

# 3. Rules and Patterns

---



## 3.1 Introduction

## 3.2 Rules

- Objectives
- Using **Rule** and **ReplaceAll**
- Using **RuleDelayed**
- Exercises

## 3.3 Patterns

- Objectives
- What is a pattern?
- Named patterns
- Exercises

## 3.4 Compound Expressions

- Objectives
- Using compound expressions
- Exercises

## 3.5 Problems

- Problem 1: Applying rules to expressions
- Problem 2: Applying rules to formulas
- Problem 3: Applying rules to data
- Problem 4: Applying rules and conditions to data

## 3.1 Introduction

---

The aim of this module is to understand, and gain a facility in using, three concepts basic to writing high level programs. These are *rules*, *patterns* and *compound expressions*.

In the module after this, we will use these concepts to discuss *functions* - the beginning of programming.

## 3.2 Rules

---

### Objectives

---

- To understand the concept of rules.
- To understand the concept of replacement.
- To understand the difference between replacement using rules and replacement using delayed rules
- To be able to use rules, delayed rules and the operation of replacement correctly.

### Using Rule and ReplaceAll

---

If we want to evaluate an expression without permanently changing the expression, we use *rules* and a *replacement operation* which applies the rules.

For example, suppose we had the expression  $(x + 1)^5$ , and we wanted to find its value when  $x$  is 2. If we enter  $x = 2$  and then enter  $(x + 1)^5$  we will get the value **243** that we expected.

```
x = 2
```

```
2
```

```
(x + 1)5
```

```
243
```

However, we have now "lost" our expression, since every time we want to use it (with  $x$  not necessarily equal to 2 any more), it just turns up the number **243**. For example

```
Expand[(x + 1)5]
```

```
243
```

We can regain it by unsetting  $x$  ( $x = .$ ), but this is pretty tedious, especially if there are a lot of different things we want to do with the expression. (But here we need to do it anyway so that we can continue).

```
x = .
```

Mathematica overcomes this problem by using *rules* and *replacement* to substitute values in expressions. Here is an example:

```
(x + 1)5 /. x → 2
```

```
243
```

Neither the value of  $x$  nor  $(x + 1)^5$  has been changed in this operation, which we can see by entering them both again.

$$\{\mathbf{x}, (\mathbf{x} + 1)^5\}$$

$$\{\mathbf{x}, (1 + \mathbf{x})^5\}$$

(We put these two required outputs in a list to save space on the page - something we will often do from now on).

The operation  $\rightarrow$  may be read "goes to".  $\mathbf{x} \rightarrow 2$  is then read " $\mathbf{x}$  goes to  $2$ ". This is the *rule*.

The operation  $/.$  may be read "given that".  $(\mathbf{x} + 1)^5 / . \mathbf{x} \rightarrow 2$  is then read " $(\mathbf{x} + 1)^5$  given that  $\mathbf{x}$  goes to  $2$ ". This is the *replacement operation*.

The internal form of an expression like  $\mathbf{A} / . \mathbf{x} \rightarrow \mathbf{a}$  is

```
HoldForm[FullForm[A /. x -> a]]
```

```
ReplaceAll[A, Rule[x, a]]
```

$\mathbf{A} / . \mathbf{x} \rightarrow \mathbf{a}$  may be read as " $\mathbf{A}$  given that  $\mathbf{x}$  goes to  $\mathbf{a}$ "

The symbol  $\rightarrow$  may also be entered as the two characters  $\rightarrow$

#### ◆ Example

You can apply several rules at once by using a list of rules.

