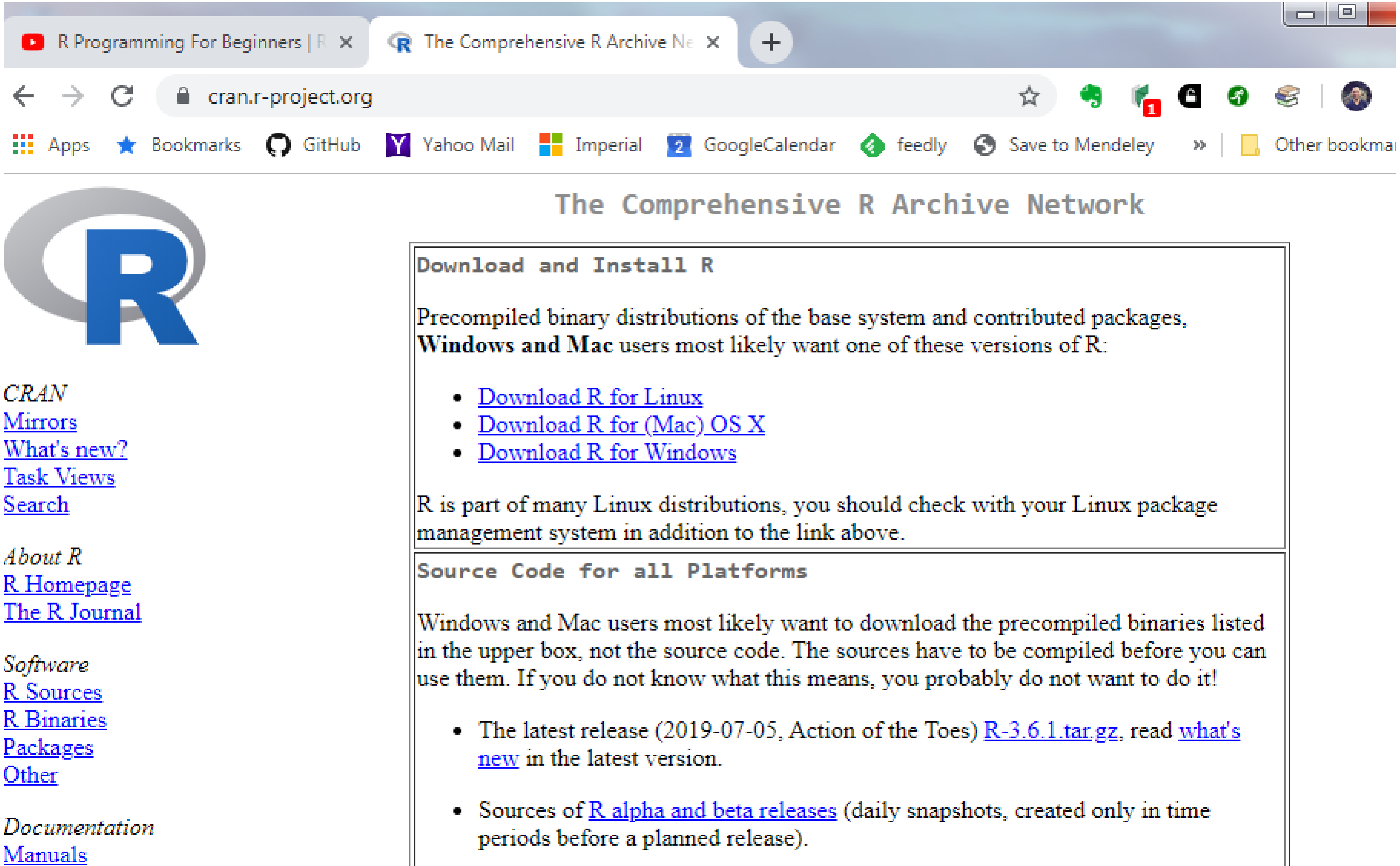
Introduction to using R

Imperial College London

Contents

- Variables
- Data Types
- Strings
- Operators
- Conditional statements
- Loops

Download R



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

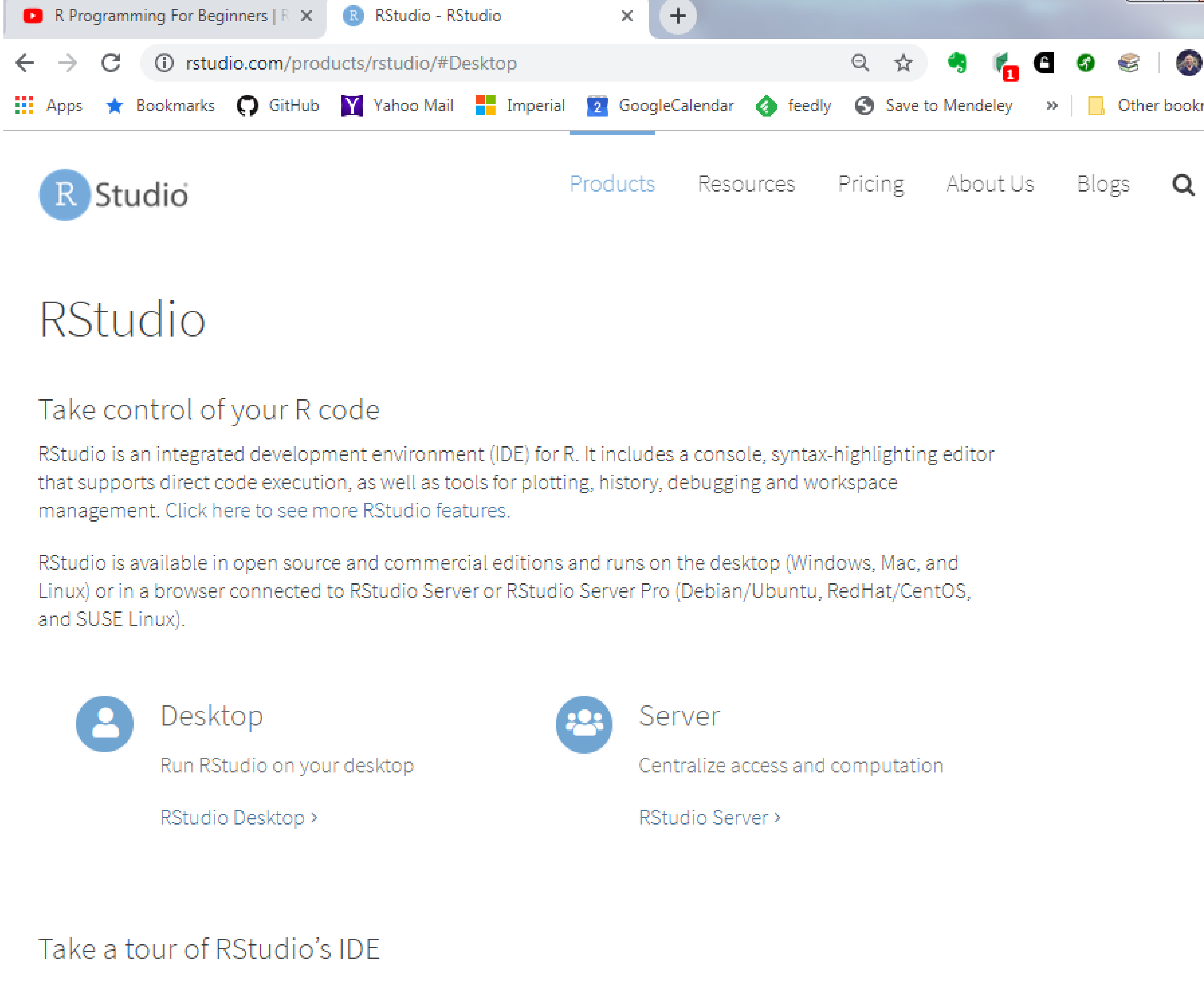
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2019-07-05, Action of the Toes) R-3.6.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).

Download RStudio



RStudio

Take control of your R code

RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management. Click here to see more RStudio features.

RStudio is available in open source and commercial editions and runs on the desktop (Windows, Mac, and Linux) or in a browser connected to RStudio Server or RStudio Server Pro (Debian/Ubuntu, RedHat/CentOS, and SUSE Linux).

Desktop

Run RStudio on your desktop

RStudio Desktop ›

Server

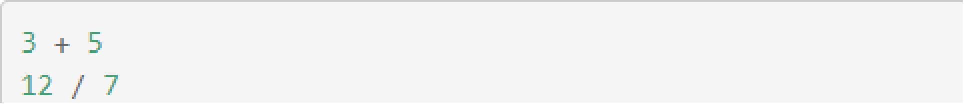
Centralize access and computation

RStudio Server ›

Take a tour of RStudio's IDE

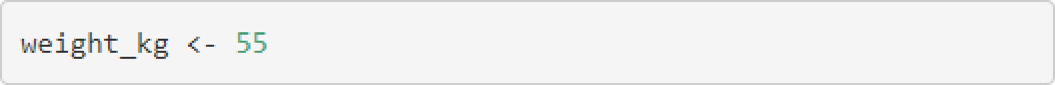
Diving in!

