4 EXPRESSION TYPES

Sections

- 4.1 Types
- 4.2 Real Numbers
- 4.3 Complex Numbers
- 4.4 Extended Numerics
- 4.5 Boolean Expressions
- 4.6 Multiple Expressions
- 4.7 Names
- 4.8 Miscellaneous
- 4.9 Application: Looking for Something Better
- Summary
- Key Terms
- Problems

Objectives

After reading this chapter, you should be able to:

- Classify Maple's expression types
- Identify an expression's surface and/or structured type
- Build expressions using different types of numbers
- Build expressions using Boolean values and relations
- Build expressions consisting of collections of other expressions
- Label and extract elements from indexed lists, sets, sequences, and indexed names

4.1 Types

In Chapter 3, you used Maple's tokens, like names and operators, to build relatively basic expressions. As you solve more complicated models, your expressions will combine tokens in different ways to produce a wider variety of expressions. Maple classifies all expressions according to their mathematical structure. This chapter provides an overview of this classification scheme to help you build expressions that commonly arise in solving science and engineering problems. For online overviews, you should also investigate Programming…Data Types….

4.1.1 Type Definition

Maple defines a classification of an expression as a ***type***. According to **`?type[definition]`**, a type is a Maple expression recognized by the **`type`** function. (Yes, I know that sounds like circular reasoning.) For now, think of a Maple type as a common classification to which an expression may belong. Sometimes an expression fits under multiple categories. For instance, the expression **`72`** classifies as **`integer`**, but you could also choose the types **`positive`** and **`posint`**, according to Maple. In general, every Maple expression has at least one type.

4.1.2 Checking a Type

How does the `type` function work? To test the type ***`type`*** of expression ***`expr`***, enter `type(`***`expr`***`,`***`type`***`)`. Maple will report *true* or *false*, depending on whether or not ***`expr`*** classifies under ***`type`***. For example, see if Maple knows that 72 is an integer:

Step 48: The `type` Function

```
> type(72,integer);                    Check if the expression 72 is an integer.
                        true           Yes, 72 is an integer.
```

You might be wondering how you are supposed to know all the types Maple offers. Look at `?type` to discover the complete list, but don't worry! You do not need to memorize all of the types – just remember how you found the list.

4.1.3 Type Classifications

Suppose you wish to check the expression $\sqrt{a+b}$. Does this expression classify as addition, function, or a power? You need to guide Maple a bit. Using types, Maple classifies expressions in three manners using the types found in `?type`, described below: surface, structured, and nested.

4.1.4 Surface Type

A ***surface type*** is a the uppermost element, or top node, in an expression tree. Since operators may be the top node, Maple include them as possible surface types. For instance, the surface type of $\sqrt{a+b}$ is a power, which Maple signifies as a `` `^` ``. When a type is an operator, you must use backquotes so Maple doesn't think you're trying to use that operator! Other surface types include more general classifications, like `even` and `mathfunc`. For a full list of surface types that Maple recognizes, consult `?type[surface]`.

4.1.5 Structured Type

A ***structured type*** is a type built from individual types in an expression. Forming a structured type resembles building an expression tree with only the types of the individual nodes. For instance, the structured type of a^2 is `name^integer`. Finding the structured type for $\sqrt{a+b}$ is left as a homework problem. For a listing of rules for forming a structured type, consult `?type[structured]`.

4.1.6 Nested Type

A ***nested type*** is a type that Maple looks for within the entire expression. For instance, the expression $a+1$ is not `constant`. But, Maple considers $1+\sqrt{2}$ `constant` because all elements are constant. Maple provides the brief list of nested types along with surface types in `?type[surface]`.

4.1.7 Finding Types

What if you want Maple to tell you the type of an expression? Maple will tell you the surface type of an expression with `whattype(expr)`. For instance, ask Maple to report the type for `a+b*c`:

Step 49: Reporting Surface Type

```
> restart;                                        Refresh your worksheet.

> whattype(a+b*c);                  Query Maple about the type for a+bc.
                      +                     Maple reports the surface type +.
```

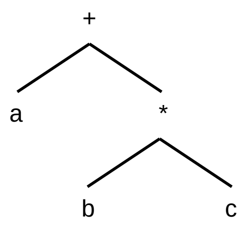


Figure 4-1 Expression Tree for $a+bc$

Why did Maple report the addition operator (`+`)? In the expression tree, the plus sign is the top node that connects the subexpressions `a` and `b*c`, as shown in Figure 4-1. For more information and related functions, consult `?whattype`, `?op`, and `?hastype`. Now that you know how to find and check types, you will explore the types Maple offers throughout the rest of this chapter.

Practice!

1. Does Maple consider the surface type of the expression $\sin(x)$ as `function`?
2. How do you check the type for the expression $a+b$? Hint: Investigate `?type[arithop]` and `?arithop`.
3. Determine the structured type of the expression a^{2^b}.
4. How might the expression types `anything` and `type` help you test expressions? Hint: See `?type[type]`.
5. Determine the surface and structured expression types for $\sin(x+y)$.
6. Enter `whattype(1+2)`. Why is the expression type not +?

4.2 Real Numbers

Most forms of engineering and science measure quantities with real numbers. Thankfully, Maple enumerates numerous numerical number types reviewed by this section. Experienced users seeking detailed information and background on numerical computations should investigate **?numerics**.

4.2.1 Finding Number Types

When you want to determine the type of number in Maple, use **NumericClass(*expr*)**. Maple will attempt to identify the classifications that ***expr*** might fit. For example, Maple knows that 10 is a positive integer:

Step 50: Identifying Number Types

```
> NumericClass(10);                    Query Maple about the type of the expression 1 .
                    posint             Maple determines that 1  is a positive integer.
```

All numerical types described in this chapter are known by **NumericClass**. For a full listing of numeric types, consult **?numeric_types**.

