Data Types

Let's begin with discussing the basic data types in R, declaring variables and more.

We'll cover the following

Basic Data Types in R

> Everything in R is an object.

R is a language designed to store, manipulate, and work with *Data*. Thus, it makes sense to make all utilities of R an **object** that stores *data*.

In every computer language, **variables** are used for accessing data stored in memory. R does not provide direct access to the computer's memory but rather provides several specialized **data structures** referred to as **objects**. These objects can be symbols, variables, functions, etc.

Variables in R

Variables are used to **store data**. Their value can be changed, used and manipulated according to need. A unique name given to a variable (function or object as well) is called an `identifier`.

> **Important points while writing Identifiers in R:** Identifiers can have a combination of letters, digits, one period `.` and one underscore `_`. However, they must start with a **letter** or a **period**. If it starts with a period, it cannot be followed by a **digit**. Reserved words in R cannot be used as identifiers!

Reserved words are keywords that have special meaning and functions, for example, `pi` is a reserved word. It has a value of $3.141...$ As we move ahead in the course we will encounter many reserved words and learn their functions.

To declare a variable, we need to assign a variable an identifier.

To add value to the variable, use `<-` or `=`.

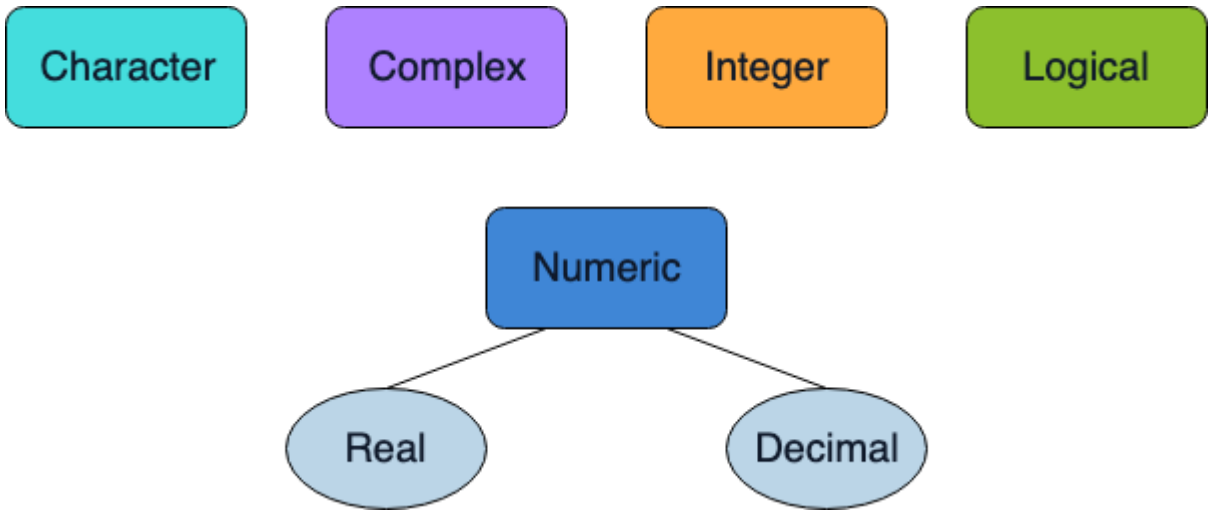
Syntax for declaring variables in R

```
# declaring a variable using `<-`
nameOfVariable <- value

# declaring a variable using `=`
nameOfVariable = value
```

Main Data Types

To make the best of the R language, we need a strong understanding of the basic data types in R. R has **five** main data types.



In the following code, we declare the different types of variables and use `class()` to determine **what kind of object** it is. This is called the **high-level** data type of an object.

```
# Variables of different types

# Real Numeric
myRealNumeric <- 10
class(myRealNumeric)

# Decimal Numeric
```

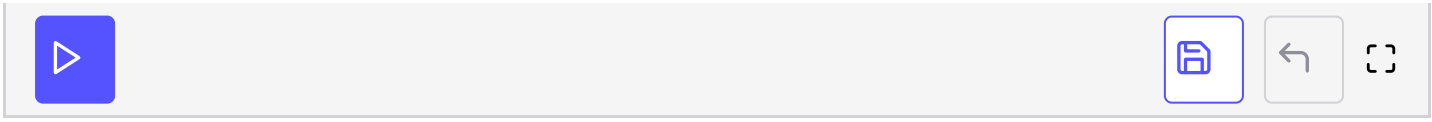
```
# Decimal Numeric
myDecimalNumeric <- 10.0
class(myDecimalNumeric)

# Character
myCharacter <- "10"
class(myCharacter)

# Logical
myBoolean <- TRUE
class(myBoolean)

# Integer
myInteger <- 0:10
class(myInteger)

#Complex
myComplex <- 5i
class(myComplex)
```

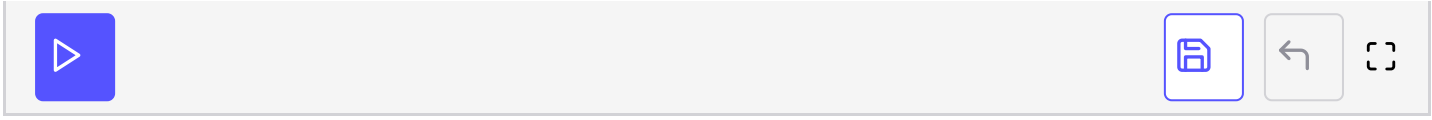


Defining variables of different data types

To determine the **data type of the object** at the **low level** we can use `typeof()`. Another method that we'd like to introduce here is `length()` that returns the length of the data in the variable.

In the code snippet below, we print the **type** and **length** of the variable `myInteger`:

```
# Testing typeof() and length()
myInteger <- 1:10
cat("The type of the variable is ", typeof(myInteger), "\n") # returns low level data type
cat("The length of the variable is ", length(myInteger), "\n") # returns length of variable
```

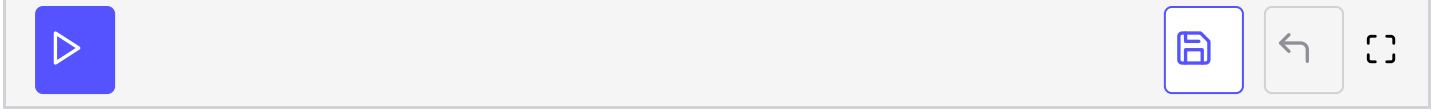


Using typeof() and length()

In R language, if you change the data in a variable, i.e., **overwrite** any previous information stored in an object, then the previous information will be deleted. So remember not to use names that are already taken.

```
myInteger <- 1:10
cat("myInteger: ")
cat(myInteger, "\n")

myInteger <- 11:20 # Overwriting myInteger
cat("myInteger: ")
cat(myInteger, "\n")
```



Overwriting a variable

We will learn how to find and delete all the variables used in the current workspace of R in the next lesson.