- We can get output simply by typing maths in to the console

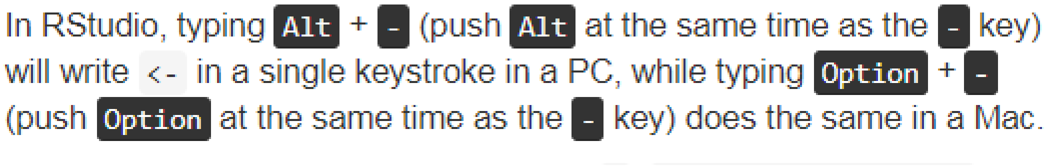


```
3 + 5
12 / 7
```

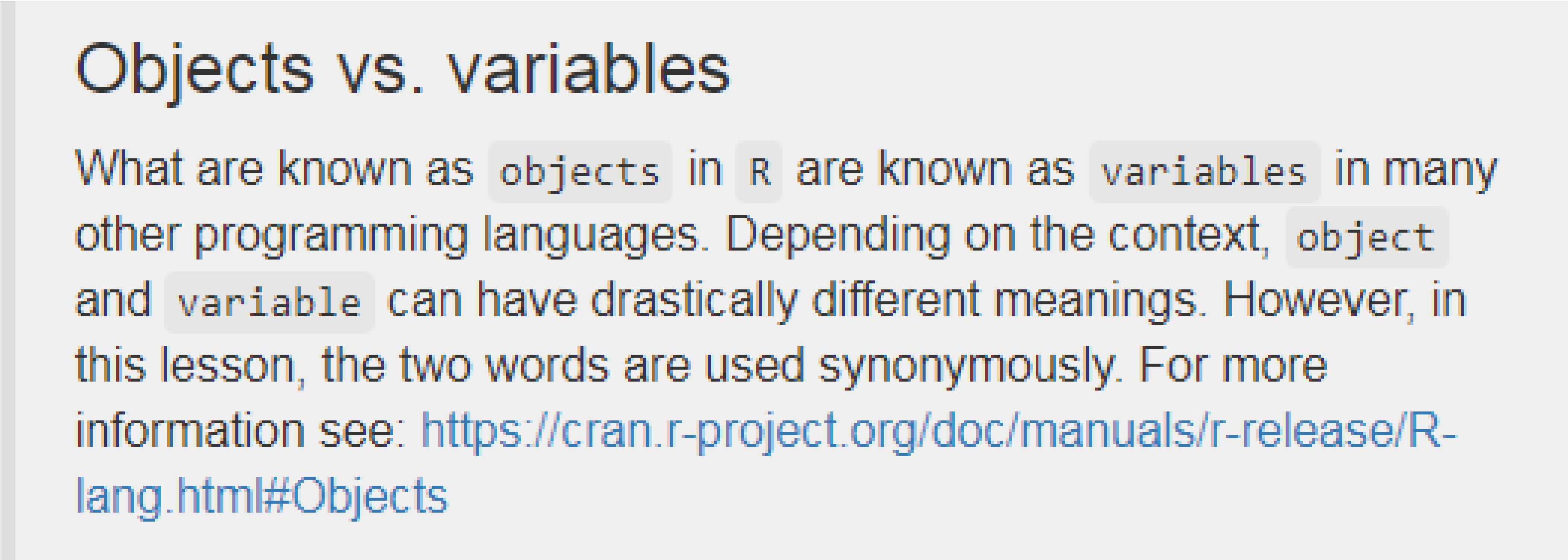
- More interesting is assigning values to variables or objects
- Use the <- operator (more later)



```
weight_kg <- 55
```



In RStudio, typing `Alt` + `-` (push `Alt` at the same time as the `-` key) will write `<-` in a single keystroke in a PC, while typing `Option` + `-` (push `Option` at the same time as the `-` key) does the same in a Mac.



Objects vs. variables

What are known as `objects` in `R` are known as `variables` in many other programming languages. Depending on the context, `object` and `variable` can have drastically different meanings. However, in this lesson, the two words are used synonymously. For more information see: https://cran.r-project.org/doc/manuals/r-release/R-lang.html#Objects

```
weight_kg <- 55    # doesn't print anything
(weight_kg <- 55)  # but putting parenthesis around the call pr
weight_kg          # and so does typing the name of the object
```

Now that R has `weight_kg` in memory, we can do arithmetic with it. For instance, we may want to convert this weight into pounds (weight in pounds is 2.2 times the weight in kg):

```
2.2 * weight_kg
```

We can also change an object's value by assigning it a new one:

```
weight_kg <- 57.5
2.2 * weight_kg
```

This means that assigning a value to one object does not change the values of other objects For example, let's store the animal's weight in pounds in a new object, `weight_lb`:

```
weight_lb <- 2.2 * weight_kg
```

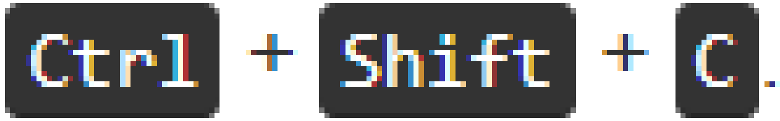
and then change `weight_kg` to 100.

```
weight_kg <- 100
```

Comments

- The comment character in R is #, anything to the right of a # in a script will be ignored by R. It is useful to leave notes and explanations in your scripts. RStudio makes it easy to comment or uncomment a paragraph:

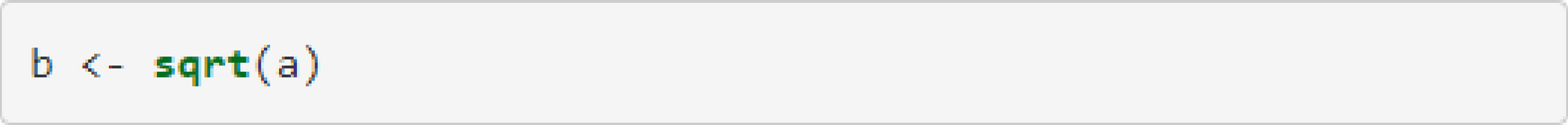
- Press to comment/uncomment a line



Ctrl + Shift + C.

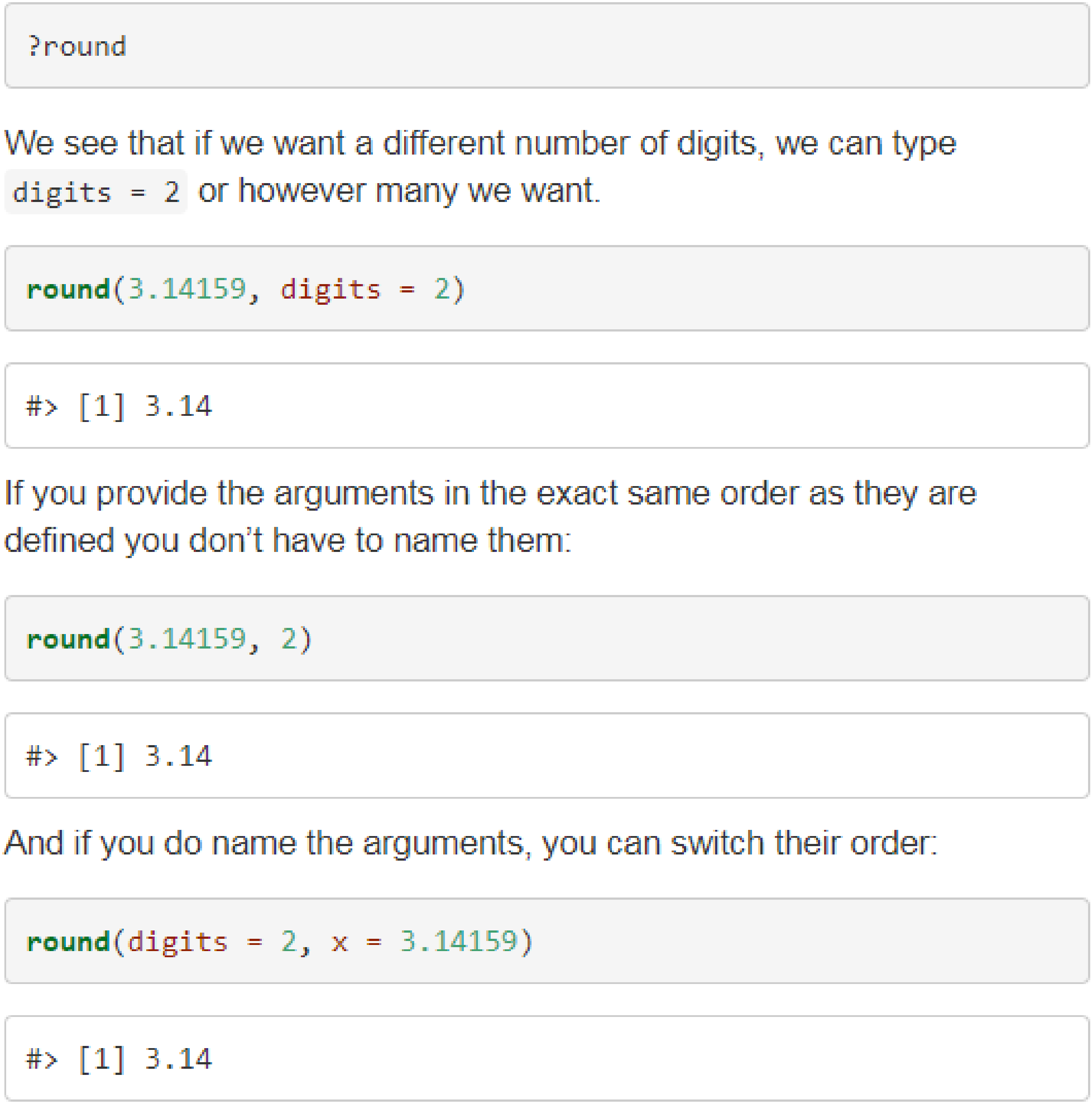
Functions and their arguments

- Automate more complicated sets of commands, as a black-box
- Available by importing *packages*
- A functions has *arguments* and can return a *value*
- *Executing* or *running* a function is called *calling* the function



```
b <- sqrt(a)
```

Multiple arguments



```
?round
```

We see that if we want a different number of digits, we can type `digits = 2` or however many we want.

```
round(3.14159, digits = 2)
```

```
#> [1] 3.14
```

If you provide the arguments in the exact same order as they are defined you don't have to name them:

```
round(3.14159, 2)
```

```
#> [1] 3.14
```

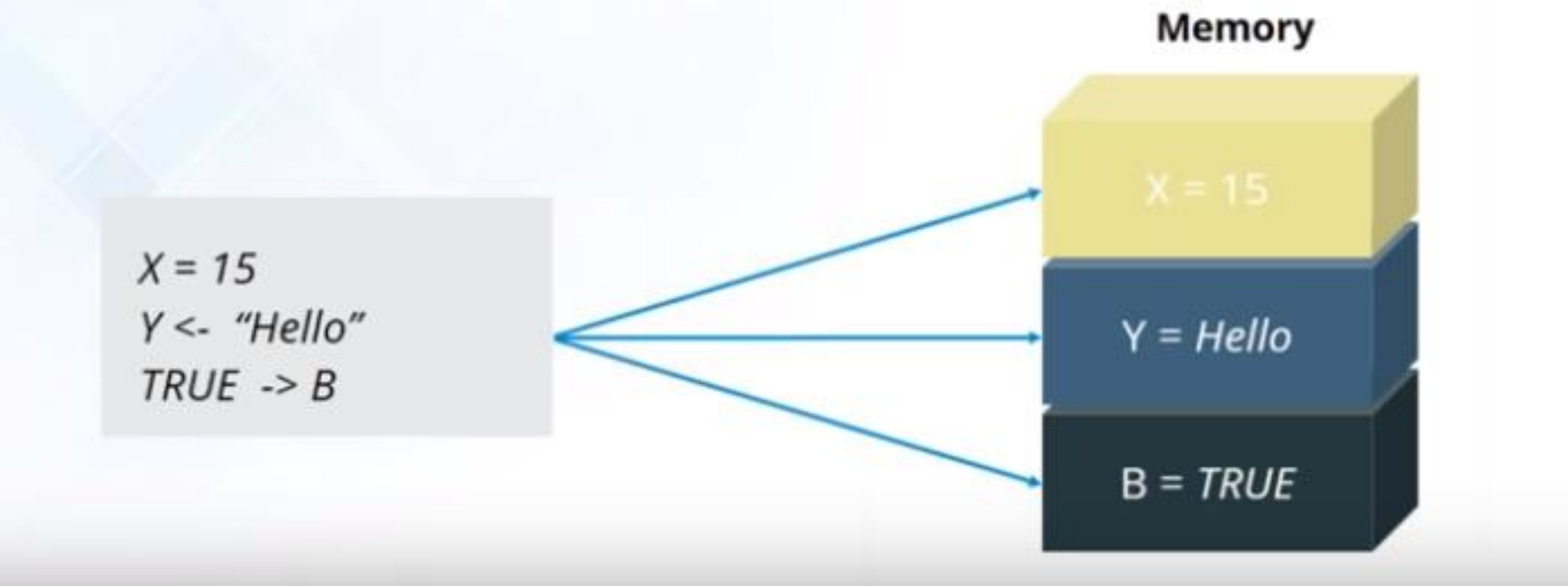
And if you do name the arguments, you can switch their order:

```
round(digits = 2, x = 3.14159)
```

```
#> [1] 3.14
```