4.2.2 Integers

As introduced in Section 3.2.2, integers are exact, whole numbers written as a series of digits with no decimal point.

- *Natural integers* are strictly positive.
- *Signed integers* have positive or negative signs.

For example, 72 and 1216 are integers, whereas 72.0, 0.01, and $\sqrt{2}$ are not integers. Maple prefers integers because integer operations produce exact results:

Step 51: Integers

```
> 1+10+111;                    Add integers together. Do not use decimal points.
                    111        Maple retains exact values whenever possible.
```

Consult **?integer** and **?type[integer]** for more information. You will find integer-related functions in Mathematics...Numbers...Integer Functions....

4.2.3 Fractions

Maple defines ***fractions*** as numbers of the form $\frac{signed\ integer}{natural\ integer}$, where the signed integer is divided by the natural integer. You may also call the “top-value” the *numerator*, and the “bottom-value” the *denominator*. Because a fraction involves division, enter a fraction using the syntax ***numer/denon***. If you give a negative denominator, Maple will automatically move the minus sign to the numerator to maintain fractional form:

Step 52: Fractions

```
> 2/(-3);                    Divide two integers to produce a fraction.
```

$$\frac{-2}{3}$$

Maple converts the expression to fractional form.

Why does Maple leave a fraction undivided? Maple strives to maintain exactness whenever possible. So, expressions that cannot be automatically simplified are left untouched. For more information and related functions, consult `?fraction` and `?type[fraction]`.

4.2.4 Rational Numbers

Rational numbers include both integers and fractions. The word *rational* arises from the notion of *ratio* of values. Maple automatically simplifies rational numbers by removing common factors. As with fractions, Maple will maintain exactness and avoid dividing fractions that might introduce non-rational results. In the following example, Maple can use automatic simplification without any problem:

Step 53: Rational Numbers

```
> 1/3 + 1/3 + 1/3 - 1;                         Add rational numbers together.
                         0                     Maple simplified the expression.
```

For more information and related functions, consult `?fraction` and `?type[rational]`.

Practice!

7. Does Maple simplify the expression $\frac{72}{42}$?

8. Will entering `1./3.+1./3.+1./3.` produce integer output 1? Why or why not?

4.2.5 Floating-Point Numbers

Since the "real world" cannot always provide exact numerical values, numerical analysis relies on ***floating-point numbers***, or just *floats*, to measure quantities. Floats are base-10 or *decimal* numbers, like 11., 10.1, and 0.01. Maple considers a number input with a decimal point (`.`) as a float, but you may enter floats in other ways, as shown in Table 4-1.

Table 4-1 Entering Floats

Syntax	Example Input	Example Output
int.int	`100.1`	100.1
.int	`.01`	.01
int.	`10.`	10.
int.intEint	`1.2E3`	1200.
intEint	`1E2`	100.
`Float(`*x,y*`)`	`Float(1.2,-3)`	.0012

When using `E`, you may interchange `e` for `E`. The `E` and `e` notation abbreviates the function `Float(x,y)`, which represents a float in scientific-notation $x \times 10^y$, given *mantissa* x and *exponent* y. When reporting "big" numbers, Maple will show them in scientific notation without the multiply symbol (×) in the output. For instance, enter 1 million, where $1000000 = 1 \times 10^6$:

Step 54: Scientific Notation

```
> Float(1,6);
                         .1 10^7
```

You could also enter `1E6` or `1e6`.

Maple calculates $1 \times 10^6 = 0.1 \times 10^7$.

Have you wondered why floats aren't included as Maple tokens? Maple treats all floats that you enter as a pair of integers, mantissa `x` and exponent `y`, using scientific notation. For more information on floats and additional background, consult `?float`, `?type[float]`, and `?type[numeric]`.

4.2.6 Floating-Point Arithmetic

Arithmetic operations with floats produce floats:

Step 55: Floats

```
> Float(-1,2) - 20. - 3.0 - 0.4 - 5E-2 - 6.0E-3 - 7e-4;
                         -123.4567
```

The floats above demonstrate the variety of forms.

Maple automatically converts integers to floats when types are mixed:

Step 56: Mixing Floats with Integers

```
> 0.5 + 1/2;
                         1.000000000
```

Add the float 0.5 to the rational number $\frac{1}{2}$.

Mixing floats and rational numbers produces floats.

Beware that mixing floating-point numbers with other types tends to force all evaluations to decimal form.

Practice!

9. Express 123.0 in Maple floating-point notation with three different syntaxes.

10. Why does entering `123*10^(-1)` yield a rational number? Change the input to produce a floating-point answer.

4.2.7 Evaluating Floats

Suppose you wish to compute a floating-point result from an integer operation. Remember that Maple keeps calculations exact if possible, so dividing 1 by 2, for example, will not yield a float:

Step 57: Integer Division

```
> 1/2;                                                  Divide 2 by 1.
                         1
                         -                Integer division produces a fraction, not a float.
                         2
```

So, how do you force Maple into producing a float? Either enter one of the numbers as float, since mixing a float with integers produces a float:

Step 58: Produce Float Using a Decimal

```
> 1/2.0;                                                Divide 1 by the float 2.0.
                    .5000000000                Mixing floats and integers produces floats.
```

Or, you may `evalf(expr)` ("evaluate float") to force a floating-point computation:

Step 59: Produce Float Using `evalf`

```
> evalf(1/2);                            Evaluate the floating-point result of 1/2.
                    .5000000000                                  evalf produces a float.
```

4.2.8 Digits

How does Maple know how many digits to use and display for floats? Maple predefines a variable called `Digits` that determines the number of digits Maple uses in floating-point evaluations. By default, Maple sets `Digits` to 10, which explains why you have seen upwards of 10 digits in floating-point output. To see the default value of digits, enter `Digits`:

