R you ready for R? Let's get started!

```
print("Hello world!")
```

Prac:1 Expression vs Assignment

Expressions are evaluated, results are printed. Values are not stored.

Expression:

```
2+5;
```

An assignment statement is evaluated and value is stored. Semicolon ';' is optional to end a statement. Hash, is used to comment single line.

```
x=2;
```

Prac2 Explore inbuilt functions of R

Functions are inbuilt R codes which perform specific tasks.

Syntax: Function Name followed by parenthesis (). We can pass values/arguments to the function inside parenthesis.

```
getwd();
setwd("c:/Users/sillu/Desktop/");
getwd();
```

- getwd(): returns working directory
- setwd(): change the working directory by passing path of new directory

```
print("hii");
print(x);
ls();
rm(x);
print(x);
```

- ls():Lists all objects/variables created
- rm():Delete provided object
- print(x):print value of existing variable x

DATA TYPES AND STRUCTURES

VECTOR

Prac3 Creating a numeric Vector

```
x1=c(1,100,202,301)
x2=200:250
x3=250:200
x4=c(x1,x2)
x5=seq(from=0, to=1, by=0.01)
x6=seq(from=0, to=1, length=10)
x7=rep(c(1,5),times=10)
x8=rep(c(1,5),each=10)
x9=sample(1:50,10)
x10=sample(20:50,100, replace=TRUE)
x11=rnorm(100)
x12=rnorm(100,mean=5,sd=2)
```

Tasks: 1 to 8

Prac4: Fetching elements from a vector

```
x=c(10,20,30,40,50,60)
length(x)
x[1]
x[4]
x[2:3]
x[c(1,3,5)]
x[3:length(x)]
x[-1]
x[-c(1,3)]
x[10] #NA
x[1,3,5] #Error
```

Tasks: 9 to 12

Prac5: Vector manipulation

Delete element(s) from existing vector

```
x=10:20
x=x[-3]
x=x[-c(7,3)]
x=x[-c(2,length(x))]
```

Add element(s) to existing vector

```
x=10:20
x=c(x,55)
x=c(33,x,77)
y=seq(100,110,0.5)
x=c(x,y)
```

Replace value(s) of element(s) in existing vector

```
x=sample(20:30,5)
x[2]=15
x[c(1,3)]=c(8,9)
```

Task: 13

Prac6: Inbuilt functions for numeric vector

```
x=seq(4,8,length=10)
length(x)
sort(x)
order(x)
max(x)
min(x)
range(x)
mean(x)
median(x)
quantile(x,0.25)
summary(x)
sd(x)
var(x)
sin(x)
cos(x)
log(x,base=2)
log(x,base=10)
```

For pairs of vectors

```
x=10:30
y=seq(3,6,length=21)
cor(x,y)
```

Set operations

```
union(x,y)
intersect(x,y)
setdiff(x,y)
```

Prac7: Arithmetic expressions for vector

Vector recycling

```
x=c(3,4,5,6)
y=c(6,7,8,9)
z=c(1,2)
p=c(9,10,11)
x+y
x+z
x+p
```

Operators

```
x=c(1,2,3)
y=c(6,7,8)
z=c(10,11,12)
z
x-y
x*4
x*y
y/5
y%%5
y^3
```

Operator precedence

```
x+2*y+z
x+2*y/z
(x+2)*y+z
(x+2)*(y+z)
n=10
1:n-1
1:(n-1)
```

Tasks: 14 and 15

Prac8: Conditions for numeric vector

```
x=10:30
x>15
x==15
x>15 & x%%2==0
x[x>15 & x%%2==0]
x>15 | x%%2==0
x[x>15 | x%%2==0]
tempind=which(x>15 & x%%2==0)
tempind
x[tempind]
```

Task: 16

Prac9: Data types of vector

```
x1=1:10
str(x1)
x2=c(1.2,2.3,-4.5,5.6)
str(x2)
x3=c("aaa","bbb")
str(x3)
x4=c('ccc','ddd')
str(x4)
x5=c(TRUE,FALSE)
str(x5)
x6=letters
str(x6)
```

Prac10: Implicit Data type conversion

```
x1=c(1,"abc",TRUE)
str(x1)
x2=c(1,TRUE)
str(x2)
```

Prac11: Character vector and related functions

```
x=c("gene1","gene2","gene3")
y=c('prot','rna','lipid')
z=letters
nchar(x)
nchar(y)

substr("abcdef", 2, 4)
substr(c(x,y),2,3)
substr(x,2,3)="**"

paste(letters,1:5)
paste(letters,1:5,sep=":")
paste(letters,1:5,sep=":",collapse = ",")
```