Variables

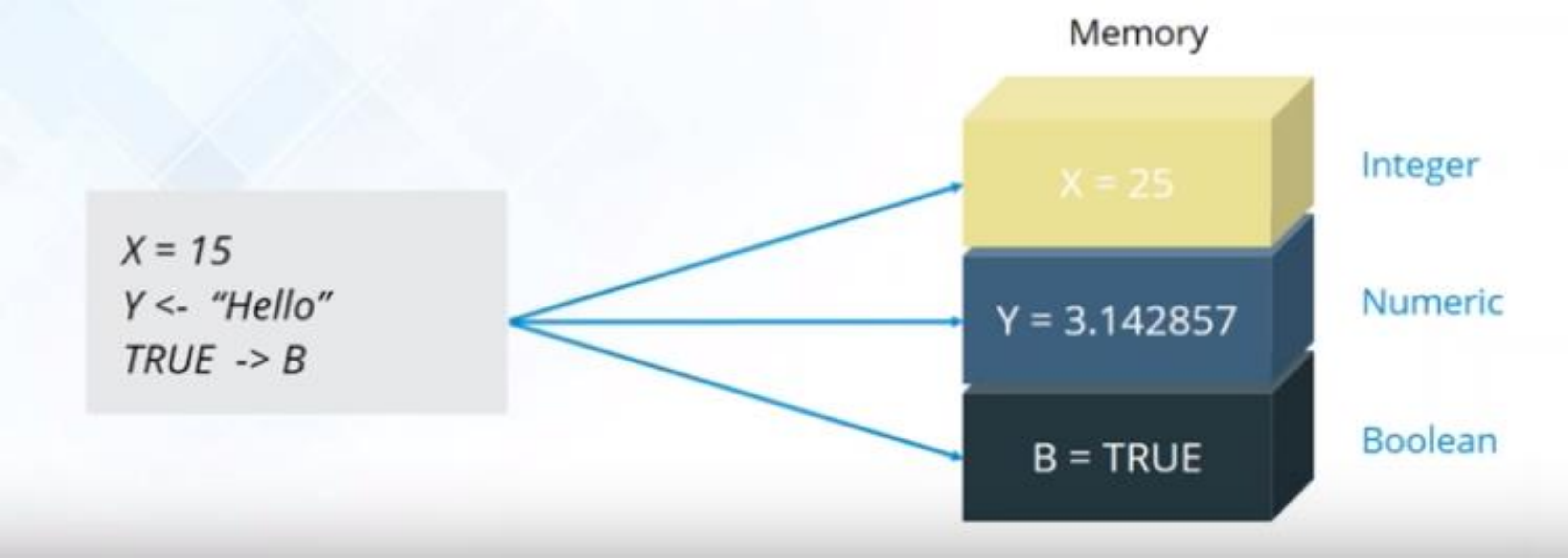
- Variable are reserved memory locations to store values.



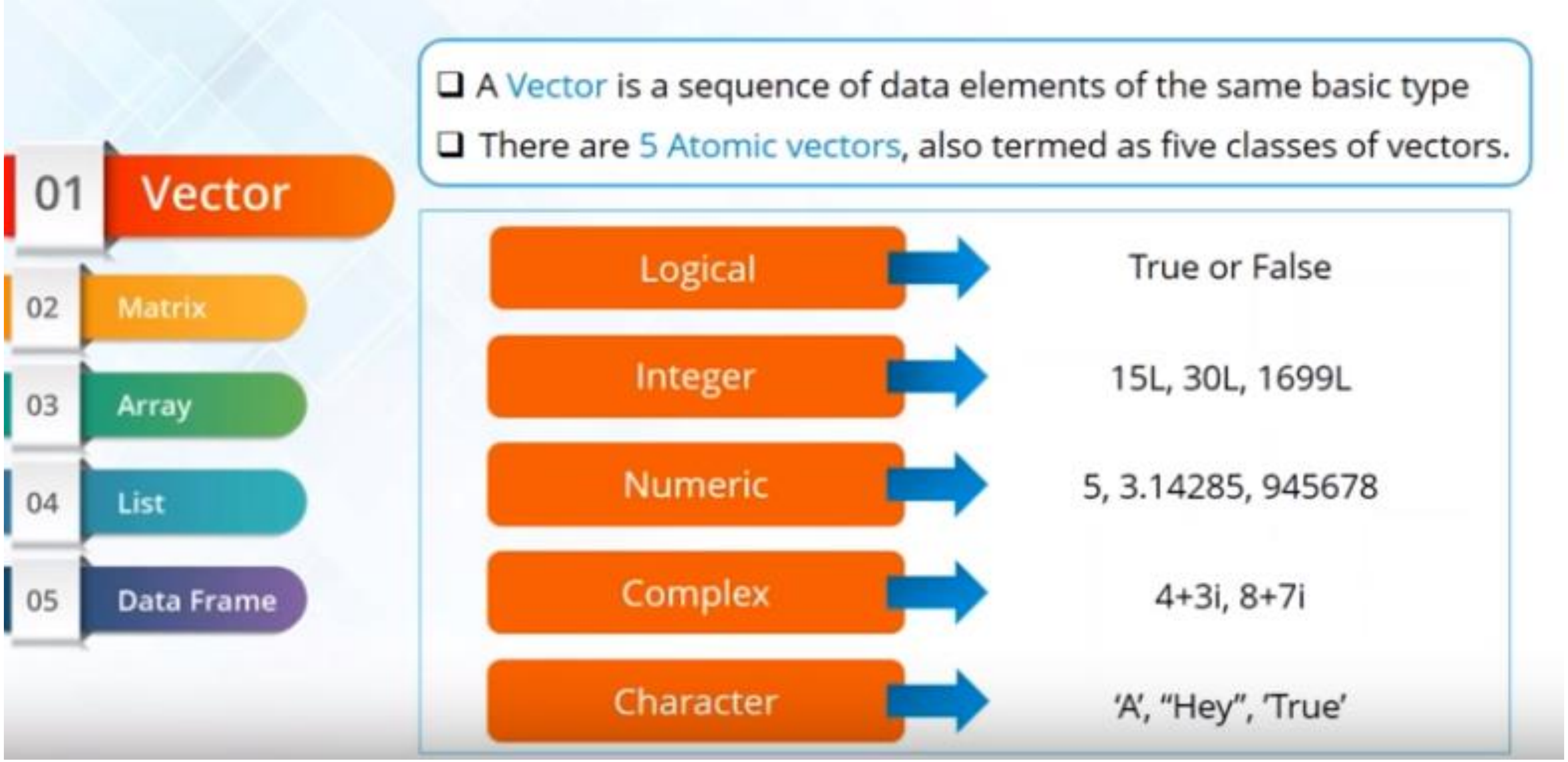
```
X = 15
Y <-  "Hello"
TRUE  -> B
```

Data types

- A classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error



Vector



01 Vector

02 Matrix

03 Array

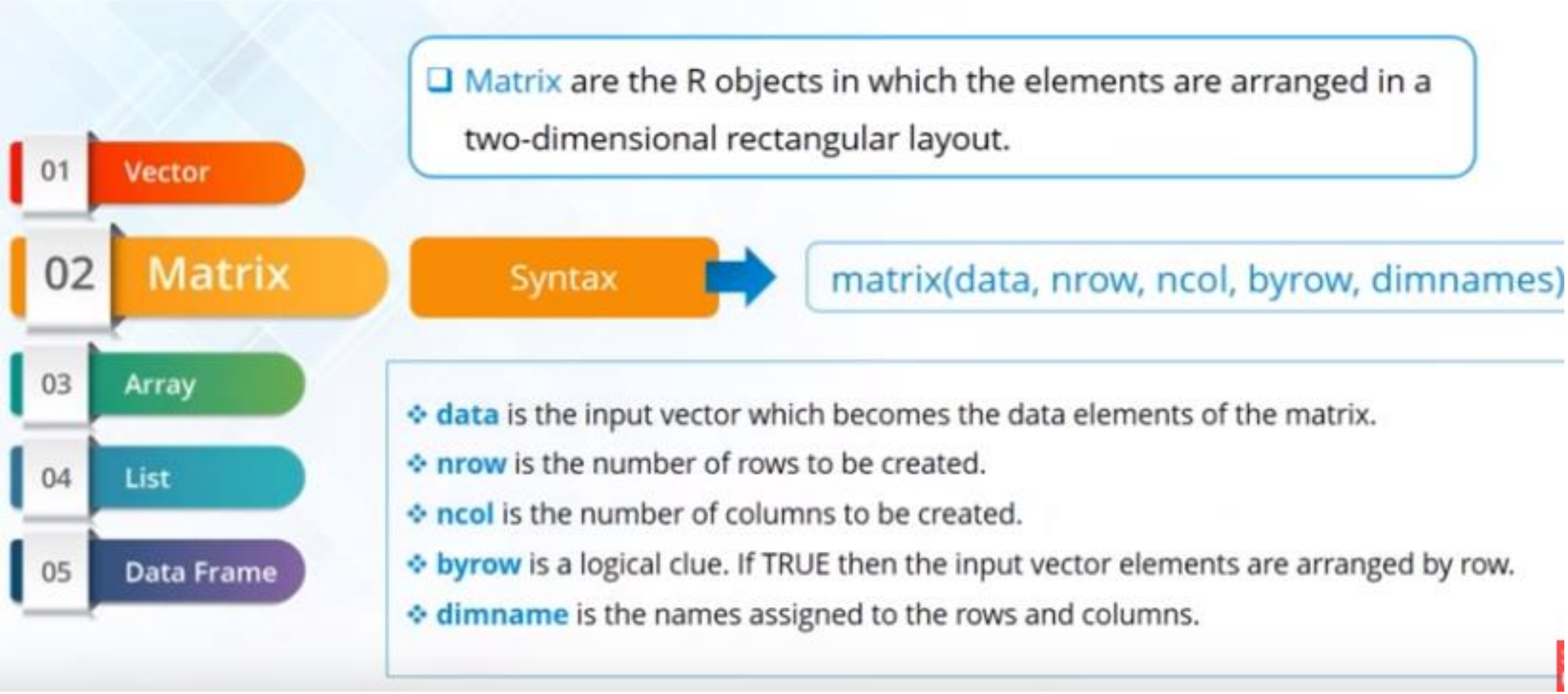
04 List

05 Data Frame

- □ A Vector is a sequence of data elements of the same basic type
- □ There are 5 Atomic vectors, also termed as five classes of vectors.