Step 60: Checking `Digits`

```
> Digits;                                            Display Digit's current value.
                         10          Assuming no previous altering, 10 is the default value.
```

To change the number of digits Maple uses, assign to `Digits` a new value:

Step 61: Changing `Digits`

```
> Digits:=3:                              Change the number of digits Maple uses to 3.
```

```
> 1/2.0;                                                Divide 1 by the float 2.0.
                        .500                    Maple used only 3 digits to compute 1/2.
```

To reset the value of `Digits`, reassign the value of 10:

Step 62: Reset `Digits`

```
> Digits:=10:                                              Reset the old value of 10.
```

Usually, you should adjust `Digits` when you have many computations to perform. When you wish to evaluate a single expression, you should choose `evalf(`*`expr`*`,`*`n`*`)`, which sets `Digits` to *`n`* only during the evaluation of *`expr`*. Note that *`n`* must be less than or equal to `Digits`.

Step 63: Evaluate to n Digits

```
> evalf(1/2,3);                    Evaluate the floating-point result of 1/2 to 3 digits.
                     .500                          evalf produces a float with only 3 digits.
```

For more information, consult `?evalf` and `?Digits`. `Digits` is an ***environment variable***, which is a predefined name that can affect the behavior of evaluation. You may read more about `Digits` and other environment variables in `?environ` and Appendix E.

4.2.9 Rounding

What if your input's amount of digits exceeds `Digits`? Maple will evaluate the expression with every digit you have entered and *round* the evaluated result. Rounding involves approximating a numerical value with one that is relatively "close." But what does Maple do when you have a rounding "conflict?" A common conflict happens when Maple encounters the digit 5 in a number that must rounded up or down. For example, try the following input statements which require 1 digit for evaluation:

Step 64: Rounding Mystery

```
> evalf(0.05+0.4,1);                     Evaluate the result of 0.05+0.4 to one digit.
                      .4                            Maple rounded 0.45 down to 0.4 .

> evalf(0.05+0.5,1);                     Evaluate the result of 0.05+0.5 to one digit.
                      .6                              Maple rounded 0.55 up to 0.6 .
```

Maple does not round $0.05 + 0.4$ or $0.05 + 0.5$ to 0.5 in either case. Why? Maple rounds to the nearest even value by default. You may confirm this rounding scheme by investigating the environment variable `Rounding`:

```
> Rounding;                                  Check Maple's current rounding scheme.
                    nearest         Maple rounds to the nearest even value for a conflict.
```

You may change the rounding method by assigning one of the alternatives listed in `?Rounding`. For related rounding functions, investigate `?trunc`.

Practice!

11. Evaluate 1/3 to 4 digits.

12. Why does the input `Digits:=2: 1.05+1.05;` produce the output 2.0?

4.2.10 Irrational Numbers

Irrational numbers cannot be expressed as a ratio of two integers. When expressed as floats, irrational numbers have non repeating decimals with no determined bounds. Some examples of irrational numbers include $\sqrt{2}$, π, and the exponential constant e. To maintain exactness, Maple will typically "refuse" to generate a floating-point result and, instead, leave an irrational quantity in symbolic form. For instance, Maple the input `sqrt(2)` is irrational:

Step 65: Display Irrational Number

```
> sqrt(2);                                          Attempt to evaluate √2.
                          √2          Maple cannot further reduce the irrational value √2.
```

To evaluate a floating-point approximation of `sqrt(2)`, you may enter `sqrt(2.0)`:

Step 66: Compute Irrational Number

```
> sqrt(2.0);                         Evaluate a numerical approximation of √2.
                     1.414213562     The float 2.0 forces a floating-point evaluation.
```

Or, use `evalf` by entering `evalf(expr)`:

Step 67: Compute Irrational Number

```
> evalf(sqrt(2));                    Evaluate a numerical approximation of √2.
                     1.414213562             evalf forces a floating-point evaluation.
```

4.2.11 Pi

A famous – and very useful – irrational number used throughout science and engineering is ***Pi***, which has the value 3.1415…. You will often see Pi written as the Greek character π. Maple implements π as the predefined name `Pi`:

Step 68: Numerical Pi

```
> Pi;                               Maple defines Pi as the numerical value π.
                          π                  Maple automaically displays Pi as π.
```

To show the numerical value, use `evalf`:

Step 69: Evaluate Numerical Pi

```
> evalf(Pi);                        Maple defines Pi as the numerical value π.
                     3.141592654             Maple automaically displays Pi as π.
```

Remember that Maple is case sensitive! A common mistake new users make is entering `pi` instead of `Pi`, because Maple outputs both as π:

Step 70: pi is Not the Numerical Pi!

```
> pi;                                    This input is lowercase pi, not Pi!
                    π                    The lowercase Greek name π is pi.
```

```
> evalf(pi);                  What happens if you attempt to evaluate pi?
                    π                 You do not obtain the numerical Pi!
```

As discussed in Section 4.7.2, Maple calls **Pi** a *symbolic constant*, which is a predefined name used as a value in expressions.

Practice!

13. Classify the following expressions as rational or irrational: $\sin(0)$, 10.1, $\sqrt{2}$, e.

14. Evaluate π to five digits.

15. Compute $4^{\frac{1}{3}}$ in both exact and floating-point forms.

4.3 Complex Numbers

Can you take the square root of a negative number? Technically, no, but some physical quantities are negative due to sign conventions. To resolve this dilemma, imaginary values were conveniently invented to represent square roots of negative numbers which often arise in mathematical analysis. How imaginary is an imaginary value? See below....

