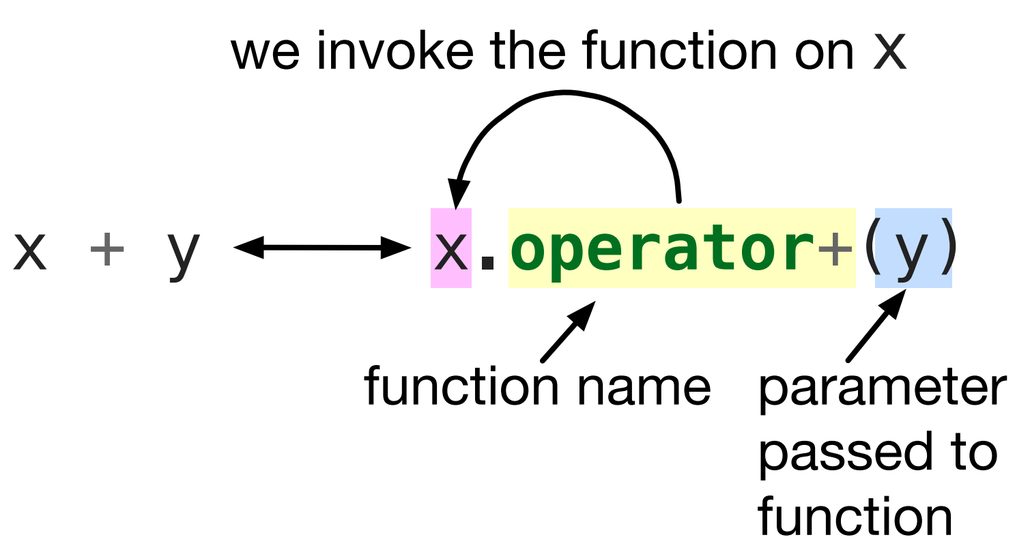


```
class Complex {
    private:
        double real;
        double img;
public:
    Complex() {
        real = 0; img = 0;
    }
    Complex(double r, double i) {
        real = r; img = i;
    }
```

return type *function name* *input type* *object name (is `y` passed to `rhs`)*

```
    Complex operator+(Complex rhs) {
        Complex temp;
```

real of object on which ***operator+*** *is invoked on (or x)*

real of rhs (or y) passed to **operator+**

```
        temp.real = real + rhs.real;
        temp.img = img + rhs.img;
```

return by value

```
        return temp;
    }
};
```

Pass by reference to be **memory efficient**

```
Complex operator+(const Complex& rhs) const {
```

if **`rhs.real = 0;`** *is written in the function, you'll get a compile-time error!*

if **`real = 0;`** *is written in the function, you'll get a compile-time error!*

```
    Complex temp;
    temp.real = real + rhs.real;
    temp.img = img + rhs.img;
    return temp;
}
```