Logical	True or False
Integer	15L, 30L, 1699L
Numeric	5, 3.14285, 945678
Complex	4+3i, 8+7i
Character	‘A’, “Hey”, ‘True’

Matrix

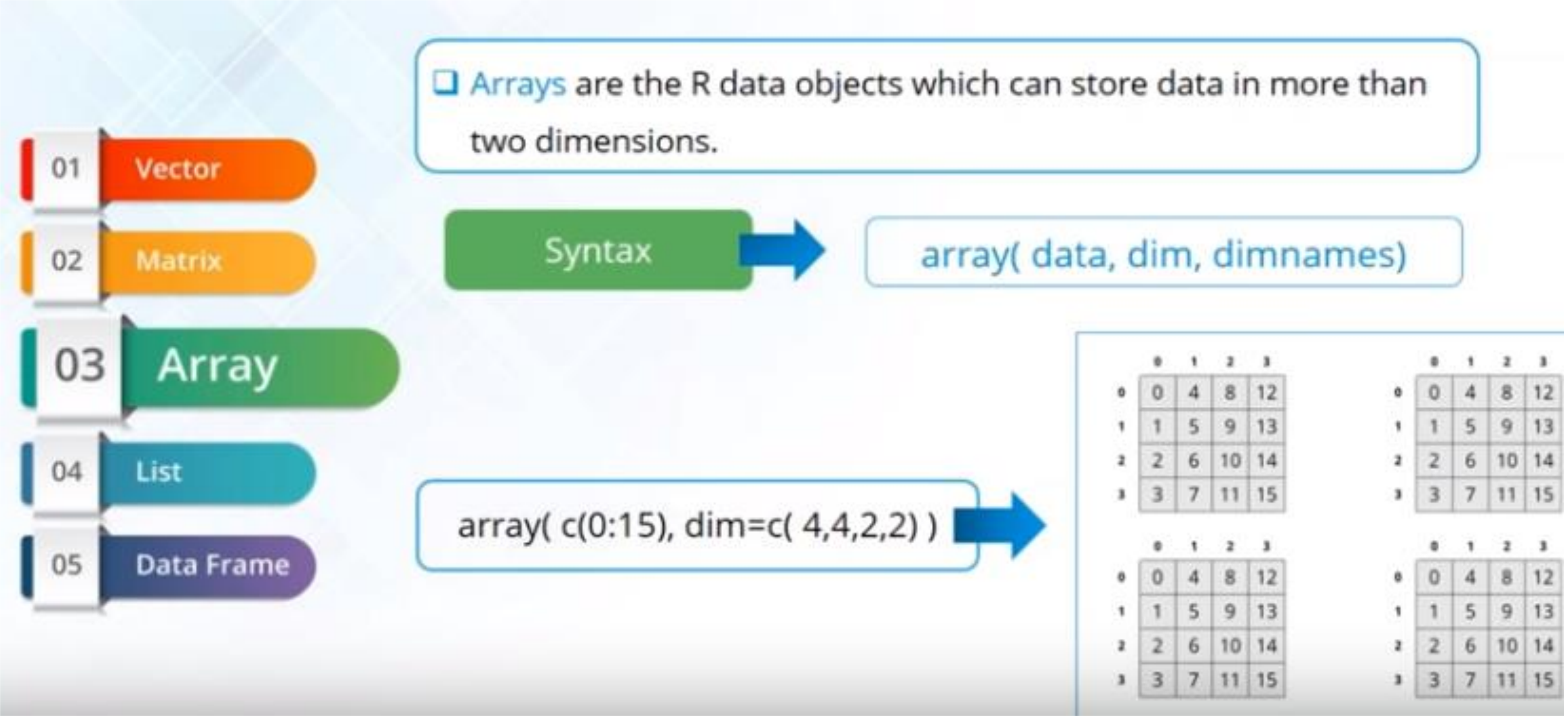


❑ Matrix are the R objects in which the elements are arranged in a two-dimensional rectangular layout.

Syntax → matrix(data, nrow, ncol, byrow, dimnames)

- data is the input vector which becomes the data elements of the matrix.
- nrow is the number of rows to be created.
- ncol is the number of columns to be created.
- byrow is a logical clue. If TRUE then the input vector elements are arranged by row.
- dimname is the names assigned to the rows and columns.

Array



01 Vector

02 Matrix

03 Array

04 List

05 Data Frame

❑ Arrays are the R data objects which can store data in more than two dimensions.

Syntax → array(data, dim, dimnames)

```
array( c(0:15), dim=c( 4,4,2,2) )
```

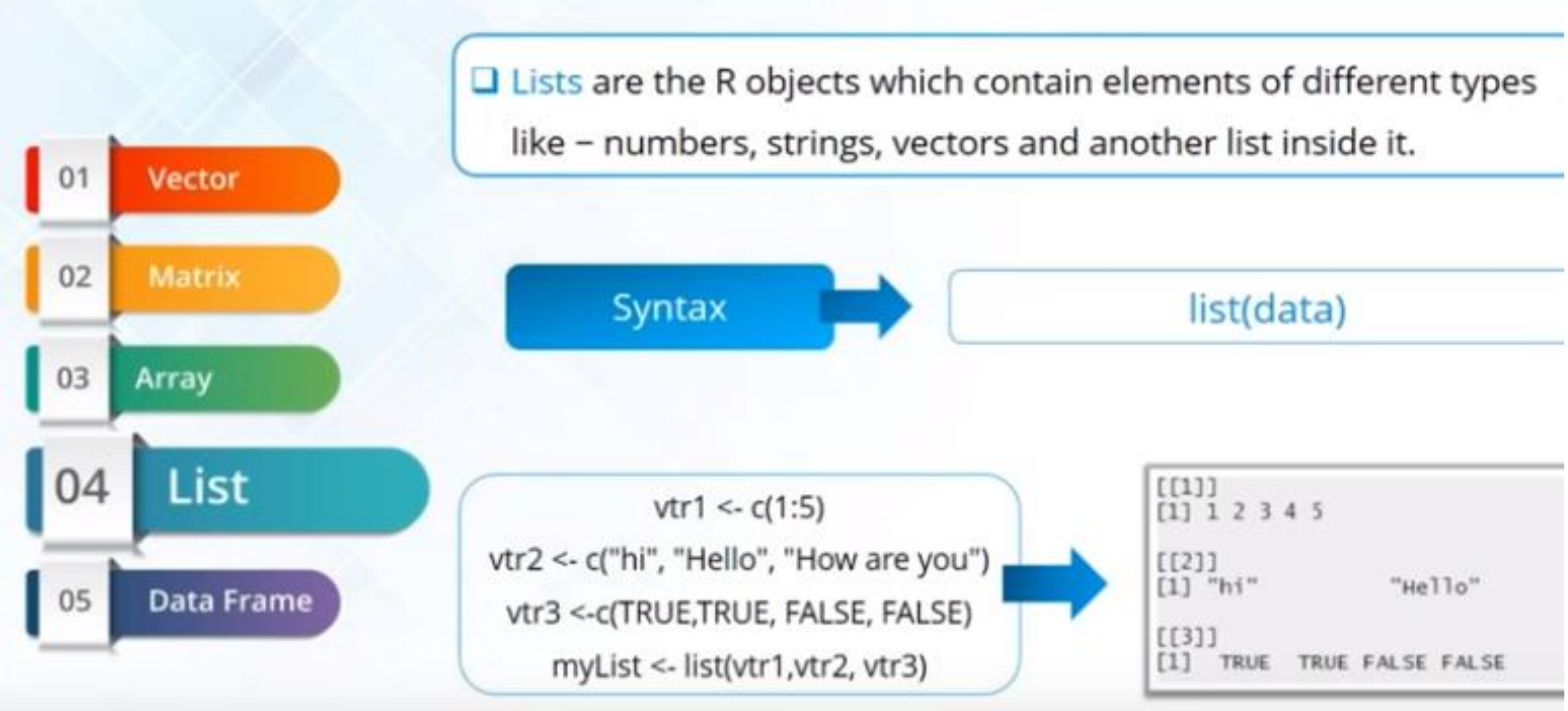
	0	1	2	3
0	0	4	8	12
1	1	5	9	13
2	2	6	10	14
3	3	7	11	15

	0	1	2	3
0	0	4	8	12
1	1	5	9	13
2	2	6	10	14
3	3	7	11	15

	0	1	2	3
0	0	4	8	12
1	1	5	9	13
2	2	6	10	14
3	3	7	11	15

	0	1	2	3
0	0	4	8	12
1	1	5	9	13
2	2	6	10	14
3	3	7	11	15

Lists



01 Vector

02 Matrix

03 Array

04 List

05 Data Frame

❑ Lists are the R objects which contain elements of different types like – numbers, strings, vectors and another list inside it.

Syntax → list(data)