4.3.1 Definitions

To handle negative square roots, the name i is defined as follows:

$$i = \sqrt{-1}. \tag{4-1}$$

Therefore, $i^2 = -1$. So, wherever you see a negative square root, write i instead, as in $\sqrt{-2} = \sqrt{2}i$. In general, a number in the form $\sqrt{-n}$ is written as ni and called a *purely imaginary number*. Adding a purely imaginary number to a real number, as in $a + bi$, creates a *general complex number*. For instance, to express $4 - \sqrt{-3}$, you would write as $4 - 3i$. The term ***complex number*** refers to both purely imaginary and general complex numbers. Consult **?complex**, **?type[imaginary]**, and **?type[complex]** for more information.

4.3.2 Arithmetic

When you need $\sqrt{-1}$, click on i from the symbol palette or enter **I**. Maple will show i as uppercase I.

Step 71: Complex Number I

```
> I;                                          Enter i using Maple's I.
                    I                         Maple represents √-1 = i as I.
```

To create a complex number or general expression in the form $a+bi$, you would enter **a+b*I**. You can use complex expressions in most of Maple, which will attempt to evaluate both real and imaginary components. For instance, try one of the arithmetic operations summarized in Table 4-2.

Table 4-2 Complex Arithmetic

Operation	Rule
addition	$(a+bi)+(c+di) = (a+c)+(b+d)i$
subtraction	$(a+bi)-(c+di) = (a-c)+(b-d)i$
multiplication	$(a+bi)(c+di) = (ac-bd)+(bc+ad)i$
division	$\frac{a+bi}{c+di} = \frac{ac+bd}{c^2+d^2}+\frac{bc-ad}{c^2+d^2}i$

Step 72: Complex Arithmetic

```
> (1+2*I) + (2+I);                            Add two complex numbers.
                    3 + 3I                    (1+2i)+(2+i) = (1+2)+(2+1)i = 3+3i.
```

In earlier releases, Maple implemented **I** as an *alias* (see **?alias**), but now **I** automatically calls **Complex(1)**, which represents i. You may enter an imaginary number ***a**I** as **Complex(*a*)** and a complex number ***a*+*b**I** as **Complex(*a*,*b*)**, if you prefer:

Step 73: Complex Numbers using Complex

```
> Complex(1) + Complex(1,3);                  Add two complex numbers.
                    1 + 4i                    (0+i)+(1+3i) = (0+1)+(1+3)i = 1+4i.
```

For more information, consult **?I** and **?complex**. Adventurous students who want to change Maple's implementation of i should consult **?interface**.

4.3.3 Functions

For functions that relate to complex numbers, check out **`?Re`**, **`?Im`**, and **`?evalc`**, listed with other functions are listed in Mathematics...Numbers...Complex Numbers.... These functions are reviewed in later chapters.

Practice!

16. Do the inputs **`sqrt(-1)`**, **`(-1)^(1/2)`**, and **`I`** produce different outputs?

17. Evaluate $(1+2i)(-1-i)$.

4.4 Extended Numerics

This section describes some handy commands that relate to numerical analysis in Maple. Consult Mathematics...Numerical Computations...Maple Numerics... for a broad listing of numerical features that Maple offers.

4.4.1 Infinity

Many calculations require that you use a really, really big number called ***infinity***, denoted as ∞. Maple's extended numerics consist of the ability to represent ∞ with the symbolic constant **`infinity`**.

Step 74: `infinity`

```
> infinity;                                  Evaluate the symbolic constant infinity.
                          ∞                  You may also use the symbol palette to enter ∞ .
```

Of course, adding a number to ∞ cannot make ∞ any bigger, and Maple agrees:

Step 75: Operations with `infinity`

```
> infinity + 32;                                                    Evaluate ∞+32 .
                          ∞                             Maple can perform operations with ∞ .
```

For more information, consult **`?infinity`**, **`?type[infinity]`**, and **`?type[numeric]`**.

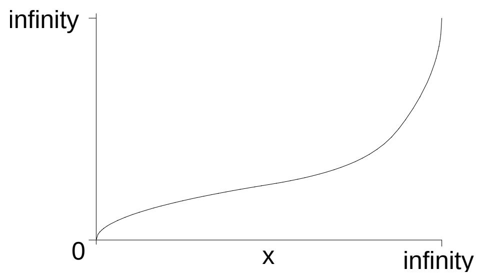
4.4.2 Example

Why should you be concerned with infinity? One handy application is the generation of "general" plots without having to specify numerical ranges. For instance, if you wish to investigate the behavior of the square function $\sqrt{x}$ for all positive numbers, you could plot the function for the range $0 \le x \le \infty$. In Maple, you would do the following:

Step 76: Plotting Infinity

```
> plot(sqrt(x),x=0..infinity);
```

Plot $\sqrt{x}$ for $0 \le x \le \infty$.



4.4.3 Undefined

As an amusing thought exercise, consider what subtracting infinity from infinity might mean. Should the result be zero, and if so, what guarantee do you have a really big number is the same "size" as another? Maple sidesteps this issue by using the predefined name `undefined` for operations that cannot produce a number:

Step 77: `undefined`

```
> infinity - infinity;
                         undefined
```

Attempt to evaluate $\infty - \infty$.

Maple reports undefined as a "value."

Maple classifies number that contain `infinity` and/or `undefined` as ***extended numerics***. Chances are you won't confront `undefined` too often, but it does propagate if you perform unrecognized operations with ∞. In fact, you may actually use `undefined` as a kind of value that you operate on, as discussed in `?undefined` and `?type[undefined]`.

4.4.4 Complex Extended Numerics

Maple describes handling infinite complex numbers in `?complex`.

Practice!

18. How does Maple treat division by zero?

19. Plot $\exp(x)$ for $-\infty \le x \le \infty$.

4.5 Boolean Expressions

Maple provides more than just numerical tools – you can create expressions that incorporate logic and comparison relationships. These expressions are called ***Boolean*** because they deal with the truth of an expression. This section demonstrates that component types that Boolean expressions use.

