Rsquared Academy

Variables & Data Types

In this module, we will learn about variables & data types. Specifically:

- what is a variable?

- how to create variables in R?

- how to use variables created?

- components of a variable

- naming conventions for a variable

- data types

    - numeric/double
    - integer
    - logical
    - character
    - date/time

variables

What is a variable?

- variables are the fundamental elements of any programming language

- they are used to represent values that are likely to change

- they reference memory locations that store information/data

Let us use a simple case study to understand variables. Suppose you are computing the area of a circle whose radius is 3. In R, you can do this straight away as shown below:

```
3.14 * 3 * 3
```

```
## [1] 28.26
```

But you cannot reuse the radius or the area computed in any other analysis or computation. Let us see how variables can change the above scenario and help us in reusing values and computations.

A variable consists of 3 components:

- variable name

- assignment operator

- variable value

We can store the value of the radius by creating a variable and assigning it the value. In this case, we create a variable called `radius` and assign it the value 3 using the assignment operator `<-`.

```
radius <- 3
radius
```

```
## [1] 3
```

Now that we have learnt to create variables, let us see how we can use them for other computations. For our case study, we will use the `radius` variable to compute the area of a circle.

We will create two variables, `radius` and `pi`, and use them to compute the area of a circle and store it in another variable `area`.

```
# assign value 3 to variable radius
radius <- 3

# assign value 3.14 to variable pi
pi <- 3.14

# compute area of circle
area <- pi * radius * radius

# call radius
radius
```
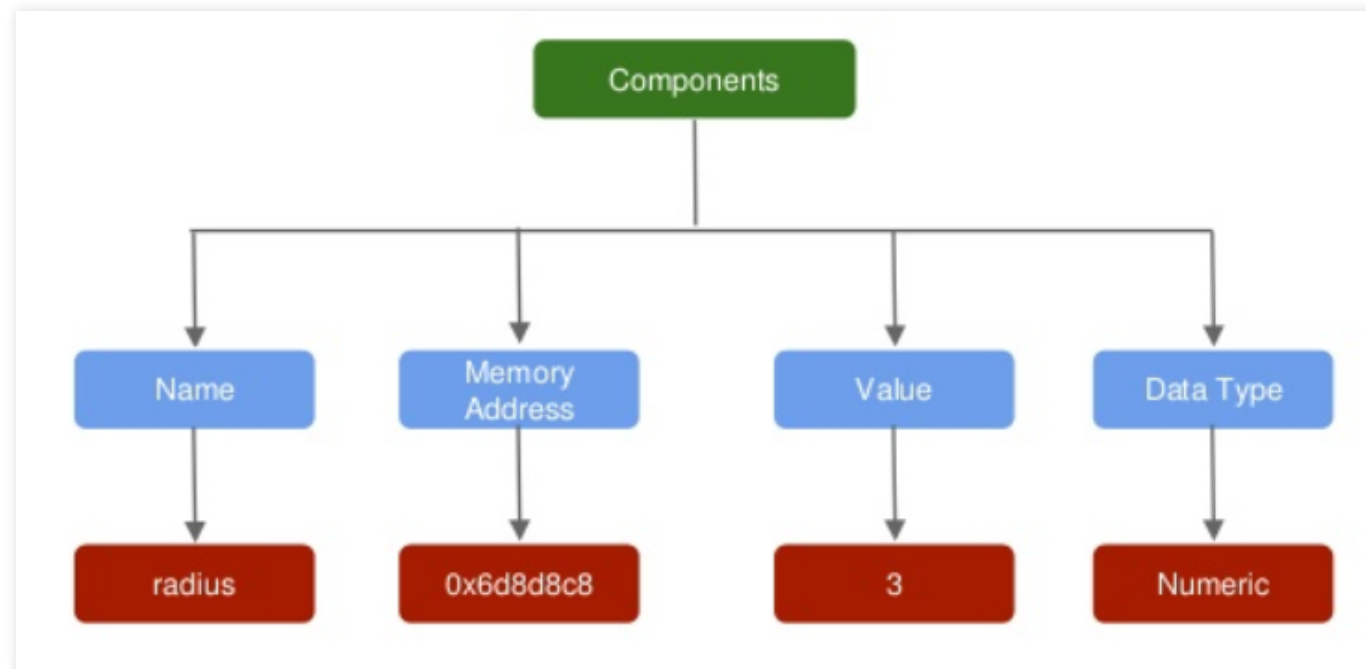
```
## [1] 3
```

```
# call area
area
```

```
## [1] 28.26
```

Naming Conventions

- Name must begin with a letter. Do not use numbers, dollar sign ($) or underscore (_).

- The name can contain numbers or underscore. Do not use dash (-) or period (.).

- Do not use the names of keywords and avoid using the names of built in functions.

- Variables are case sensitive; `average` and `Average` would be different variables.

- Use names that are descriptive. Generally, variable names should be nouns.