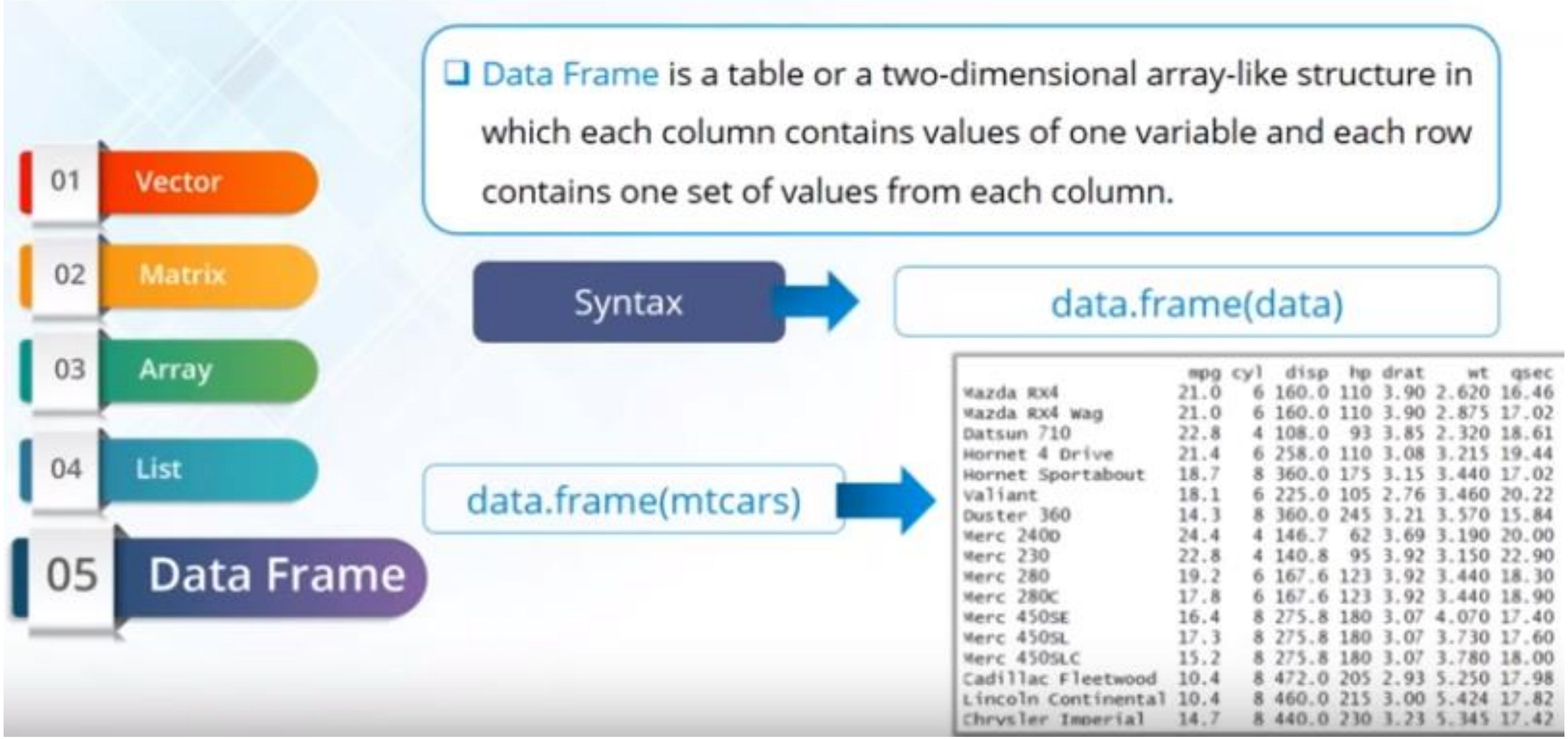
```
vtr1 <- c(1:5)
vtr2 <- c("hi", "Hello", "How are you")
vtr3 <-c(TRUE,TRUE, FALSE, FALSE)
myList <- list(vtr1,vtr2, vtr3)
```

```
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] "hi"          "Hello"

[[3]]
[1]  TRUE  TRUE FALSE FALSE
```

Data Frame



❑ Data Frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

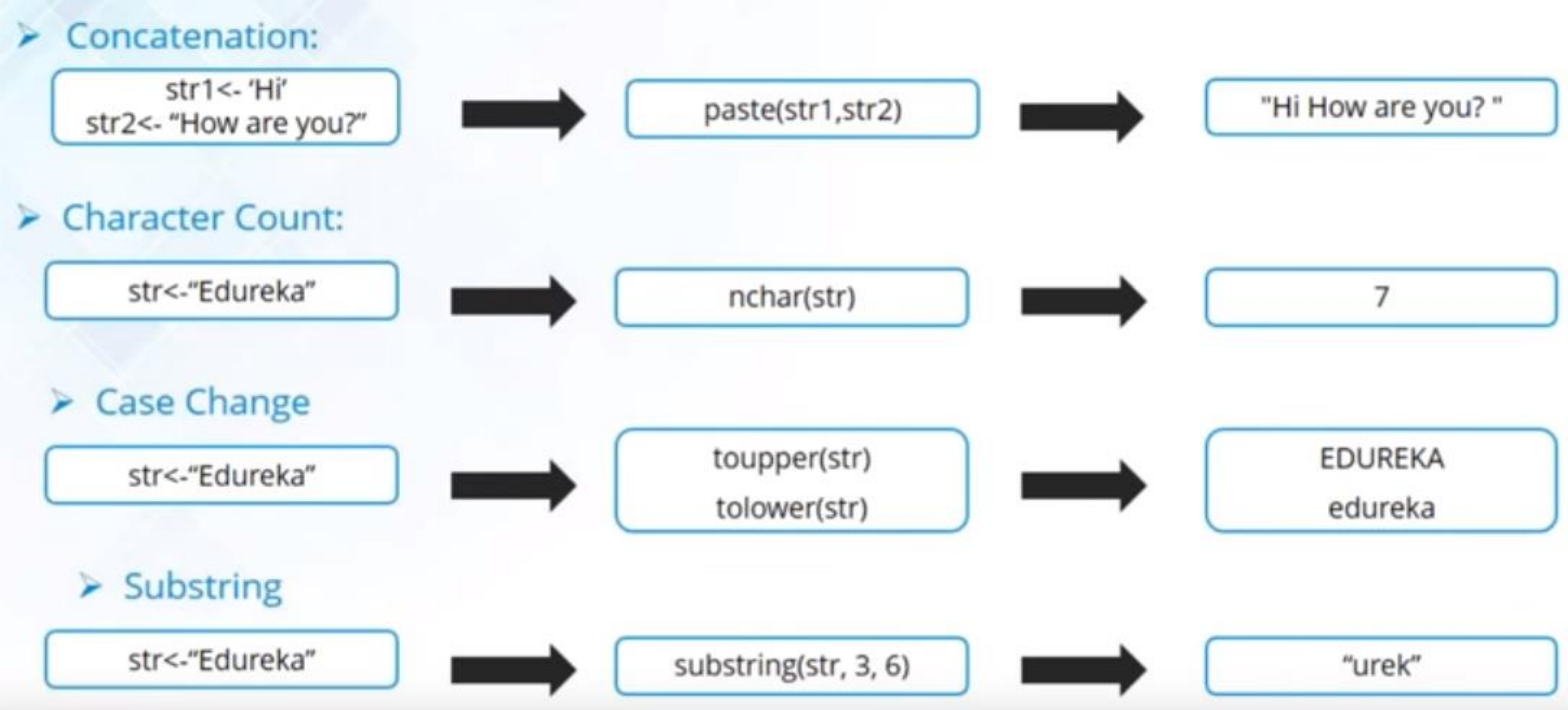
Syntax → data.frame(data)

data.frame(mtcars) →

```
                     mpg cyl  disp  hp drat    wt  qsec
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02
Valiant             18.1   6 225.0 105 2.76 3.460 20.22
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42
```

Strings

- Any value written with a pair of single of double quotes



➢ Concatenation:

str1<- ‘Hi’
str2<- “How are you?” → paste(str1,str2) → "Hi How are you? "

➢ Character Count:

str<-“Edureka” → nchar(str) → 7

➢ Case Change

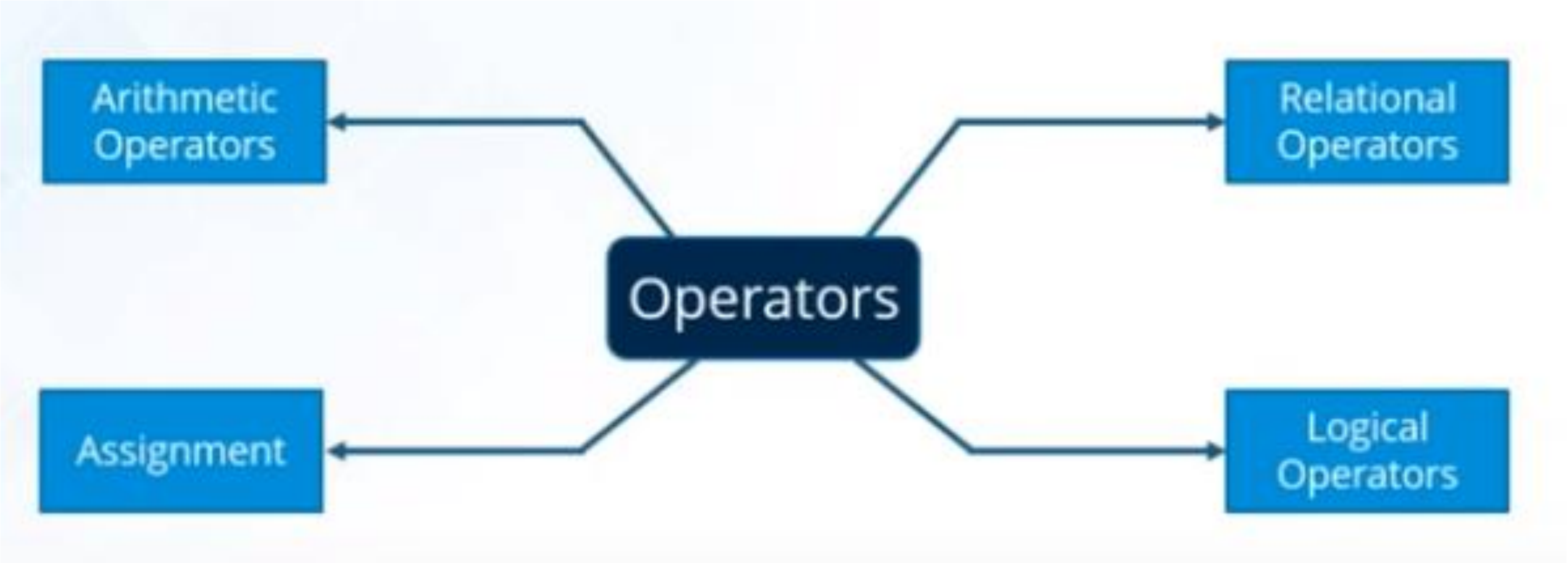
str<-“Edureka” → toupper(str)
tolower(str) → EDUREKA
edureka

