**What are Data types in R?**

**R Data types** are used to specify the kind of data that can be stored in a variable.

For effective memory consumption and precise computation, the right data type must be selected.

Each R data type has its own set of regulations and restrictions.

Variables are not needed to be declare with a data type in R, data type even can be changed.

Data Types in R Programming Language

Each variable in R has an associated data type. Each R-Data Type requires different amounts of memory and has some specific operations which can be performed over it.

Data Types in R are:

1. numeric – (3,6.7,121)
2. Integer – (2L, 42L; where ‘L’ declares this as an integer)
3. logical – (‘True’)
4. complex – (7 + 5i; where ‘i’ is imaginary number)
5. character – (“a”, “B”, “c is third”, “67”)
6. raw – ( as.raw(55); raw creates a raw vector of the specified length)

**Numeric Data type**

Decimal values are called numeric in R. It is the default  R data type for numbers in R.

If you assign a decimal value to a variable x as follows, x will be of numeric type.

Real numbers with a decimal point are represented using this data type in R. It uses a format for double-precision floating-point numbers to represent numerical values.

# A simple R program to illustrate Numeric data type

# Assign a decimal value to x

x = 5.6

# print the class name of variable

print(class(x))

# print the type of variable

print(typeof(x))

Even if an integer is assigned to a variable y, it is still saved as a numeric value.

|  |
| --- |
| # A simple R program to illustrate Numeric data type  # Assign an integer value to y  y = 5  # print the class name of variable  print(class(y))  # print the type of variable  print(typeof(y)) |

When R stores a number in a variable, it converts the number into a “double” value or a decimal type with at least two decimal places.

This means that a value such as “5” here, is stored as 5.00 with a type of double and a class of numeric.

**Integer Data type**

R supports integer data types which are the set of all integers.

You can create as well as convert a value into an integer type using the **as.integer()** function.

You can also use the capital ‘L’ notation as a suffix to denote that a particular value is of the integer R data type.

|  |
| --- |
| # A simple R program to illustrate integer data type  # Create an integer value  x = as.integer(5)  # print the class name of x  print(class(x))  # print the type of x  print(typeof(x))  # Declare an integer by appending an L suffix.  y = 5L  # print the class name of y  print(class(y))  # print the type of y  print(typeof(y)) |

**Complex Data type**

R supports complex data types that are set of all the complex numbers. The complex data type is to store numbers with an imaginary component.

|  |
| --- |
| # A simple R program to illustrate complex data type  # Assign a complex value to x  x = 4 + 3i  # print the class name of x  print(class(x))  # print the type of x  print(typeof(x)) |

**Logical Data type**

R has logical data types that take either a value of **true**or **false**.

A logical value is often created via a comparison between variables.

Boolean values, which have two possible values, are represented by this R data type: FALSE or TRUE

|  |
| --- |
| # A simple R program to illustrate logical data type  # Sample values  x = 4  y = 3  # Comparing two values  z = x > y  # print the logical value  print(z)  # print the class name of z  print(class(z))  # print the type of z  print(typeof(z)) |

**Character data type**

R supports character data types where you have all the alphabets and special characters.

It stores character values or strings. Strings in R can contain alphabets, numbers, and symbols.

The easiest way to denote that a value is of character type in R data type is to wrap the value inside single or double inverted commas.

|  |
| --- |
| # A simple R program to illustrate character data type  # Assign a character value to char  char = "Hello All!!!"  # print the class name of char  print(class(char))  # print the type of char  print(typeof(char)) |

**Type verification**

You can verify the data type of an object, if you doubt about it’s data type.

To do that, you need to use the prefix “is.” before the data type as a command.

**Syntax:**

* is.data\_type(object)

|  |
| --- |
| # A simple R program  # Verify if an object is of a certain datatype    # Logical  print(is.logical(**TRUE**))  # Integer  print(is.integer(3L))  # Numeric  print(is.numeric(10.5))  # Complex  print(is.complex(1+2i))  # Character  print(is.character("12-04-2020"))  print(is.integer("a"))  print(is.numeric(2+3i)) |

**Coerce or Convert the Data Type of an Object to Another**

The process of altering the data type of an object to another type is referred to as coercion or data type conversion. This is a common operation in many programming languages that is used to alter data and perform various computations.

When coercion is required, the language normally performs it automatically, whereas conversion is performed directly by the programmer.

Coercion can manifest itself in a variety of ways, depending on the R programming language and the context in which it is employed.

In some circumstances, the coercion is implicit, which means that the language will change one type to another without the programmer having to expressly request it.

**Syntax**

as.data\_type(object)

|  |
| --- |
| # A simple R program convert data type of an object to another  # Logical  print(as.numeric(**TRUE**))  # Integer  print(as.complex(3L))  # Numeric  print(as.logical(10.5))  # Complex  print(as.character(1+2i))  # Can't possible  print(as.numeric("12-04-2020")) |