- If the name is made of more than one word, use underscore to separate the words.

data types

In R, numbers are represented by the data type `numeric`. We will first create a variable and assign it a value. Next we will learn a few methods of checking the type of the variable.

```r
# create two variables
number1 <- 3.5
number2 <- 3

# check data type
class(number1)
## [1] "numeric"
class(number2)

## [1] "numeric"

# check if data type is numeric
is.numeric(number1)
## [1] TRUE
is.numeric(number2)
## [1] TRUE
```

If you carefully observe, `integers` are also treated as `numeric/double`. We will learn to create integers in a while. In the meanwhile, we have introduced two new functions in the above example:

- `class()`: returns the `class` or `type`

- `is.numeric()`: tests whether the variable is of type `numeric`

Integer

Unless specified otherwise, integers are treated as `numeric` or `double`. In this section, we will learn to create variables of the type `integer` and to convert other data types to `integer`.

- create a variable `number1` and assign it the value 3

- check the data type of `number1` using `class`

- create a second variable `number2` using `as.integer()` and assign it the value 3

- check the data type of `number2` using `class`

- finally use `is.integer()` to check the data type of both `number1` and `number2`

```r
# create a variable and assign it an integer value
number1 <- 3

# create another variable using as.integer
number2 <- as.integer(3)

# check the data type
class(number1)
## [1] "numeric"
class(number2)
## [1] "integer"


# use is.integer to check data type
is.integer(number1)
## [1] FALSE
is.integer(number2)
## [1] TRUE
```

Letters, words and group of words are represented by the data type `character`. All data of type `character` must be enclosed in single or double quotation marks. In fact any value enclosed in quotes will be treated as `character`. Let us create two variables to store the first and last name of a some random guy.

```r
# first name
first_name <- "jovial"

# last name
last_name <- 'mann'

# check data type
class(first_name)
## [1] "character"
class(last_name)
## [1] "character"

# use is.charactert to check data type
is.character(first_name)
## [1] TRUE
is.character(last_name)
## [1] TRUE
```

You can coerce any data type to `character` using `as.character()`.

```r
# create variable of different data types
age        <- as.integer(30) # integer
score      <- 9.8            # numeric/double
opt_course <- TRUE           # logical
today      <- Sys.time()     # date time

as.character(age)
## [1] "30"
as.character(score)

## [1] "9.8"
as.character(opt_course)
## [1] "TRUE"
as.character(today)
## [1] "2018-08-03 14:05:30"
```

Logical data types take only 2 values. Either TRUE or FALSE. Such data types are created when we compare two objects in R using comparison or logical operators.

- create two variables x and y

- assign them the values TRUE and FALSE respectively

- use `is.logical()` to check data type

- use `as.logical()` to coerce other data types to `logical`

```r
# create variables x and y
x <- TRUE
y <- FALSE

# check data type
class(x)
## [1] "logical"
is.logical(y)
## [1] TRUE
```

Logical

The outcome of comparison operators is always `logical`. In the below example, we compare two numbers to see the outcome.

```
# create two numeric variables
x <- 3
y <- 4

# compare x and y
x > y
## [1] FALSE
x < y

## [1] TRUE

# store the result
z <- x > y
class(z)
## [1] "logical"
```

Logical

TRUE is represented by all numbers except `0`. FALSE is represented only by `0` and no other numbers.

```
# TRUE and FALSE are represented by 1 and 0
as.logical(1)
## [1] TRUE
as.logical(0)
## [1] FALSE

# using numbers
as.numeric(TRUE)
## [1] 1

as.numeric(FALSE)
## [1] 0

# using different numbers
as.logical(-2, -1.5, -1, 0, 1, 2)
## [1] TRUE
```

Use `as.logical()` to coerce other data types to `logical`.

```r
# create variable of different data types
age         <- as.integer(30) # integer
score       <- 9.8            # numeric/double
opt_course  <- TRUE          # logical
today       <- Sys.time()    # date time

as.logical(age)
## [1] TRUE
as.logical(score)

## [1] TRUE
as.logical(opt_course)
## [1] TRUE
as.logical(today)
## [1] TRUE
```

- Variables are the building blocks of a programming language

- Variables reference memory locations that store information/data

- An assignment operator <- assigns values to a variable

- A variable has the following components:

  - name
  - memory address

  - value
  - data type

- Certain rules must be followed while naming variables

- numeric, integer, character, logical and date are the basic data types in R

- `class()` or `typeof()` return the data type

- `is.data_type` checks whether the data is of the specified data type

  - `is.numeric()`
  - `is.integer()`
  - `is.character()`
  - `is.logical()`

  - `is.date()`

- `as.data_type` will coerce objects to the specified data type

  - `as.numeric()`
  - `as.integer()`
  - `as.character()`
  - `as.logical()`
  - `as.date()`

Thank You

For more information please visit our website
www.rsquaredacademy.com