➢ Substring

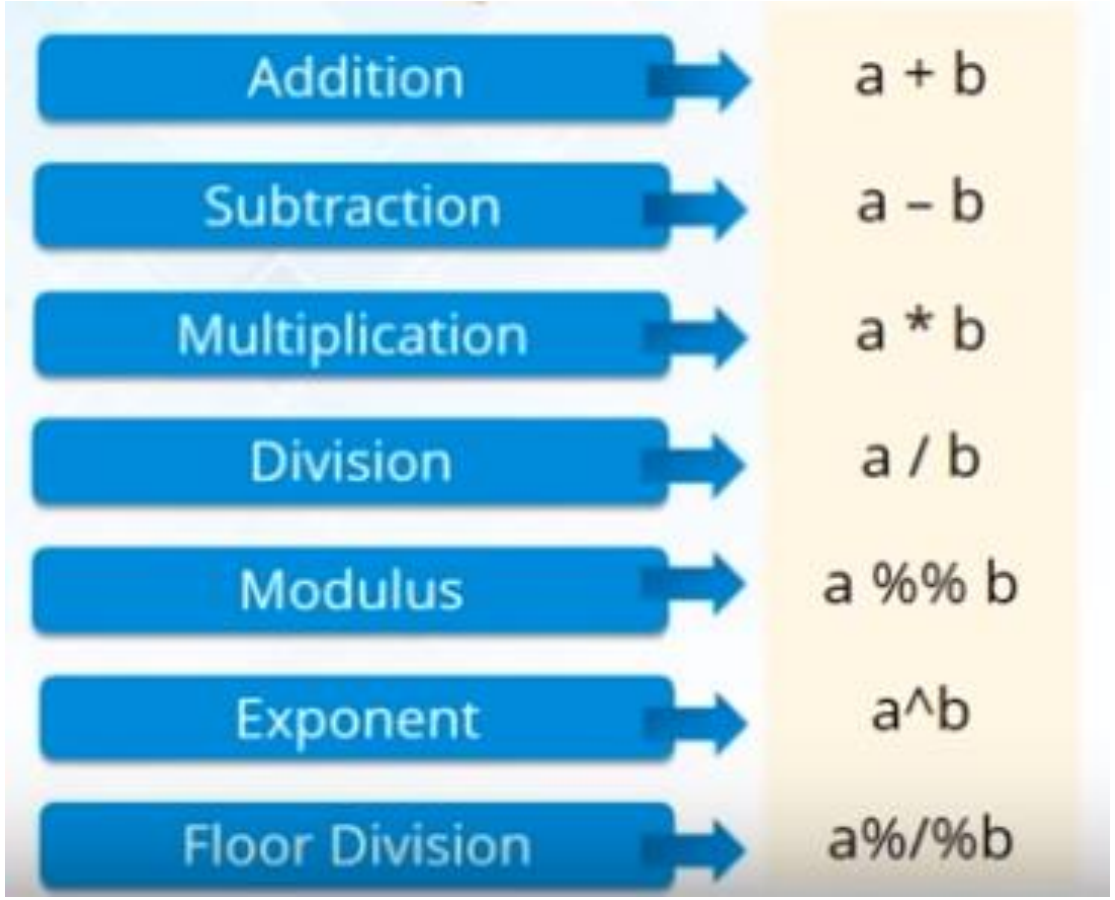
str<-“Edureka” → substring(str, 3, 6) → “urek”

Operators

- Constructs that can manipulate the value of operands

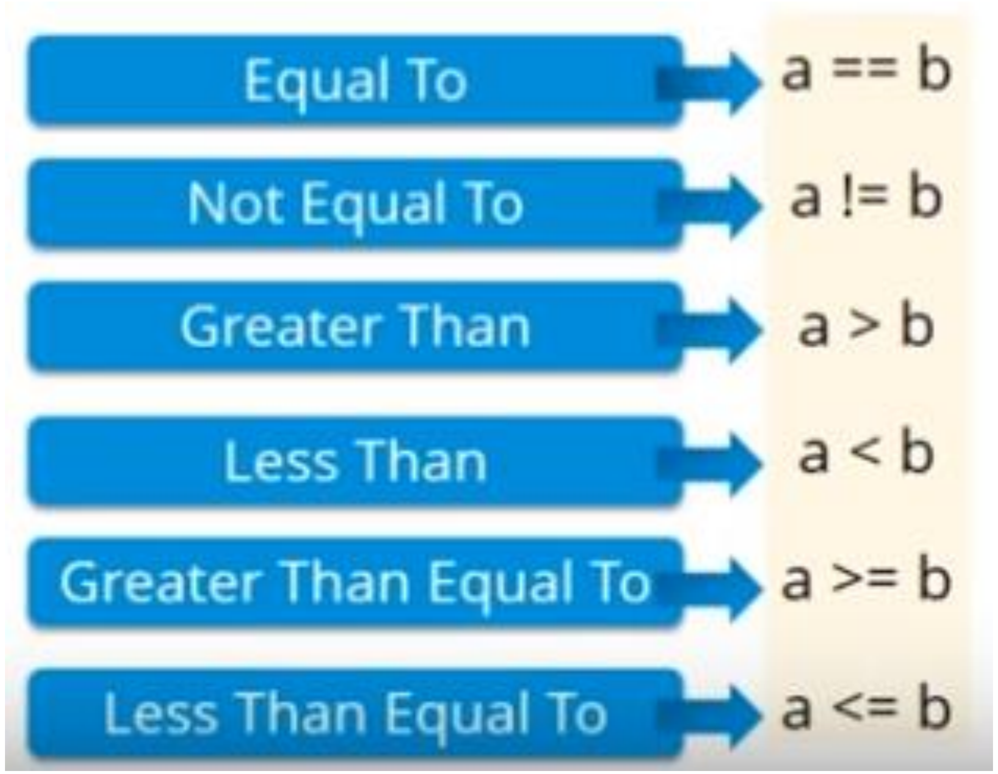


Arithmetic



Addition	a + b
Subtraction	a – b
Multiplication	a * b
Division	a / b
Modulus	a %% b
Exponent	a^b
Floor Division	a%/%b

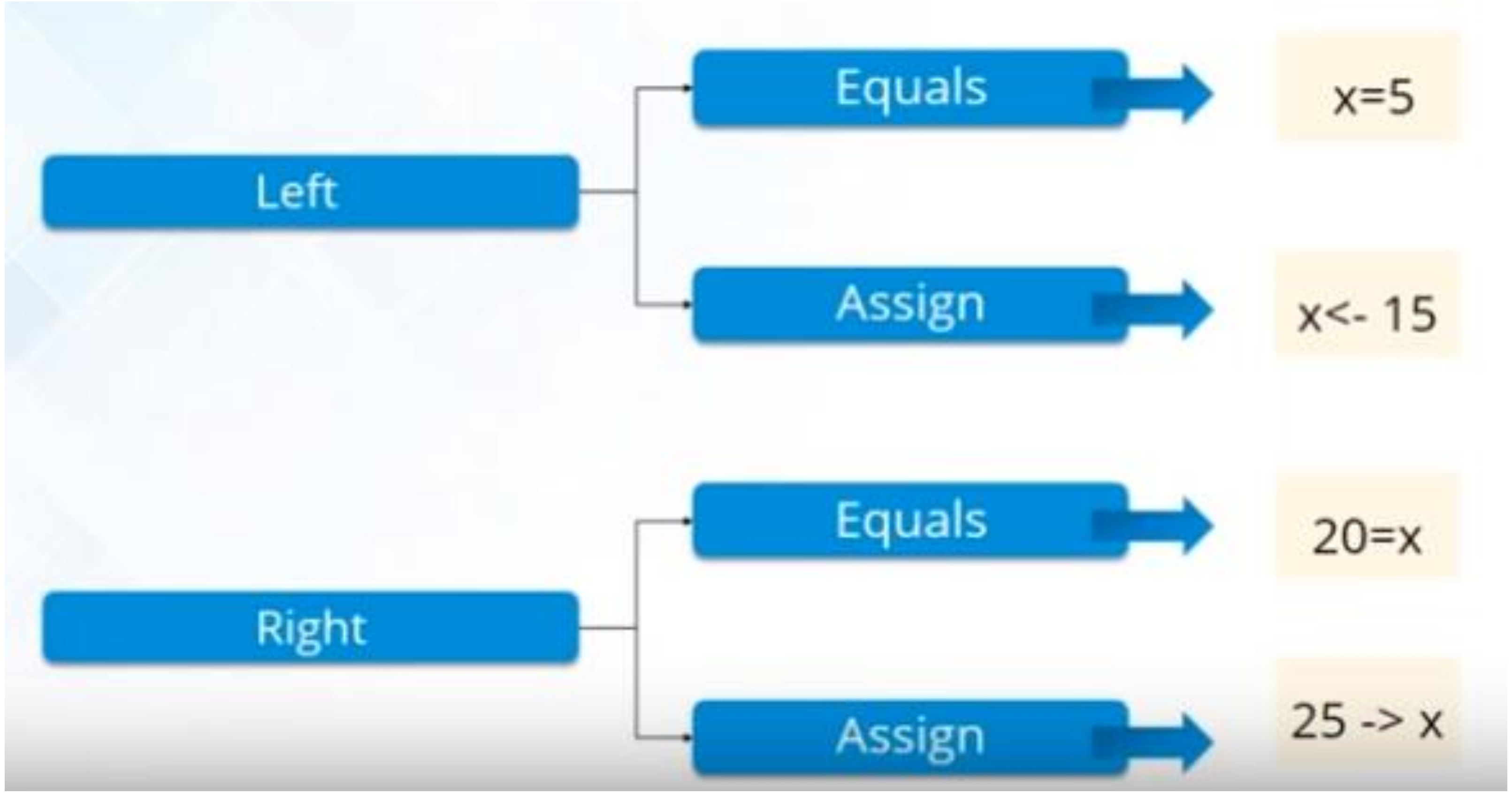
Relational



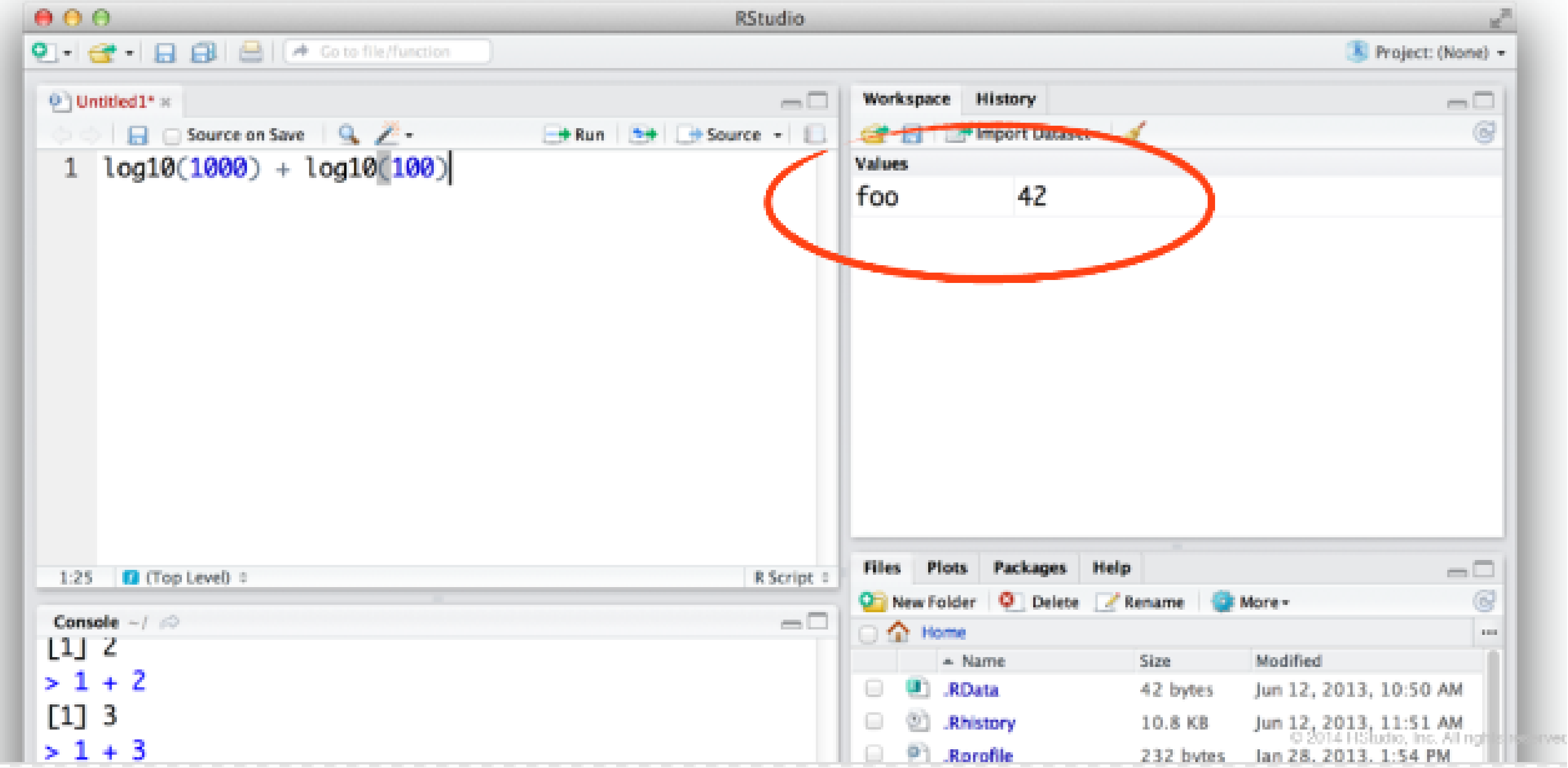
Equal To	a == b
Not Equal To	a != b
Greater Than	a > b
Less Than	a < b
Greater Than Equal To	a >= b
Less Than Equal To	a <= b