4.5.1 Logic

You can create ***logical*** expressions by combining expressions with the operators **and**, **or**, and **not**. See Table 4-3 for the Standard Math notation.

Table 4-3 Logical Operators

Operator	Standard Math	Maple Notation
and	$\wedge$	**and**
or	$\vee$	**or**
not	$\neg$	**not**

For instance, suppose you want to store the expression $a \wedge b$ in the variable x:

Step 78: Logical Expression

```
> restart;                                   Refresh your worksheet.
```

```
> x := a and b;                              Assign to x the expression a∧b
                    x := a and b             Remember that and is an operator.
```

So, what can you do with a logical expression? Logical operators act upon the logical values *true* and *false* which Maple implements as the symbolic constants **true** and **false**, respectively. For example, what happens if you assign *true* to both *a* and *b*? Logic dictates that $true \wedge true = true$. For example, how might you test if Maple considers the number 10.1 both floating-point and numeric? Use the **type** function which reports *true* or *false* when testing an expression:

Step 79: Using a Logical Expression

```
> a := type(10.1,float):          Assign to a the result of checking if 10.1 is a float.
> b := type(10.1,numeric):        Assign to a the result of checking if 10.1 is numeric.
```

```
> x;                                         Evaluate x.
                    true                     Maple reports that a∧b evaluated to true.
```

In the above step, Maple evaluated *a* as *true* and then *b* as *true*. Then, after you entered x, Maple evaluated x as a **and** b, which evaluated to $true \wedge true$, which simplified to *true*. Table 4-4 summarizes the full list of rules for combining *true* (T) and *false* (F) found in **?boolean**.

4.5.2 Relational

Logical values commonly arise when you compare values using ***relations***. A relation is an expression that compares two expressions by using a relational operator ***rel*** with the syntax ***expr1 rel expr2***. Table 4-5 summarizes common relational operators.

Table 4-4 Logical Operations

$\wedge$	T	F
T	T	F
F	F	F

$\vee$	T	F
T	T	T
F	T	F

$\neg$	
T	**F**
F	**T**

Table 4-5 Relational Operators

Operator	**Standard Math**	**Maple Notation**	**Example**
Equal	$=$	`=`	`1=1`
Not Equal	$\neq$	`<>`	`1<>2`
Greater Than	$>$	`>`	`2>1`
Greater Than or Equal	$\geq$	`>=`	`2>=1`
Less Than	$<$	`<`	`1<2`
Less Than or Equal	$\leq$	`<=`	`1<=2`

Maple automatically reverses "greater than" inequalities into "less than" form. For example, try storing the relation $1 > 0$ in the name REL:

Step 80: Store Relation

```
> REL := 1>0;                         Store the expression 1>0 inside REL.
                 REL := 0 < 1         Maple automatically converts ">" into "<" form.
```

Maple stored the equivalent relation $0 < 1$, instead. Note that REL now refers to the entire relation $0 < 1$:

Step 81: Check Relation

```
> type(REL,relation);                 Check if REL stores an expression of type relation.
                 true                 Yes, REL is a relation.
```

You will see how to evaluate a relation in Section 4.5.4. For more information about relations and their operators, consult `?inequality` and `?operators[binary]`.

4.5.3 Equation

An ***equation*** is a relation that uses the equals operator (`=`). So, an equation has the syntax ***`expr1 = expr2`***. Remember that the equal sign (`=`) does not assign expressions! Instead, use equations for comparison and solving, not for assignment:

Step 82: Assign Equations

```
> restart;
```
Refresh your worksheet.

```
> EQN := y=m*x+b;
```
Assign to the name EQN an equation.

$$y = mx + b$$

EQN was assigned to the expression $y = mx+b$.

```
> EQN;
```
Show the value of EQN.

$$y = mx + b$$

Yes, EQN really has the value $y = mx+b$.

```
> y;
```
Show the value of y.

$$y$$

The equals sign does not assign!

Maple uses equations very often. For instance, to solve for x in the equation $y = mx + b$, you could use the `solve` function. Entering `solve(eqn,var)` solves equation ***eqn*** for variable ***var***:

Step 83: Solving Equations

```
> solve(y=m*x+b,x);
```
Solve $y = mx+b$ for x.

$$\frac{y-b}{m}$$

$x = \frac{y-b}{m}$.

x is not assigned unless you enter `x:=solve(EQN,x)`.

Equations are discussed further in `?equation` and `?type[equation]`. See `?solve` for more information about solving equations.

4.5.4 Boolean

Expressions that contain relational operators (`=`, `>`, `>=`, `<`, `<=`, `<>`) or logical operators (`and`, `or`, `not`) are called ***Boolean***. Thus, logical and relational expressions also classify as Boolean, and consequently, Boolean expressions involve tests and conditions. Boolean expressions evaluate to *true*, *false*, and a third symbolic constant called *FAIL*, using rules of logic. Boolean operations return *FAIL* when they cannot determine the truth of an expression. Maple performs a Boolean evaluation in the following situations:

- an expression contains a logical operator: `and`, `or`, or `not`
- you perform a test inside an `if` or `while` clause: see Appendix C.
- you enter `evalb(expr)` to evaluate the Boolean value of ***expr***.

For example, experiment with the following steps:

Step 84: Boolean Evaluation

```
> 1>2 or 1<2;
```
Ask Maple if 1 is greater than or less than 2.

$$true$$

Maple evaluates your Boolean expression as true.

```
> evalb(1>2);
```
Evaluate the Boolean value of the expression $1>2$.

$$false$$

Maple evaluates your Boolean expression as $false$.

When using `evalb` beware that

- Maple will not simplify `expr` except for automatic simplifications.
- `evalb` does not assign expressions and only produces *true*, *false*, and *FAIL*.