Here we enter a formula for the friction torque on an unloaded journal bearing.

$$\mathbf{T} = \frac{4 \pi^2 \mu \mathbf{N} \mathbf{L} \mathbf{R}^3}{\mathbf{c}}$$

$$\frac{4 \mathbf{L} \mathbf{N} \pi^2 \mathbf{R}^3 \mu}{\mathbf{c}}$$

The formula for the friction torque (in appropriate units) on bearings of radius  $\mathbf{R}$  of  $20$  mm, length  $\mathbf{L}$  of  $30$  mm and clearance  $\mathbf{c}$  of  $0.1$  mm is then derived by applying the substitutions

$$\mathbf{T} / . \{\mathbf{R} \rightarrow 20, \mathbf{L} \rightarrow 30, \mathbf{c} \rightarrow 0.1\}$$

$$9.47482 \times 10^7 \mathbf{N} \mu$$

Here the speed  $\mathbf{N}$  and the viscosity  $\mu$  are still unevaluated.

#### ◆ Example

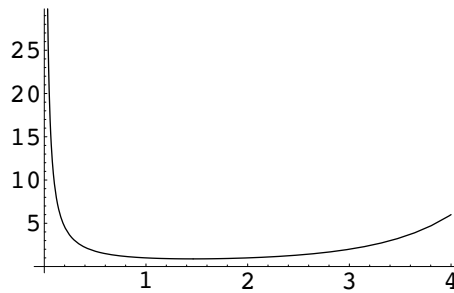
You can also replace heads of expressions. We could set up an expression

`gPlot[Gamma[x], {x, 0, 4}]` which Mathematica does not understand (because the symbol `gPlot` has not been defined), and only later make it evaluate by replacing the symbol `gPlot` with the symbol `Plot` (which Mathematica understands).

```
p = gPlot[Gamma[x], {x, 0, 4}]
```

```
gPlot[Gamma[x], {x, 0, 4}]
```

**p /. gPlot → Plot**



- Graphics -

### ■ Advanced concept

Remember that since the left side of `/.` is the first argument to **ReplaceAll**, it is evaluated *before* the replacement is made. (Remember the evaluation sequence).

```
(x + 2 x) /. (2 x → y)
3 x
```

Hence `x+2x` is evaluated to `3x` first, and there is no `2x` left to be replaced by `y`.

(We needed to put parentheses here because *Mathematica* would have had trouble parsing the sequence `... x/.2 ...`). *Parsing* is the operation of interpreting a string of symbols.

### Using RuleDelayed

---

Just as with **Set** and **SetDelayed**, you may wish to have the right hand side of the rule evaluated *only at the time the rule is applied*.

The symbol `:=` (or `:>`) can be used instead of `→` (or `->`) to ensure that the evaluation of the right hand side of the rule is delayed until the rule is applied using `/.`

For example, returning to our random number example, applying **Rule** (`→`) evaluates the right hand side of the rule once and then uses that value in each of the three replacement operations.

```
{x, x, x} /. x → Random[]
{0.356103, 0.356103, 0.356103}
```

On the other hand, applying **RuleDelayed** (`:=`) evaluates the right hand side of the rule *each time* it makes a replacement.

```
{x, x, x} /. x := Random[]
{0.3731, 0.681004, 0.328797}
```

## Exercises

---

### ◆ Exercise: Using rules in data

Take the expression **Engine**[{**Bore**, **Stroke**},{**MaxPower**,**Revs**}] and make the substitutions **Engine** → **Engine2001**, **Bore** → **83 mm**, **Stroke** → **75 mm**, **MaxPower** → **123 Watt**, **Revs** → **5950 rpm**.

### ◆ Exercise: Using rules in formulas

Take the relation between the torque and the force on a clutch

$$\mathbf{T} = \frac{2}{3} \mu \mathbf{F} \left( \frac{\mathbf{r}_0^3 - \mathbf{r}_i^3}{\mathbf{r}_0^2 - \mathbf{r}_i^2} \right) \mathbf{n}$$

and replace symbol for the radius **r** by the symbol **R**.

### ◆ Exercise: Using rules and delayed rules

Take the expression **x<sup>n</sup> + y<sup>n</sup> + z<sup>n</sup>** and replace **n** by a random number: first by using a rule, and second by using a delayed rule. Understand the difference between the two resulting expressions.

◆

## 3.3 Patterns

---

### Objectives

---

- To understand the concept of patterns.
- To understand the concept of replacement using patterns.
- To be able to use patterns correctly.

