R Notebook

Code

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

Hide

```
# R Programming Assignment - 1
# MDSC-201 LAB
# Saideva Sathvik.R
```

Hide

```
# First Program

string <- "Hello World!!"
print(string)
```

```
[1] "Hello World!!"
```

Hide

```
# Data Types In R:
#   R has five basic or “atomic” classes of objects:
#    • character
#    • numeric (real numbers)
#    • integer
#    • complex
#    • logical (True/False)

# Numbers
# Numbers in R are generally treated as numeric objects (i.e. double precision real numbers).

number1 <- 1
```

Hide

```
# If you explicitly want an integer, you need to specify the L suffix. So entering 1 in R giv
es you a
# numeric object; entering 1L explicitly gives you an integer object.

number1.int <- 1L
print(number1.int)
```

```
[1] 1
```

Hide

```
print(typeof(number1.int))
```

```
[1] "integer"
```

Hide

```
# Logical:
# Logical variables in R Either Take TRUE/FALSE Only.

logic_variable = TRUE
print(logic_variable)
```

```
[1] TRUE
```

Hide

```
# Character:
# Characters in R are same as in other programming languases. They take A-Z or a-z.

character_variable <- 'R'
print(character_variable)
```

```
[1] "R"
```

Hide

```
# Complex:
# Complex Numbers Can be expressed in R and can be worked upon easily.

complex_variable.1 <- 2+3i
complex_variable.2 <- 3+5i

sum <- complex_variable.1 + complex_variable.2

print(sum)
```

```
[1] 5+8i
```

Hide

```
print(typeof(sum))
```

```
[1] "complex"
```

Hide

```
# R Objects
# VECTOR
# The most basic type of R object is a vector. Empty vectors can be created with the vector()
function.

vector.example <- c(1,2,3,4)
```

Hide

```
print(vector.example)
```

```
[1] 1 2 3 4
```

Hide

```
# Lists:
# Lists are a special type of vector that can contain elements of different classes. Lists ar
e a very
# important data type in R.
# Lists can be explicitly created using the list() function, which takes an arbitrary number
of
# arguments.

x <- list(22234, "Saideva Sathvik", "B+", "I MSc.")
x
```

```
[[1]]
[1] 22234

[[2]]
[1] "Saideva Sathvik"

[[3]]
[1] "B+"

[[4]]
[1] "I MSc."
```

Hide

```
# Matrices:

# Matrices are vectors with a dimension attribute. The dimension attribute is itself an integ
er vector
# of length 2 (number of rows, number of columns)

m <- matrix(nrow = 2, ncol = 3)
print(m)
```

```
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
```

Hide

```
# Matrices are constructed column-wise, so entries can be thought of starting in the “upper l
eft” corner
# and running down the column.

m <- matrix(1:6, nrow = 2, ncol = 3)
print(m)
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Hide

```
# Matrices can also be created directly from vectors by adding a dimension attribute.

m <- 1:10
print("Vector:")
```

```
[1] "Vector:"
```

Hide

```
m
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

Hide

```
dim(m) <- c(5, 2)
print("After converting vector into matrix")
```

```
[1] "After converting vector into matrix"
```

Hide

```
# Factors:
# Factors are used to represent categorical data and can be unordered or ordered. One can thi
nk of
# a factor as an integer vector where each integer has a label.
# Factor objects can be created with the factor() function.

x <- factor(c("India", "India", "Italy", "America", "America", "England"))
x
```

```
[1] India   India   Italy   America America England
Levels: America England India Italy
```

Hide

```
table(x)
```

```
x
America England   India   Italy
      2       1       2       1
```

Hide

```
# Data Frames:
# Data frames are used to store tabular data in R. They are an important type of object in R
and are
# used in a variety of statistical modeling applications.

# Data frames are represented as a special type of list where every element of the list has t
o have the
# same length. Each element of the list can be thought of as a column and the length of each
element
# of the list is the number of rows.

x <- data.frame(foo = 1:4, bar = c(T, T, F, F))
x
```

foo <int>	bar <lgl>
1	TRUE
2	TRUE
3	FALSE
4	FALSE

4 rows

Hide

```
NA
```

Hide

```
# Names:
# R objects can have names, which is very useful for writing readable code and self-describin
g objects.
# Here is an example of assigning names to an integer vector.

x <- 1:3
print("names before assigning: ")
```

```
[1] "names before assigning: "
```

Hide

```
names(x)
```

```
NULL
```

Hide

```
names(x) <- c("New York", "Seattle", "Los Angeles")


print("After Assigning: ")
```

```
[1] "After Assigning: "
```

Hide

```
names(x)
```

```
[1] "New York"    "Seattle"     "Los Angeles"
```

Hide

```
# Arrays are the R data objects which can store data in more than two dimensions. Arrays can
store only data type.
# An array is created using the array() function. It takes vectors as input and uses the valu
es in the dim parameter to create an array.

# Create two vectors of different lengths.
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)

# Take these vectors as input to the array.
result <- array(c(vector1,vector2),dim = c(3,3,2))
print(result)
```

```
, , 1

     [,1] [,2] [,3]
[1,]    5   10   13
[2,]    9   11   14
[3,]    3   12   15

, , 2

     [,1] [,2] [,3]
[1,]    5   10   13
[2,]    9   11   14
[3,]    3   12   15
```

Hide

```
column.names <- c("COL1","COL2","COL3")
row.names <- c("ROW1","ROW2","ROW3")
matrix.names <- c("Matrix1","Matrix2")

# Take these vectors as input to the array.
result <- array(c(vector1,vector2),dim = c(3,3,2),dimnames = list(row.names,
   column.names, matrix.names))

# Print the third row of the second matrix of the array.
print(result[3,,2])
```

```
COL1 COL2 COL3
   3   12   15
```

Hide

```
# Print the element in the 1st row and 3rd column of the 1st matrix.
print(result[1,3,1])
```

```
[1] 13
```

Hide

```
# Print the 2nd Matrix.
print(result[,,2])
```

```
     COL1 COL2 COL3
ROW1    5   10   13
ROW2    9   11   14
ROW3    3   12   15
```

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.