Prac13: Explicit Data Type Conversion

as.numeric()

```
x=c("1","2","3","4")
str(x)
sum(x) # Error
y=as.numeric(x)
str(y)
sum(y)
```

as.character()

```
x=10:20
str(x)
y=as.character(x)
str(y)
```

MATRIX

Prac14: Create a matrix

```
x=21:70
mcx=matrix(x,ncol=10)
mrx=matrix(x,ncol=10,byrow=TRUE)
```

Prac15: Functions related to matrix

```
str(mcx)
dim(mcx)
nrow(mcx)
ncol(mcx)
colMeans(mcx)
rowMeans(mcx)

mtx=t(mcx)
str(mtx)

colnames(mcx)
rownames(mcx)
```

Set names to rows and columns of a matrix

```
colnames(mcx)=c(paste("C",1:ncol(mcx),sep=""))
rownames(mcx)=c(paste("R",1:nrow(mcx),sep=""))
```

Prac16: Fetch elements from a matrix using indices

```
mcx[1,3]
mcx[3,5]
mcx[c(1,2),c(3,4)]
mcx[1,]
mcx[,3]
mcx[2,]
mcx[3]
```

Prac17: Fetch values using index matrix

```
index=c(2,3,3,4,4,5)
im=matrix(index,nrow=3,byrow=T)
mcx[im]
```

Prac18: Fetch elements from a matrix using row and column names

```
rownames(mcx)
colnames(mcx)
mcx["R1","C3"]
```

Tasks: 17 to 21

Prac19: Matrix Manipulation

Insert rows and columns by rbind() and cbind()

```
mnew=matrix(1:10,nrow=5)
mnew=rbind(mnew,999)
mnew=rbind(mnew,c(2,4))# add row
mnew=cbind(mnew,1:5) # add column
mnew
```

Create matrix using rbind() and cbind()

```
x=seq(2,4,length=10)
y=1:10
mcb=cbind(x,y)
mrb=rbind(x,y)
```

Delete row(s) and column(s)

```
mcx
mcx=mcx[-3,]
mcx=mcx[,-2]
mcx
```

DATA FRAME

Prac20: Create a data frame

```
x=1:26
z=letters
y=paste(z,x,sep=":")
d=data.frame(C1=x,C2=z,C3=y)
str(d)
```

Prac21: Functions related to data frame

```
colnames(d)
rownames(d)
dim(d)
edit(d)
```

Prac22: Fetch values from a data frame

```
d[,1]
d$C1
d[3,]
d[3,1]
d$C1[3]
```

Prac23: Insert column to data frame

```
d
d$C5=1
d$C6=1:100 #Error
d$C6=c(seq(3,8,length=26))
```

Prac24: Delete column or row from data frame

```
d
d$C5=NULL
d
```

Tasks: 22 to 28

Check the data structure of R object

```
is.matrix(m)
is.vector(1:100)
is.data.frame(d)
is.factor(f)
```

CONTROL STRUCTURE: FOR LOOP

Prac25: Create a *for loop* to print the numbers from 1 to 50.

```
for(i in 1:50){
  print(i)
}
```

Prac26: Print values and indices in a vector using for loop.

```
x <- c(2,3,4,8,9)
for(i in x){
  print(i)
}
```

Print indices of vector x

```
for(i in 1:length(x)){
  print(i)
}
```

Print values in vector x using indices

```
for(i in 1:length(x)){
  print(x[i])
}
```

Prac27: Store squares of values of above vector in separate vector s using for loop

```
s <- NULL
for(i in 1:length(x)){
  s[i] <- x[i]^2
}
s
```

Note: Usually we don't need to use for loop to perform operations on a single vector. The vector s created using above for loop can simply be created using command, `s<- x^2.`

But for loop is very useful to perform operations on multiple columns and rows of a matrix or data frame.

Prac28: Create a matrix of 5 rows for values 1 to 50. Calculate the means of all rows in a matrix and store them in a vector m.

```
mat<- matrix(1:50,nrow=5)
mat
m <- NULL
for(i in 1:nrow(mat)){
  m[i]<- mean(mat[i,])
}
m
```

FUNCTION

Prac29: Define a function to find maximum values from all columns of a matrix.