### What is a pattern?

---

Suppose we have an expression **1 + x + y + x<sup>2</sup> + y<sup>2</sup>**, and we want to replace each of the squared terms by the symbol **k**.

We can do this by constructing a rule which says something like this: "Anything with the *form* **\_<sup>2</sup>** goes to **k**". The underscore **\_** (called **Blank[]** in internal form) represents *any* symbol, in this case either **x** or **y**.

The rule would then be

$$\_{}^2 \rightarrow k$$

Applying this rule to the expression should then give  $1+x+y+2k$ .

$$1 + x + y + x^2 + y^2 / . \_{}^2 \rightarrow k$$

$$1 + 2k + x + y$$

A pattern is a way of writing *any* expression of a given *form*.

#### ◆ Example

Suppose we have a list of data and we want to put any data pairs in the list to zero. We can do this with the rule  $\{ \_, \_ \} \rightarrow 0$ .

$$\{8.523, 8.042, \{-4.392, 1.121\}, \{-1.106, 0.109\}, -3.199, 4.026, -6.986, 0.876, \{3.496, 2.271\}\} / . \{ \_, \_ \} \rightarrow 0$$

$$\{8.523, 8.042, 0, 0, -3.199, 4.026, -6.986, 0.876, 0\}$$

### Named patterns

---

The blanks discussed in the previous section are useful whenever the rule does not involve the symbols (that the blanks refer to) on the right hand side of the rule. However, in most cases it is useful to give the blanks a *name* so that we can refer to them. For example, suppose we wanted to increase the squares in the previous expression to cubes. We give the blank a name **z**, and write the rule as  $\mathbf{z\_}^2 \rightarrow \mathbf{z}^3$ . Of course,  $\mathbf{z\_}$  still stands for either **x** or **y**.

$$1 + x + y + x^2 + y^2 / . \mathbf{z\_}^2 \rightarrow \mathbf{z}^3$$

$$1 + x + x^3 + y + y^3$$

#### ◆ Example

Suppose we have a list of data pairs (say, x and y coordinates) and we want to interchange the order of the elements in each pair. We can do this with the rule  $\{\mathbf{x\_}, \mathbf{y\_}\} \rightarrow \{\mathbf{y}, \mathbf{x}\}$ .

$$\{\{1, 2\}, \{5, -3\}, \{0, -2\}, \{0, 8\}, \{6, 2\}\} / . \{\mathbf{x\_}, \mathbf{y\_}\} \rightarrow \{\mathbf{y}, \mathbf{x}\}$$

$$\{\{2, 1\}, \{-3, 5\}, \{-2, 0\}, \{8, 0\}, \{2, 6\}\}$$

### Exercises

---

#### ◆ Exercise: Using rules and patterns in formulas

Use a rule to reduce any powers in the expression  $\mathbf{z}^3 + \mathbf{Log}[2 - \mathbf{x}^5] \mathbf{Cos}[\mathbf{y}^n + 1]$  by 1.

◆ **Exercise: Using rules and patterns in data**

Use a rule to interchange the order of the *points* in the point pair  $\{\{\mathbf{x}_1, \mathbf{y}_1\}, \{\mathbf{x}_2, \mathbf{y}_2\}\}$ .

◆ **Exercise: Using rules and patterns to modify symbols**

Use a rule to change the subscripts to powers in  $\{\{\mathbf{x}_1, \mathbf{y}_1\}, \{\mathbf{x}_2, \mathbf{y}_2\}\}$ .

◆

## 3.4 Compound Expressions

---

### Objectives

---

- To understand the concept of compound expressions.
- To understand how to suppress output.