For information, consult `?boolean`, `?evalb`, `?type[boolean]`, and `?FAIL`.

Practice!

20. Evaluate $\neg((true \vee false) \wedge true)$.

21. Solve for x in the equation $ax^2 + bx + c = 0$, using the `solve` function.

22. Evaluate the truth of the relation $10 + 2 \geq 12$.

23. Set `Digits` to 2. next, evaluate the Boolean value of the relations $1.25 < 1.3$ and $1.35 < 1.4$. Why does Maple report *false* and *true*, respectively?

24. Why does entering `evalb((x^2-1)/(x-1) = (x+1))` evaluate to *false*? Hint: Does Maple automatically “know” that $x^2 - 1 = (x+1)(x-1)$?

4.6 Multiple Expressions

Many models require you to analyze and design parameters for different values. For such situations when you need to collect expressions together, Maple provides many types that can represent multiple expressions. This section reviews common types. For more information, refer to Appendix C and `?exprseq`.

4.6.1 Ranges

A ***range*** is a continuous interval of values between, and including, the endpoints. You may express a range r and the endpoints as $[a, b]$. You may also think of r as all values of a variable x such that $a \leq x \leq b$ is true. Maple represents the range r as the expression `a..b`, using the .range operator `..`. You will commonly find continuous ranges in plotting:

Step 85: Continuous Ranges

```
> plot(x^2,x=0..100):
```
Plot x^2 *for the range* $0 \leq x \leq 100$.

```
                              Plot not shown to conserve space.
```

You may also generate a range of non-continuous, or discrete, values using the syntax `$ a..b`:

Step 86: Discrete Ranges

```
> $ 1..3;                          Generate values between, and including 1 and 3.
                1, 2, 3            Maple produces a non-continuous range.
```

This "non-continuous," or *discrete*, range is called a *sequence*, which is discussed below. For more information about applications of the range operator, consult `?range`, `?plot[range]`, `?type[range]`, and `?$`.

4.6.2 Sequences

A ***sequence*** is a collection of items placed in a specific order. In terms of Maple, I'm technically "breaking the rules" a bit by introducing *expression sequences* in this chapter because they are formally *not* types. But, there are many occasions when you need to store a collection of expressions together in a variable. To do so, you may "glue" expressions together by separating each expression with commas (`,`), as in `expr,expr,...`, which forms a sequence. For example, try storing the sequence of expressions $1, 2, \sqrt{2}$ in a variable Seq:

Step 87: Discrete Ranges

```
> Seq:=1,2,sqrt(2);                Store an expression sequence in Seq.
            Seq := 1, 2, √2        Seq holds all the values in the specified order.
```

Each item inside a sequence is called an *element*.

You might get confused if you enter `whattype(Seq)`, because Maple will report *exprseq*, which seems to imply that *exprseq* is a type. But, if you enter `type(Seq,exprseq)` Maple will generate an error message! Why? Expressions inside a sequence may have a different type, so Maple wants to "avoid" possible confusion. For further explanation, consult `?type` and the overview of sequences in `?exprseq`. You will find information about related functions in `?seq`, `?$`, `?map`, and `?op`.

4.6.3 Lists

A ***list*** is an expression that contains a sequence of expressions. To create a list, surround a sequence with square brackets (`[]`). For example, suppose you wish to collect a list of numbers and names in a list called L:

Step 88: Lists

```
> List := [x1,23,-3,x2,x1];              Assign a list by surrounding a sequence with ([]).
        List := [x1, 23, –3, x2, x1]     You may mix expression types in a list.
```

As in the above example, you may repeat list elements. For more information and related functions, consult `?list`, `?type[list]`, `?op`, and `?map`.

4.6.4 Sets

Sets are expressions that collect *unordered unique* items. Create a set by surrounding a sequence with curly braces (`{}`). In the following example of assigning a set, Maple will remove duplicated elements to maintain each element's uniqueness:

Step 89: Sets

```
> Set := {e1,e2,e2,e3};                                Assign to the name Set a set.
                        {e1, e2, e3}          Maple removes repeated elements in creating a set.
```

For information about sets and related operations, consult `?set`, `?type[set]`, and `?intersect` or `?union`.

4.6.5 Selection

Besides expressing lists, square brackets (`[]`) assist with indexing and extracting elements from sequences, lists, and sets. Maple counts each expression inside a sequence, list, and set from left to right starting from 1. So, given a sequence input `Seq`, entering the expression `Seq[i..j]` extracts the *i*th to *j*th expressions from `Seq`:

Step 90: Selection

```
> restart:                                                     Refresh the worksheet.
> Seq:=a,b,c,d,e:                           Assign to Seq the sequence a,b,c,d,e .
> Seq[1..3]; Seq[5];     Extract the first through third elements. Then, extract the fifth element.
                         a, b, c      Maple extracted the 1st, 2nd, and 3rd elements from Seq .
                           e                            Maple extracted the 5th element of Seq .
```

For an extensive overview, consult `?selection`.

Practice!

25. What does the input statement `'i^2' $ 'i'=0..3` indicate and produce? Hint: Consult `?$`.

26. Produce the sequence of integers 0, 1, 8, 27 . Assign to S the result.

27. Show the first and second elements of S .

28. Create a list using S . Assign to SL the list.

29. Create a set from S. Assign to SS the set.

30. Does Maple consider SL and SS identical? Hint: Enter `evalb(SL=SS)`.

4.7 Names

As discussed in Chapter 3, Maple names act as variables that can store expressions. Since a name can represent an expression, a name must also be an expression. This section briefly reviews some aspects of names that were glossed over previously.

4.7.1 Indexing

Having learned about selection, you are ready to learn why a symbol differs from a name. According to `?name`, a name is an expression with the type `symbol` or `indexed`.

