```r
#######COURSE: Basics of R programming language for statistical analysis########

#Instructor: Marina FERENT-PIPAS
#email: marinaferent@gmail.com
#Multicultural Business Institute | mcb-institute.org

#########MEETING 01: Basic notions | 05.08.2021 ##################################

#################### GOOD PROGRAMMING PRACTICES OF THE MEETING:

    ## 1. comment, comment, comment - Challenge of the course: try to comment every line of code you write:
        # - use "#" to comment any line of your code = write "#" in front of any line that contains extra information about your code (it's not your code)
        # - R does not have a built-in function to comment out sections = no multi-line comment function in R


    ## 2. asign represetative names to each structure that you write:
        # - eg. a vector containing the performance of the participants in the R course could be named:
            #performance, performancePart, performance_part, performanceParticipants, performance_participants -> good practices
            #a, marina, performancepart -> bad practices

# These practices are:
    ## - extremely useful when working in groups and sharing codes
    ## - extremely useful for future use of the code

#############################################################################

#################### FOCUS OF THE MEETING: USE OF FUNCTIONS

# an R function - functionName(), where:
    ##functionName = name of the function
    ##whatever is in the brackets is called the arguments of a function

# whenever in doubt about how to specify an argument of a function check the function's documentation:
    ## - type <<??functionName>> in the R console and it will direct you to the function's documentation
    ## - type <<functionName() in R>> in Google
        ### eg:

# sum()     - https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/sum
# c()       - https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/c
# cbind()   - https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/cbind
# sort()    - https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/sort

    ###EXERCISE POINT_1: Check the function's documentation for <<c()>>. What does the <<c>> in <<c()>> function stand for?

#############################################################################

######################### BASIC OPERATIONS IN R

20+3        #summing up 2 values - simply type in the R console and it will return the value 23
20+3+20     #summing up 3 (more) values -  simply type in the R console and it will return the value 43
sum(20,3)   #summing up 2 values - simply type in the R console and it will return the value 23
sum(20,3,20)#summing up 3 (more) values -  simply type in the R console and it will return the value 43
20-3        #difference of 2 (more) values - simply type in the R console and it will return the value 17
sum(20,-3)  #difference of 2 (more) values - simply type in the R console and it will return the value 17
20*3        #product of 2(more) values - simply type in the R console and it will
```

```
return the value 60
prod(20,3)  #product of 2(more) values - simply type in the R console and it will return the value 60
20/3        #division of 2(more) values - simply type in the R console and it will return the value 6.666667
20^3        #y powered x - simply type in the R console and it will return the value 8000
sqrt(20)    #square root of 20 - simply type in the R console and it will return the value 4.472136
20^(1/2)    #square root of 20 - simply type in the R console and it will return the value 4.472136
        ### etc. etc. EXERCISE POINT_2: find the built-in function for power (other than 20^3)
            ####see for example: sum can be computed as 20+3, but also as sum(20,3)



#asigning values to variables
value1=20       #stores the value 20 in variable value1
value2=3        #stores the value 3 in variable value2
    #alternatively you may use "<-" instead of "="
value1<-20      #stores the value 20 in variable value1
value2<-3       #stores the value 3 in variable value2

value1          #prints the value1 variable - simply type in the R console and it will return 20
value1+value2   #computes the sum of value1 (20) and value2 (3) variables- simply type in the R console and it will return the value 23
    #all of the above listed operations work here as well
value3=value1+value2 #computes the sum of value1 (20) and value2 (3) variables and stores it in value3 variable
value3          #prints the value of value3 variable (23)


###########R STRUCTURES, PROPERTIES AND OPERATIONS: Vectors and Data Frames (part 1)
#**********Numerical representation of attributive statistical variables (part 1)

#_____________________________________________________
#@REMEMBER FROM DESCRIPTIVE STATISTICS:
 ##@statistical population = a collection of items that share the same characteristics - eg. students
 ##@statistical unit = one item in the population - eg. 1 student
 ##@statistical variables = characteristics of the statistical unit -eg. performance, age, eye colour etc.
    #******1. Quantitative variables - eg. performance (measured as grades from 1-10), age, height
    #******2. Qualitative variables - eg. eye colour
#_____________________________________________________


#DEFINE A VECTOR. VISUALISE A VECTOR'S VALUES
#*Numerical representation of attributive statistical variables: list of observations
    #*EXERCISE 1: Collect the performance of 10 students (measured in grades from 1-10) in descriptive statistics class 2021 - list of observations

performance=c(10, 9, 7, 8, 5, 10, 8, 8, 6, 9) #constructs the vector of performance (stores the values of performance into vector performance)
                                              ##the function c() combines the grades of the 10 students into the vector performance
            ###EXERCISE POINT_4: What is the grade of the 7th student?
performance                                   #prints values of vector performance - simply type in the R console and it will return 10 9 7 8 5 10 8 8 6 9
View(performance)                             #returns a table with the values of performance !wrie View with capital "V"
        ### EXERCISE POINT_3: The software returns <<Error in view(performance) : could not find function "view">>. Which is the problem?