### Using compound expressions

---

If you want to enter a sequence of expressions, but are only interested in the output from the last one, you can use a *compound expression*. For example, suppose you had  $\mathbf{x} = \mathbf{Log}[\mathbf{t}] + \mathbf{t}^2$  and  $\mathbf{y} = \sqrt{\mathbf{x}} \mathbf{Sin}[\mathbf{x}]$  and you wanted to compute  $\mathbf{x} \mathbf{y}$ . You could either enter them in separate cells, and get sequential outputs, or you could enter them as a compound expression within one cell separating the expressions with semicolons (;).

$$\mathbf{x} = \mathbf{Log}[\mathbf{t}] + \mathbf{t}^2; \mathbf{y} = \sqrt{\mathbf{x}} \mathbf{Sin}[\mathbf{x}]; \mathbf{x} \mathbf{y}$$

$$(\mathbf{t}^2 + \mathbf{Log}[\mathbf{t}])^{3/2} \mathbf{Sin}[\mathbf{t}^2 + \mathbf{Log}[\mathbf{t}]]$$

Compound expressions are important in building up a sequence of computations. However they have the disadvantage that you do not see the results of the intermediate computations. Checking the results of each of your computations as you progress is always a good idea.

We are introducing compound expressions here so that you can read what other people have written, but we do not recommend using them until we come to Modules later on.

However, there is one variant of the compound expression that you will see often, and which can be useful for suppressing the display of your output: a semicolon (;) after an expression.

## Exercises

---

◆ **Exercise: Compounding expressions**

Check that the sequential entering of  $\mathbf{x = \text{Log}[t] + t^2}$ ,  $\mathbf{y = \sqrt{x} \text{Sin}[x]}$ , and  $\mathbf{x y}$  gives the same result as entering  $\mathbf{x = \text{Log}[t] + t^2}$ ;  $\mathbf{y = \sqrt{x} \text{Sin}[x]}$ ;  $\mathbf{x y}$

◆ **Exercise: Suppressing display**

Type in an expression and follow it with a semicolon. Enter this. Show that Mathematica has in fact understood the expression, but just did not display any output.

◆ **Exercise: Understanding the internal form of a compound expression**

Find the **FullForm** of a compound expression.

◆

## 3.5 Problems

---

### Problem 1: Applying rules to expressions

---

Take any 2 dimensional graph and use a rule on its points to reflect it about the line  $y = x$ .

Get the result to plot by using the **Show** command on the result of your transformation.

◆

### Problem 2: Applying rules to formulas

---

Take the relation between the torque and the force on a clutch

$$\mathbf{T = \frac{2}{3} \mu F \left( \frac{r_0^3 - r_i^3}{r_0^2 - r_i^2} \right) \mathbf{n}}$$

and use one rule to convert the formula to be in terms of diameters  $\mathbf{D_0}$  and  $\mathbf{D_i}$  instead of radii  $\mathbf{r_0}$  and  $\mathbf{r_i}$ . Simplify the result using **Simplify**.

◆

### Problem 3: Applying rules to data

---

You have some sound data consisting of a list of pairs, the first element in the pair being the frequency in Hz, and the second element being the corresponding sound pressure in Pa. Write a rule to convert the pressure values in the list to dB according to the formula

$$\text{dB} = 10 \text{Log} \left[ \left( \frac{P}{2 \times 10^{-5}} \right)^2 \right].$$

Test out your rule on some data generated from Problem 5 in module 2, but have the frequency range from 100 to 20000.

Plot the resulting converted **data** using **ListPlot[data, PlotJoined→True]**.



### Problem 4: Applying rules and conditions to data

---

When developing a new product, an engineer may be faced with the problem of excessive noise being generated by the product when in use. One method used to analyze the situation is to determine all the frequencies being generated by the product and then to play the sound with certain frequencies removed. Once the frequencies responsible for the excessive noise are identified, the product can be investigated to determine the origin of the frequency, and a redesign proposed.

Take the converted data from Problem 3 above and write a rule to reduce all dB levels between frequencies 5000 and 10000 to zero.

*Hint:* Try using the **If** function in the form **If[1000 ≤ f ≤ 10000, 0, dB]**. (You may need to do a little research to find out what an **If** function does).

Make a **ListPlot** of the modified data. Verify that your data modification has worked as expected.