Create function

```
getcolmax<-function(m){
  mx<-NULL
  for(i in 1:ncol(m)){
    mx[i]<- max(m[,i])
  }
  return(mx)
}
```

Call a function

```
colmax<-getcolmax(mat)
colmax
```

FILE IMPORT: Reading data from a text file

Prac30: Import files separated by one or more space(s) OR tab using read.table() by passing just file name.

```
in1 <- read.table("Input_1.txt")
in1
str(in1)
colnames(in1)
rownames(in1)
str(in1$V1)
str(in1$V5)
```

- Please note that in file Input_1.txt all rows have equal numbers of columns.
- See help for read.table() to check the default values for arguments.
- Default: Header=FALSE, sep=" ", stringsAsFactors=T
- Note the structure of in1 and its columns V1 and V2. **They are all factors. Due to inter-data type conversion.**
- Columns are named as V1 to V5.
- First row consists of column names

Prac31: Try different arguments to read data in expected format.

```
in2<-read.table("Input_1.txt",header=T)
in2
str(in2)
colnames(in2)
rownames(in2)
str(in2$Sepal.Length)
str(in2$Species)
```

- Header=T allows to read first row in a file as column names vector
- Note the structure of in2. The first four columns are numeric as expected
- But Species column has been considered as factors.

```
in3<-read.table("Input_1.txt",header=T, stringsAsFactors=F)
in3
str(in3)
colnames(in3)
rownames(in3)
str(in3$Sepal.Length)
str(in3$Species)
```

- stringsAsFactors=F disables factor formatting of character columns.
- Note the structure of in3. The first four columns are numeric. Now Species column has been considered as a character vector.
- Thus, data is imported as per expectation

Prac32: Read file with unequal columns in first row using read.table.

Please note that in file Input_2.txt, first row has 5 column fields while remaining rows have 6 fields/values i.e. first row has one column less than other rows.
Under such format Header is automatically set to TRUE by read.table()

```
in4 <- read.table("Input_2.txt",stringsAsFactors=F)
in4
str(in4)
colnames(in4)
rownames(in4)
str(in4$V1)
str(in4$Species)
```

- Note that header row is printed with column names.
- in4$V1 is printed NULL, because column with name V1 does not exist.

Prac33: Read file consisting of comments, blank lines, null values

Default arguments: skip=0, comment.char="#", na.strings="NA"
Please note that in file Input_3.txt consists of

- ✓ First two lines inserted by author but are not required for processing data.
- ✓ 2 comment lines starting with character "!"
- ✓ 1 blank line
- ✓ Row 3 has one missing value shown by NULL
- ✓ First row has 5 columns while remaining rows have 6 columns. Under such input format Header is set to TRUE by read.table()

```
in5 <- read.table("Input_3.txt",
    stringsAsFactors=F, comment.char="!",
    na.strings="NULL", skip=2)
in5
str(in5)
colnames(in5)
rownames(in5)
str(in5$Species)
in6 <- na.omit(in5)
in6
```

- skip=2 removes first two lines not required for processing data by R.
- comment.char="!", instructs R to exclude lines starting with "!". Default comment.char="#"
- na.strings= "NULL" allows R to interpret NULL values as missing or NA characters, which will be helpful to remove such rows by na.omit() function. Default na.strings= "NA"
- in5 consists of 3rd row with NA characters
- na.omit(in5) removes 3rd row
- in6 consists of all rows having no NA values.

Prac34: Read CSV file

Please note that values in file Input_4.txt are separated by comma, "," and all rows have equal number of columns.
See help for read.csv to check the default values for arguments. Default Header=TRUE, sep=","

```
in7 <- read.csv("Input_4.txt",
    stringsAsFactors=F, comment.char="!",
    na.strings="NULL", skip=2)
in7
str(in7)
colnames(in7)
rownames(in7)
str(in7$V1)
str(in7$Sepal.Length)
in8<-na.omit(in7)
in8
```

Please note that column names are printed, because header is by default set to TRUE.
Remaining arguments have same meaning as explained in Prac33.

Note: read.csv and read.table differs just in terms of their default settings.

FILE IMPORT: Reading data from excel file

Prac35: Read excel file

```
library(gdata)
xl<-read.xls("Input_3.xlsx",sheet=1)
xl
str(xl)
```

- library(gdata) loads the gdata package to access read.xls() function.
- sheet option allows to choose sheet from input excel file.

FILE IMPORT: Reading lines of a file

Prac36: Read file line by line.

```
ln <-readLines("1BUW.pdb")
ln
str(ln)
```

- By default readLines() read all the lines of a file
- It returns a character vector. Check the str(ln)

```
ln <-readLines("1BUW.pdb",n=10)
ln
str(ln)
```

- n=10 allows to read first 10 lines.
- sheet option allows to choose sheet from input excel file.

FILE IMPORT: Read copied text using clipboard

First copy the text from Input_1.txt

```
read.table("clipboard")
```

FILE EXPORT: Write R objects/results in a file

Prac37: Writing processed R data variables in a file

```
w<-read.table("Input_1.txt",header=T)
PL<-w[,3]
PW<-w[,4]
SP <-w[,5]
report<-cbind(PL,PW,mean(PL),mean(PW))
report
write.csv(report,"Report.txt",quote=FALSE)
```

Similarly, we can write as many R variables in a file.