Assignment

- Left and Right



When you create an R object, you'll see it appear in your workspace pane



```
log10(1000) + log10(100)
```

Values

foo	42

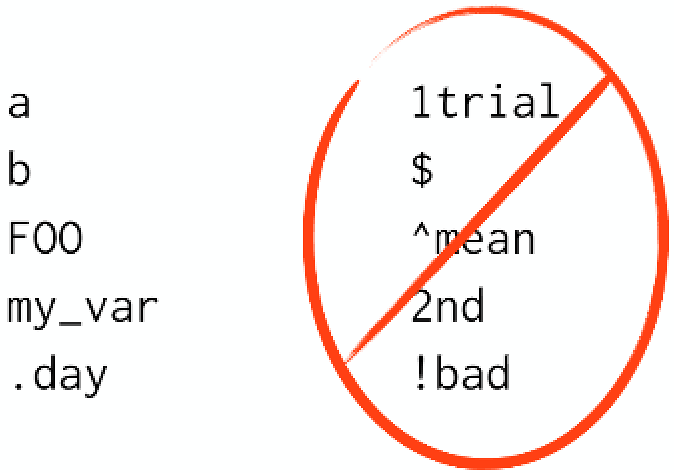
```
[1] 2
> 1 + 2
[1] 3
> 1 + 3
```

Name	Size	Modified
.RData	42 bytes	Jun 12, 2013, 10:50 AM
.Rhistory	10.8 KB	Jun 12, 2013, 11:51 AM
.Rprofile	232 bytes	Jan 28, 2013, 1:54 PM

Object names

Object names cannot begin with numbers.

Wise to avoid names already in use.



```
a          1trial
b          $
FOO        ^mean
my_var     2nd
.day       !bad
```

Capitalization matters

R will treat each of these as a different object

```
a        A
b        B
sum      SUM
```

More tips for variable names

- Make your names explicit and not too long.
- Avoid names of fundamental functions in R
  - e.g., if, else, for, see here for a complete list). In general, even if it's allowed, it's best to not use other function names (e.g., c, T, mean, data) as variable names. When in doubt check the help to see if the name is already in use.
- Avoid dots (.) within a variable name as in my.dataset
  - There are many functions in R with dots in their names for historical reasons, but because dots have a special meaning in R (for methods) and other programming languages, it's best to avoid them.
- Use nouns for object names and verbs for function names
- Be consistent with the styling of your code (where you put spaces, how you name variable, etc.)
  - In R, two popular style guides are Hadley Wickham's style guide and Google's.

rm

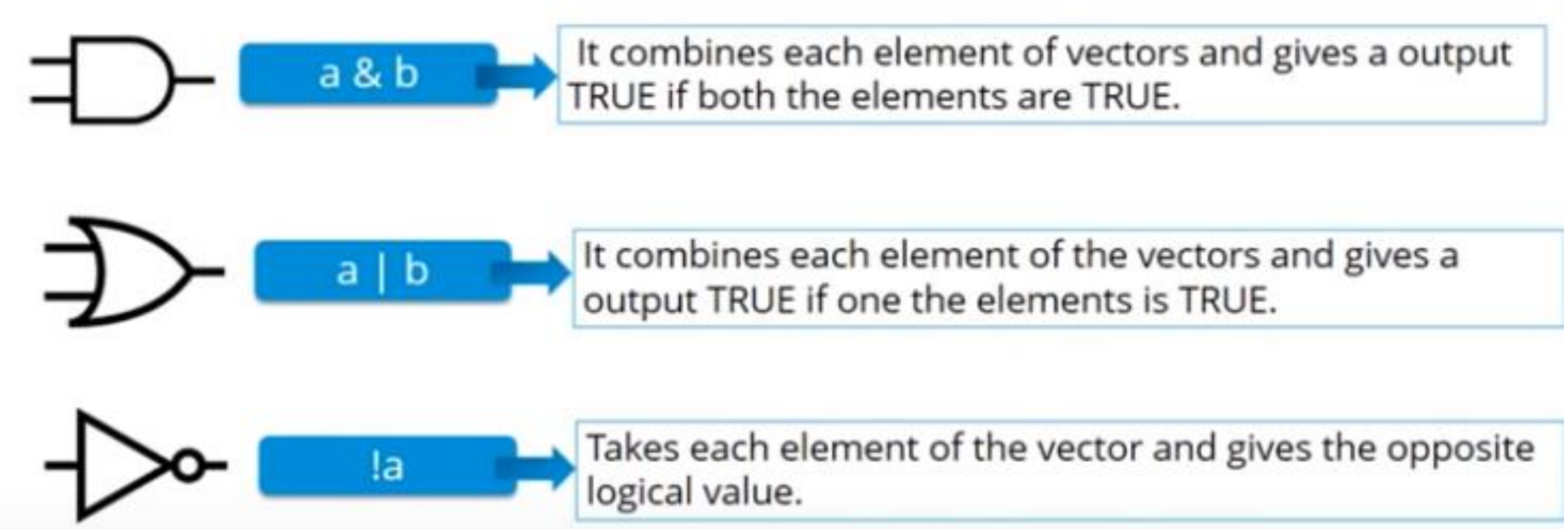
You can remove an object with `rm`

```
foo
# 4

rm(foo)
foo
# Error: object 'foo' not found
```

Logical

3 types: AND, NOT, OR



a & b

It combines each element of vectors and gives a output TRUE if both the elements are TRUE.

a | b

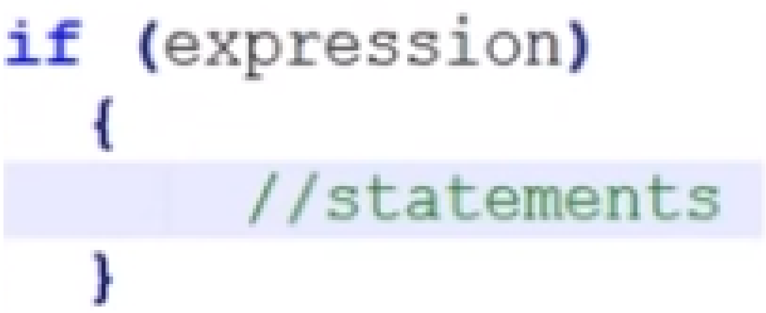
It combines each element of the vectors and gives a output TRUE if one the elements is TRUE.

!a

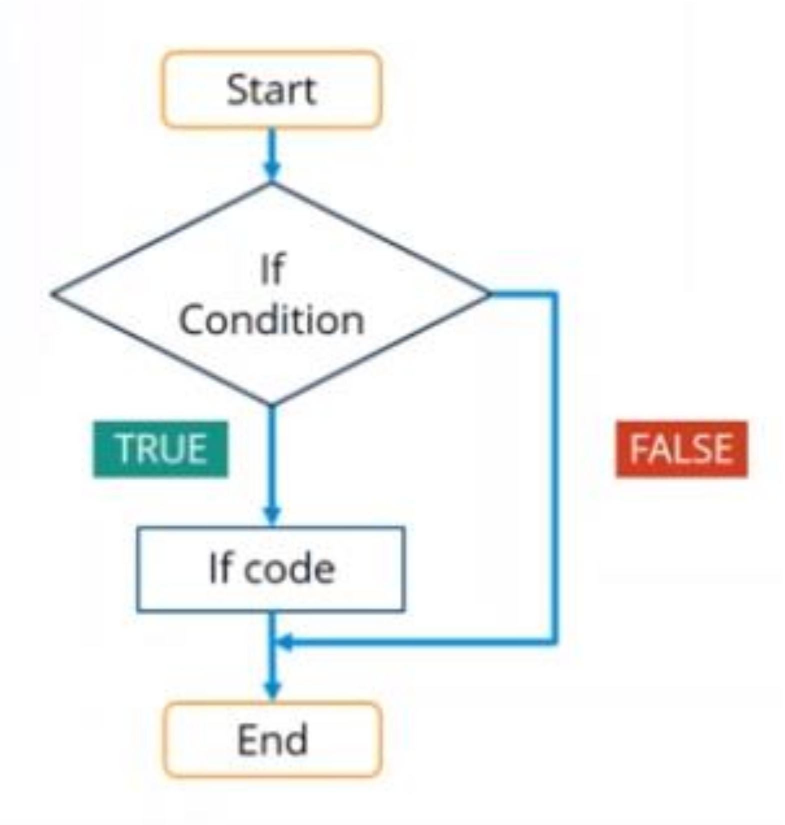
Takes each element of the vector and gives the opposite logical value.