What is an indexed expression? In Standard Math, to access an element from a quantity that stores multiple expressions, you use a *subscript*, also called an *index*. For instance, given a list $L = [a, b, c, d, e]$, the notation L_1 refers to element a, L_2 refers to element b, and so forth. To refer to any element inside L, you would say L_i. ***Indexing*** is the process of labeling a particular expression. The label, also called an index or subscript, is usually an integer. In Maple, an index may be any sequence of expressions.

When using Maple Notation, you use the square brackets to indicate an indexed expression, as in ***`expr[index]`***. When ***`expr`*** is unassigned, Maple will report ***`expr`*** with ***`index`*** as a subscript:

Step 91: Indexing

```
> x[1],x[2],x[3];                                        Provide indices for x .
```

$$x_1, x_2, x_3$$

Because x has no assignment, no elements are extracted.

When ***`expr`*** has an assigned value, Maple will perform selection by extracting the element from the expression at the position indicated by ***`index`***. Consult `?indexed`, `?type[indexed]`, and `?op` for more information.

4.7.2 Symbolic Constants

Numbers, like integers, fractions, and floats, are numeric constants. Maple also contains ***symbolic constants***, which are predefined names that store commonly used values. You may discover the entire collection of symbolic constants by entering `constants`:

Step 92: Symbolic Constants

```
> constants;                                This variable stores symbolic constants.
```

$$false, \gamma, \infty, true, Catalan, FAIL, \pi$$

Each item here is a built-in constant.

Consult `?constant` and `?type[constant]` for a full list of constants and more information.

4.7.3 Protected Names

Besides symbolic constants, many names in Maple have predefined values, as discussed in Chapter 3. Maple uses protection to prevent you from assigning a predefined name. So, protected names have their own type, as discussed in `?type[protected]`. You will find initially known names in `?constants` and `?ininames`. For a list of protected function names, investigate `?inifcns` and `?index[function]`.

4.7.4 Greek Names

Maple automatically converts names of Greek characters into their correct font. For a full list, see Appendix A and `?symbolfont`, but you may find most names using the symbol palette by selecting View→Palettes→SymbolPalette.

Step 93: Greek Names

```
> alpha,beta,gamma,delta;                    Enter some Greek letters using Maple notation.
                    α, β, γ, δ               Maple automatically converts each name.
```

Beware that Maple uses many uppercase Greek letters for functions, so their names might be protected.

Practice!

31. What Maple input generates the output $a_b + c_d$?

32. Without entering `constants`, produce these symbolic constants using Maple names: $false, \gamma, \infty, true, Catalan, FAIL, \pi$.

33. Enter the symbol $-\infty$ in Maple Notation.

34. Assign to the name β the value 1.

35. Assign to the name `Beta` the value 1. Explain the output.

4.8 Miscellaneous

Maple expressions include more than numbers and constants. This section only scratches the "surface" (so to speak) of more Maple types. You will discover more about these types throughout later chapters and your studies. See also Appendix C.

4.8.1 Arithmetic

As shown in Sections 3.3.3 and 4.1.4, some expressions have arithmetic operators as the “top” node in their trees. Maple classifies such expression as arithmetic types, using the tokens `‘+‘`, `‘*‘`, and `‘^‘`. For more information, consult `?type[arithop]`.

4.8.2 Strings

Strings provide text labels for commenting. Some functions, like `plot`, use strings for labels and titles. As briefly introduced in Table 3-1, to create a string, surround characters with double quotes (`"`). Strings act as “clumps” of characters that you can manipulate and move around. For instance, you may concatenate (think, “stick together”) two strings with `cat(`***`string1`***`,`***`string2`***`)`:

Step 94: Concatenating Strings

```
> Str1:="ab":                                   Assign to Str1 the string "ab".
> Str2:="cd":                                   Assign to Str2 the string "cd".
> String:=cat(Str1,Str2);        Append "cd" to "ab" and assign result to String.
                    String := "abcd"                 The concatenated string.
```

For more information and examples, consult `?string`, `?type[string]`, and the worksheet `?examples[string]`.

4.8.3 Functions

Functions are one of Maple’s most types. But, don’t worry – I haven’t forgotten! Functions have the form ***`expr(exprseq)`*** and are discussed in the next chapter. In fact, the remaining chapters in this book are devoted to exploring Maple’s wide variety of functions.

4.8.4 And More…

Consult Appendix C and `?type` for many other useful types. Many of these types will assist you in developing programs in Maple.

4.9 Application: Looking for Something Better

This section is deliberately left blank so I can find something better to use. Such is life.

Professional Success: Units

Never let software foster laziness! Many programs allow numerical input without unit labels. However, engineering and science quantities measure *units*, which are properties associated with a physical system. Always write your units, especially to help those who check your work – your instructors/managers will thank you. Consider these tips when using Maple:

- ❐ Pick unit names:

Predefine certain names for unit labels by protecting your units with **`protect(name)`**:

```
> restart:                                              Refresh worksheet.
> protect(m): protect(ft):              Prevent unit names from being assigned.
```

```
> ft := 1:                                           Attempt to use ft anyway.
Error, attempting to assign to `ft` which is protected     Maple complains.
```

❒ Label your input:

For example, by interpreting the quantity "one foot" as $one \times foot$, you can multiply expressions by unit labels:

```
> Side:=10.0*ft: Area := Side^2;            Evaluate Area in terms of feet (ft).
```

$$Area := 100\, ft^2$$

Maple now shows units!

❒ Convert your units:

Let Maple ease your work. The function, **`convert`**, will convert U.S. units into metric:

```
> Digits:=3:                  Set a reasonable number with which to show results.
> Area2 :=convert(Area,metric);            Convert your area into square meters.
```

$$Area2 := 9.30 m^2$$

Maple reports your results in metric units.