```

```r
#VECTOR PROPERTIES
is.vector(performance)          #returns TRUE if age is a vector of the specified mode having no attributes other than names. It returns FALSE otherwise.

    #*EXERCISE 2: How many students are in the sample? = sample size | VECTOR LENGTH
length(performance) #returns the number of elements stored in vector performance
                                ###computes the length/dimension of vector performance = sample size = number of units | returns the value 10
noStudents=length(performance) #optionally, we can store the number of students into a new variable noStudents

    #*EXERCISE 3: What is the minimum and the maximum grade obtained by the 10 students?
minPerformance=min(performance) #finds the lowest value in the range and stores it in variable minPerformance
maxPerformance=max(performance) #finds the greatest value in the range and stores it in variable maxPerformance
        #alternatively:
performanceSorted=sort(performance)          #orders the values of performance in ascending order
#sort(performance, decreasing=TRUE) #to sort in decending order set decresing=TRUE | by default decreasing=FALSE
        ### EXERCISE POINT_5: I type <<sort(performance, descending=TRUE)>>. The software returns <<Error in sort.int(x, na.last = na.last, decreasing = decreasing, ...) : unused argument (descending = TRUE)>>.
                #What is the problem?
minPerformance=performanceSorted[1]     #returns the first element of my ordered vector
maxPerformance=performanceSorted[length(performance)]   #returns the last element of my ordered vector | alternatively, I can use <<maxPerformance=performanceSorted[10]>> since my last element is 10


#OPERATIONS WITH VECTORS
    #*EXERCISE 5: The students are granted extra-credit - 1 point each. What are the new grades?

#we should add the value 1 to each value in the performance vector
extraCredit=c(1,1,1,1,1,1,1,1,1,1)      #creates a vector extraCredit with 10 values of 1 (1 for each student)
        #alternatively:
extraCredit=rep(1,10)                   #creates a vector containg the value 1 for 10 times | rep(value, vector length/number of repetitions)
        #alternatively:
extraCredit=rep(1,length(extraCredit))  #creates a vector containg the value 1 for 10 times (number of students)

newPerformance=performance+extraCredit  #sums up element i in vector performance with element i in vector extracredit, i=1:10 and stores all the sums in vector newPerformance
newPerformance                          #prints values of newPerformance

#alternatively:
newPerformance=performance+1            #sums up element i in vector performance with 1, i=1:10 and stores all the sums in vector newPerformance

    #*EXERCISE 6: The students receive different extra credit: 1,1,1,1,1,5,1,1,2,0 points. What are the new grades?
extraCredit=c(1,1,1,1,1,5,1,1,2,0)      #stores the extra credit values into extraCredit vector
newPerformance=performance+extraCredit  #sums up element i in vector performance with element i in vector extracredit, i=1:10 and stores all the sums in vector newPerformance
newPerformance                          #prints values of newPerformance

        ### EXERCISE POINT_7: What is the differece between <<newPerformance=performance+extraCredit>> above and <<newPerformance=c(performance,extraCredit)>>?

        ### EXERCISE POINT_8:Comment the code below (TIP: run the code line by line and not altogether)
performance=c(10, 9, 7, 8, 2, 10, 8, 3, 6, 9)
performance
```

```r
View(performance)
extraCredit=c(1,1,1,1,1,5,1,1,2,0)
newPerformance=performance+extraCredit

gradeBook_1=cbind(performance,extraCredit)
gradeBook_1
View(gradeBook_1)
nrow(gradeBook_1)
ncol(gradeBook_1)
dim(gradeBook_1)
rownames(gradeBook_1)
colnames(gradeBook_1)

gradeBook_2=rbind(performance,extraCredit)
View(gradeBook_2)
nrow(gradeBook_2)
ncol(gradeBook_2)
dim(gradeBook_2)
rownames(gradeBook_2)[2]
colnames(gradeBook_2)

gradeBook_3=cbind(performance, extraCredit, newPerformance)
View(gradeBook_3)
dim(gradeBook_3)
gradeBook_3[,1]
gradeBook_3[,3]
gradeBook_3[6,]
gradeBook_3[1:5,2]
length(dim(gradeBook_3))
dim(gradeBook_3)[2]

passed=c("passed", "passed", "passed", "passed", "failed", "passed", "passed",
"failed", "passed", "passed")
gradeBook_3=cbind(gradeBook_3,passed)
dim(gradeBook_3)
gradeBook_3[,4]

write.csv(gradeBook_3,"E:/Work/Multicultural Business Institute/R/Grade Book.csv")
#change with your own path to be able to run it
#####!!!when writing the location/path use "/". Ubuntu and Mac users: copies the path
with "/". Windows users: copies the path with "\" - change it in R.
gradeBook=read.csv("E:/Work/Multicultural Business Institute/R/Grade Book.csv")

colnames(gradeBook)
gradeBook$passed
View(gradeBook$passed)

```