Conditional statements

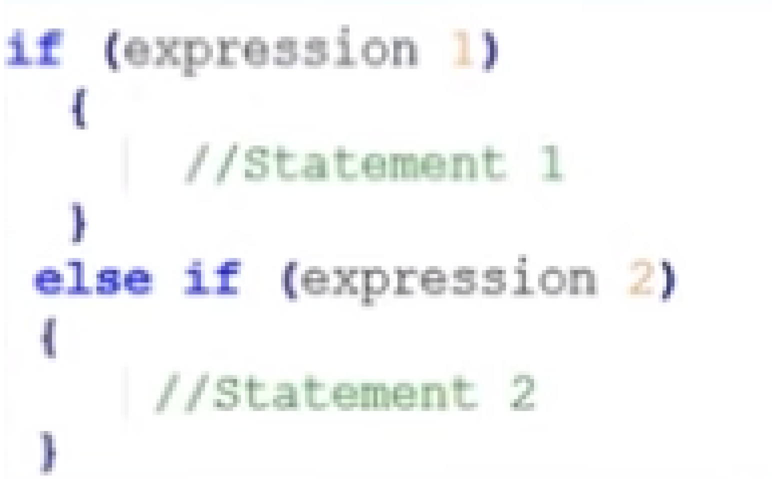
if statement



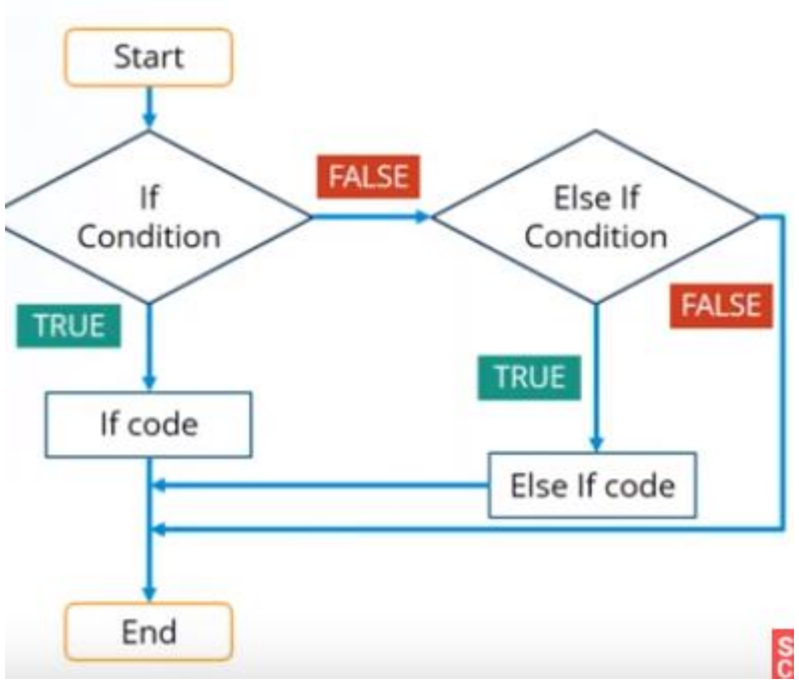
```
if (expression)
  {
        //statements
  }
```



- Else-if statement

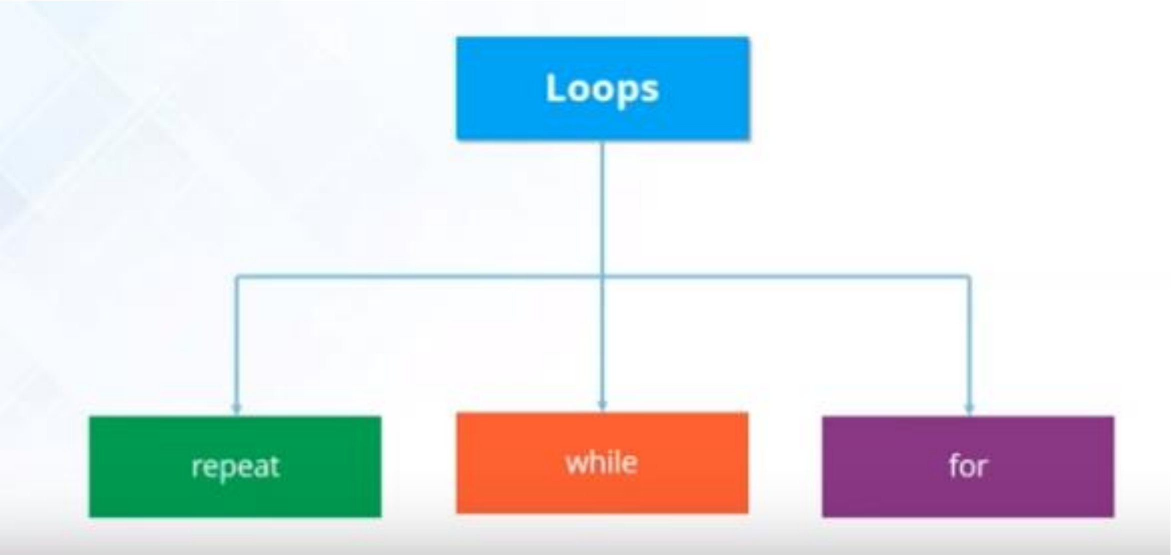


```
if (expression 1)
  {
      //Statement 1
  }
 else if (expression 2)
 {
      //Statement 2
 }
```



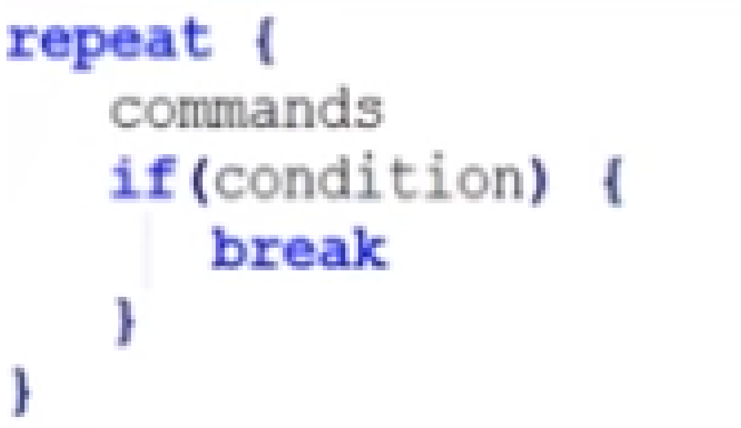
Loops

- Allows us to execute a statement or group of statements multiple times.

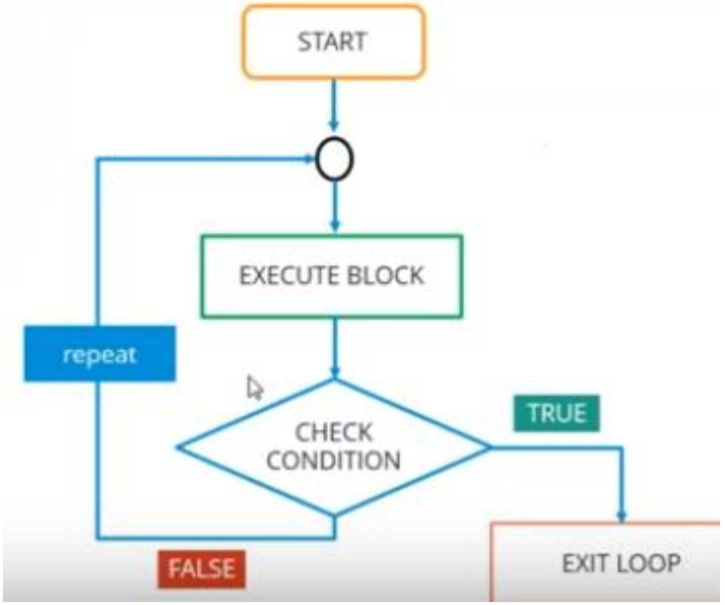


Repeat

- Test condition AFTER executing body

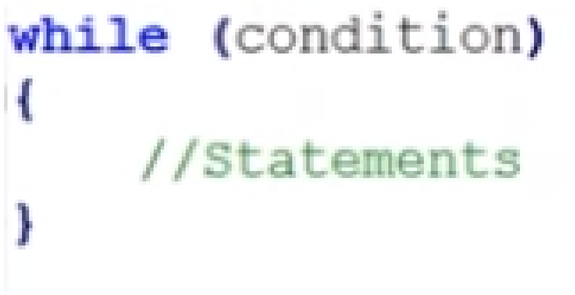


```
repeat {
    commands
    if(condition) {
        break
    }
}
```

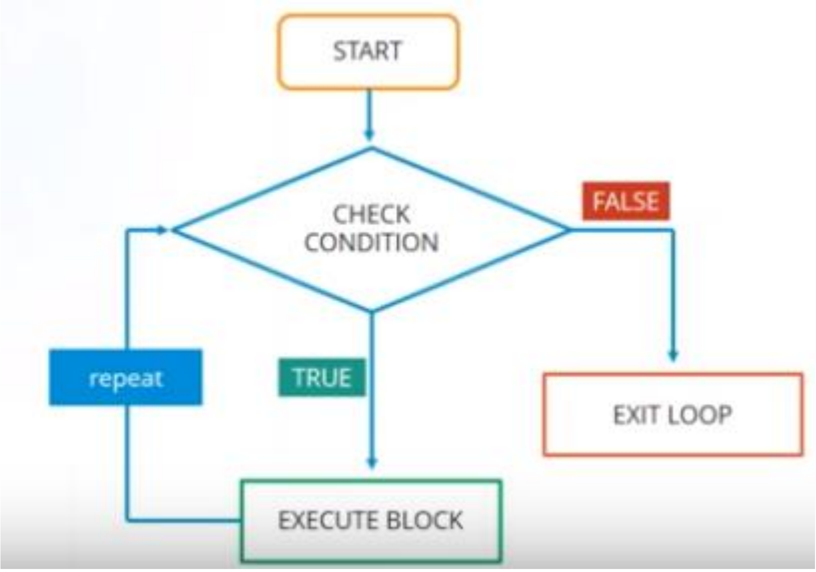


While

- Test condition BEFORE executing body

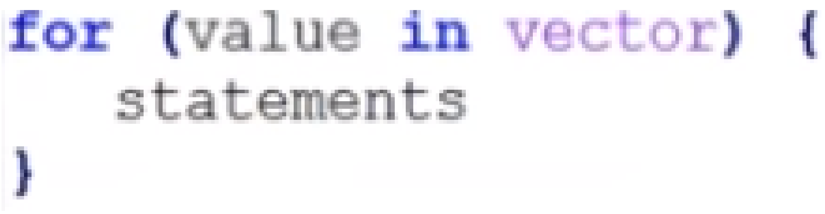


```
while (condition)
{
     //Statements
}
```

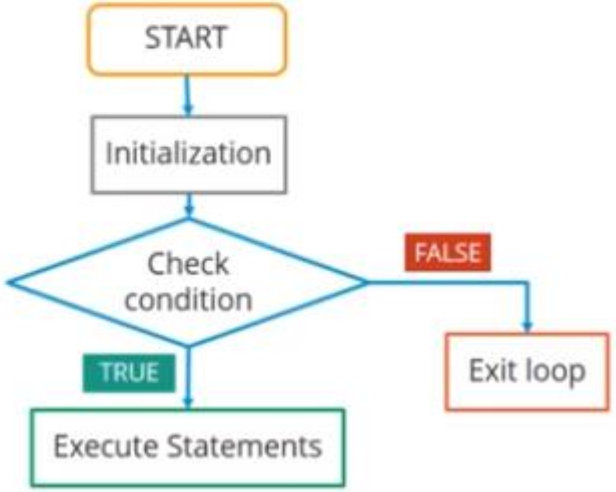


For loop

- Repeats body a fixed number of times



```
for (value in vector) {
   statements
}
```



START
Initialization
Check condition
FALSE
Exit loop
TRUE
Execute Statements