Consult **`?convert[metric]`**, **`?convert[degrees]`**, and **`?convert[decimal]`** for other great conversion tricks.

Summary

- Maple classifies expressions according to type.
- For a full list of Maple types, see **`?type`**.
- A surface type is the top node in an expression tree.
- Maple attempts to preserve exactness in all numerical operations.
- Mixing floating-point values tends to breed further floats.
- For exactness, Maple does not produce floats for irrational numbers.
- You may set rounding with **`Rounding`** and **`Digits`**.
- Maple represents imaginary components of complex numbers with **`I`**.
- You may represent infinite quantities with the symbolic constant **`infinity`**.
- To represent the truth- and falsehood of expressions, use Boolean values and relations.
- Sequences, lists, and sets store multiple expressions.
- Whereas sequences are not formally expressions, lists and sets are.
- Use the **`[]`** operator to index a name or extract an element from a sequence, list, and set.

Key Terms

Boolean
complex number
environment variable
equation
extended numerics
floating-point number
fraction
indexing
infinity
irrational number
list
logical
nested type
Pi
range
rational number
relation
sequence
set
structured type
surface type
symbolic constant

Problems

1. What is a Maple data type?
2. Explain the difference between surface and structured types.
3. Identify the surface and structured type of the expression $a + bc$.
4. Explain the difference between real and complex numbers.
5. Explain the difference between rational and irrational numbers.
6. Explain the differences and similarities between sequences, lists, and sets.
7. Evaluate the following expressions using Maple. You must demonstrate both exact and floating-point (decimal) results for all output. Hints: Be careful with operator precedence and associativity. To find decimal results, either enter floats in the input or use **`evalf(`*`expr`*`)`**.

7a. $1 + \frac{1}{2} + \frac{1}{3}$

7b. $1 + 0.51$

7c. $1 + \sqrt{2}$

7f. i^2

7g. $\frac{2i + 3}{4i}$

7h. $\sqrt{-10}$

7d.	$\frac{1}{3}+\sqrt{0.1}$	7i.	e^2+3
7e.	2π	7j.	$e^{\sin\left(\frac{\pi}{3}\right)}$

8. Is the number zero (0) an integer? Prove your answer using Maple.
9. Maple's **Digits** variable can often produce surprising results. Consider the following session:

```
> restart:
> 2*1.05;
                              2.10
```

```
> Digits:=2:
> 2*1.05;
                              2.2
```

Demonstrate the same Maple session. Why do you think that Maple produced 2.2 after you set **Digits** to **2**?

10. Determine the structured type for the expression $\sqrt{a+b}$. Hint: You need the **&** operator. Demonstrate with type function answering "true" to your input.
11. Derive a formula for complex division using Maple. Hint: use *complex conjugate* of $a+bi$, which is $a-bi$.
12. Why does entering **evalb(exp(1)=e)** produce $false$?
13. The volume of a sphere is $V = \frac{4}{3}\pi r^3$ for radius r. Do the following tasks:

13a. Given $r = 3$ cm, evaluate V. Be sure to specify units in your assignments!
13b. Find the floating-point value of your answer in Problem13b.
13c. Convert your result to inches. 1 in = 2.54 cm. Hint: Maple will not permit **in** as name. Try **?in** to see. Use **inch**, instead.

14. Given the data in Table 4-6, answer the following questions:

Table 4-6 Data Points

Index	1	2	3	4	5
x	2	–4	3	1	0
y	–1	5	2	6	1

14a. Assign to X the sequence of x values.
14b. Assign to Y the sequence of y values.

14c. Generate a list of data pairs. Each pair of data must have a list structure in the form $[x_i, y_i]$ as well. Assign to the variable DP your result.

14d. Extract the y value from the DP's third value, DP_3. Hint: Use the selection operator `[]` twice.

15. One of DeMorgan's Laws states that $(A \cup B)^c = A^c \cap B^c$. Demonstrate this relationship with Maple. Use two sets $A = \{a, b, c\}$ and $B = \{a, d\}$ along with a *Universe of Discourse* $U = \{a, b, c, d, e\}$. Hint: A complement of a set S, called S^c, is the difference between U and S. See also `?minus`.

16. Electrical engineers often distinguish complex numbers with the letter j to prevent confusion with the symbol for current, i. Change Maple's default imaginary number from `I` to `j`. Next, demonstrate your new name by evaluating $\sqrt{-1}$ with `evalc`. Hint: See `?I`.

17. Assume part of a circuit you are analyzing contains a *resistor* R (Ω or Ohms) and *capacitor* C (F or farads) connected together in a series, as shown in 4-2:

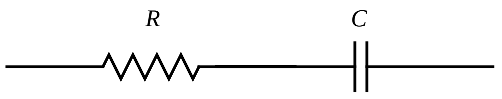


Figure 4-2 Resistor and Capacitor in Series

(Capacitors store electrical charge. Section reviews resistors.) *Impedance* measures the effect elements, such as capacitors and resistors have on electrical current. You can calculate this configuration's impedance Z (Ohms) with the formula

$$Z = R - \frac{j}{\omega C}$$

where ω (rad/s) represents the angular frequency of the voltage source. The symbol j represents an imaginary number as discussed in Problem 17. Given $\omega = 400$ rad/s, $R = 5\,\Omega$, and $C = 1 \times 10^{-9}$ F:

17a. Evaluate Z using Maple's default imaginary number, variable I.

17b. Evaluate Z using j, as described in 17.

18. [note to self: restore interval analysis problem]

19. [note to self: more circuit analysis -- old chapter 4 application?]

20. [note to self: program for checking all ?keywords if they're protected. need to write program using type(thing,protected). must backquote each. find out that Maple thinks operators are protected and all other keywords are reserved words.]