Als Methode oder als Funktion?

Oftmals kann man etwas als Objektmethode oder als klassische Funktion oder als `friend`-Funktion realisieren. Dann ist es eine Frage des persönlichen Stils, was man bevorzugt. Ein paar Fälle, in denen die optimale Wahl eindeutig ist:

1) Es gibt schon Funktionen mit dem gleichen Namen für andere Typen (z.b Absolutbetrag von `double`). Hier sollte man die gleiche Syntax nehmen, denn `abs(x) + z.abs()` sieht nicht gut aus.
2) Das Klassenelement steht nicht an der ersten Stelle eines Operators (z.B. `cout << z` entspricht `operator<<(cout, z)`. Eine solche Funktion kann man nicht als Methode für komplexe Zahlen `z` definieren. Das Referenzelement eines Operators als Methode ist immer der erste Methoden-Parameter.
3) Man muss auf private Bestandteile des Objekts zugreifen. Objektmethoden dürfen das immer, Funktionen nur, wenn sie als Freund (`friend`) des Objekts in der Objektdefinition deklariert (oder definiert) sind.
4) Man kann von Typ-Umwandlungen in den Objekttyp profitieren. Das ist eher bei einer Funktion als bei einer Methode möglich:

zu 4): Es gebe eine automatische Umwandlung von `double` in `Komplexe_Zahl` (die gibt es bei uns noch nicht, kommt aber bald) und wir wollen den Additionsoperator + (d.h. `operator+()` definieren und damit `a+b` berechnen. Als Objekt-Methode wäre das

```
Komplexe_Zahl operator+(Komplexe_Zahl b) const
{    return { re + b.re, im + b.im; } };
```

Als Methode ist das Referenzelement kein Argument, d.h. es steht oben nur der 2. Summand `b` in der Klammer `()`. Aufrufen kann man diese Methode:
Langform: `a.operator+(b)`
Kurzform: `a + b;`

Als Funktion sieht das so aus:

```
Komplexe_Zahl operator+(Komplexe_Zahl a, Komplexe_Zahl b)
{    return { a.re + b.re, a.im + b.im; } };
```

Diese Definition ist schöner, weil sie bezüglich `a` und `b` symmetrisch ist. Aufrufen kann man diese Methode :
Langform: `operator+(a, b)`
Kurzform: `a + b;`

Sind Real-/Imaginärteil private Datenattribute von `Komplexe_Zahl`, so muss man diese Funktion noch als `friend`-Funktion innerhalb von `Komplexe_Zahl` deklarieren.

Da man nur die Kurzform verwenden wird, sind beide Varianten (als Methode bzw. als Funktion) beim Aufruf optisch gleich, aber:
`a + 1.` : Bei beiden Varianten wird `1.` in `Komplexe_Zahl` gewandelt und addiert
`1.` + `b` : Fehler bei der Methode, ok bei der Funktion.

Bei der Methode ist der linke Operand eigentlich das Referenzelement (das wird bekanntlich NIE umgewandelt) und muss den richtigen Typ haben. Bei der Funktion können beide Argumente umgewandelt werden.

Es wird folgerichtig empfohlen, binäre Operatoren als (ev. befreundete) Funktionen und nicht als Methoden zu programmieren, weil nur so die Symmetrie bezüglich der Operanden vorhanden ist.