex o2); // Friend function prototype
public:
    void setNumber(int n1, int n2) {
        a = n1;
        b = n2;
    }
    void printNumber() {
        cout << "Your number is " << a << " + " << b << "i" << endl;
    }
};
// Friend function definition
Complex sumComplex(Complex o1, Complex o2) {
    Complex o3;
    o3.setNumber((o1.a + o2.a), (o1.b + o2.b)); // Access private members using objects
    return o3;
}

```

Main Program

```
int main() {
    Complex c1, c2, sum;
    c1.setNumber(1, 4);
    c1.printNumber();
    c2.setNumber(5, 8);
    c2.printNumber();
    sum = sumComplex(c1, c2); // Friend function call
    sum.printNumber();
    return 0;
}

```

Output

```
Your number is 1 + 4i
Your number is 5 + 8i
Your number is 6 + 12i

```

---

3. Properties of Friend Functions

1. **Not in Class Scope**:
   - A friend function is not part of the class it is declared in.
   - Cannot be called using an object (`c1.sumComplex()` is invalid).
2. **Invoked Without an Object**:
   - Called directly like a normal function (`sumComplex(o1, o2)`).
3. **Access Private/Protected Members**:
   - Requires an object to access members (`o1.a`, `o2.b`).
4. **Declared Anywhere in Class**:
   - Can be declared under public or private sections; placement does not affect access.
5. **Uses Objects as Arguments**:
   - Typically operates on objects passed to it.

---

4. Short Notes

Friend Function Key Points

1. Allows access to private/protected members.
2. Declared in the class using `friend`.
3. Defined outside the class like a normal function.
4. **Example Syntax**:

```
class ClassName {
    friend ReturnType FunctionName(Arguments);
};

```

Properties

1. Not in the class scope.
2. Cannot be called using an object.
3. Invoked like a normal function.
4. Can be declared in public or private sections.
5. Needs an object to access members (`object_name.member_name`).

Example Workflow

- **Declare Friend**: `friend Complex sumComplex(Complex o1, Complex o2);`
- **Define Friend**: Access members using passed objects.
- **Call Friend**: Pass objects to perform operations.