```
# Lecturer Notes- Jan 18 and Jan 19

# anything after # will be ignored by R, but only this line.
# another comment

#========================Data Types=======================
# Basic Data types (numeric, integer, complex, character, logical)

#numeric
  x<- 23.768

#integer
  y<- 34L

#Character/String
  z<- "R is wonderful"
  "HELLO!"
  'YES'

#Logical/Boolean
  h<- TRUE
  h
  g<- FALSE

  "TRUE"

# Variable
  a<-3*9*7*6
  age <- 30
  age+4
  a<- 30
  green<- 5

#========================Operators====================
# operators(Addition, Subtraction, Multiplication, Division,
# Exponent, Remainder from division, Integer division)

  5+5
  5/6
  5^2
  5%%2
  15%/%2

#======================Functions=====================

  sum(4,8,7)

  mean(3,9,6)

  prod(3,9,7,6)

  print(a)
  a

# note you can see the results with using function print or without function print

  s<- "tree"
  class(s)
  n<- TRUE
  class(n)
  a<- 34
  class(a)
  a<- 34L
```

```
  class(a)

# Example of mean function
  a<- c(2,4,7)
  mean(a)

#============================Data Structures====================
# Data structures in R: Vector, Dataframe, Matrices, List, Array, Factors

# A vector is simply a list of items that are of the same type.

  h<- c(1,2,9)
  h

  v<- c(2,4,5,6,8,  9)
  v[3]
  v[4]

  # k<- c(from=5: to =9)
  K<- c(5:9)
  K

  length(K)# length is a function to see the length of a vector

  G<- seq(5,9)# we can create a vector by using the seq function
  G

  U<- seq(4,18,0.5)
  U

  Days<- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")

  Days[3]
  names(Days)<- c(0,1,2,3,4,5,6)
  Days
  #Classes<-c(0, 1, 2, 1, 3, 1, 0)
  #sum(Classes)
  #names(Days)<- Classes # names is a function to set the name of an object

  schooldays<- Days[2:5]

  print(schooldays)
  sum(schooldays)
  sum
# create random sample

  sum(Days) # Be careful to use sum, mean or prod functions on numerical
  a<- c(2,9, 5,7, 2, 6,7, 9)
  sample(a,2, FALSE)# sample() function is used to take a random sample of individuals
                  #from a dataset or vector, either with and without replacement

  sample(4:70, 5)
  sample(4:70, 5, replace=T)# with replacement trueだと同じ文字出てくる

  set.seed(10)# gives same sample #これでseed(10)にsampleが固定されたから同じ結果になる
  sample(4:70, 5)

  classes=c("STAT123", "STAT345", "STAT255", "STAT226")

  set.seed(125)
  sub_classes=sample(classes, 2)
  sub_classes
```

```
#============================Datasets=================================
  data()# To see the list of available datasets, use data() function

  ?airquality # ? shows the description and information in help environment
  class(airquality)#class() function shows the type as represented
  typeof(airquality)#typeof() function shows the type as stored in the memory
  ?typeof

  airquality
  head(airquality)# head function uses to display the first n rows present in the input
data frame
  nrow(airquality)#The number of rows
  ncol(airquality)#The number of columns

#============Example of making a random sample from a data frame=======
  set.seed(10)
  sample(1:nrow(airquality), 5)

#============================Dataframe=====================
  # Data Frames are data displayed in a format as a table.
  # You can create a dataframe by using data.frame() function
  data("trees")
  head(trees)
  nrow(trees)
#You can use single brackets [ ], double brackets [[ ]]
#or $ to access columns from a data frame
  trees$Height
  trees$Girth
  trees["Height"]
  trees[["Height"]]
  mean(trees$Height)
  mean(trees["Height"])


#make a sample
  set.seed(18)
  sample_rows <- sample(1:nrow(trees), 2)
  sample<- trees[sample_rows,]
  trees$Volume
  mean(trees$Volume)
  round(mean(trees$Volume),2)#round numbers in R: round(x, digits= the number of decimals)


  Data_farme<- data.frame(Subject= c("stat123", "stat233", "stat255", "Stat145"),
                          Section= c("A01", "B02", "C01", "D01"))
  Data_Frame

  Data_Frame_test<- data.frame(
    Col1=c(1,2,3,5,6),
    col2=c(6,8,9,0,9)
  )
  Data_Frame

#Use the rbind(): 新しいrowの追加
  Data_Frame <- data.frame (
    Training = c("Height", "Weight", "Other"),
    Pulse = c(100, 150, 120),
    Duration = c(60, 30, 45)
  )
  Data_Frame

  # Add a new row with rbind()
  New_row_DF <- rbind(Data_Frame, c("Strength", 110, 110))
```

```
  New_row_DF

# Use the cbind() function to add new columns in a Data Frame
  New_col_DF <- cbind(Data_Frame, Steps = c(1000, 6000, 2000))

  New_col_DF

#Use the c() function to remove rows and columns in a Data Frame
  Data_Frame <- data.frame (
    Training = c("Strength", "Stamina", "Other"),
    Pulse = c(100, 150, 120),
    Duration = c(60, 30, 45)
  )

  # Remove the first row and column using -c()
  Data_Frame_New <- Data_Frame[-c(1), -c(1)]

  Data_Frame_New

#Use the dim() function to find the amount of rows and columns in a Data Frame
  dim(Data_Frame_New)

# Use the length() function to find the number of columns in a Data Frame (similar to
ncol())
  length(Data_Frame_New)


#Use the summary() function to summarize the data from a Data Frame
  summary(Data_Frame)

#============================Matrices======================

#A matrix is a two dimensional data set with columns and rows.Using matrix()
  sample_matrix <- matrix(c(1,2,3,4,5,6,5,7,8,9,1,0,4,2,4), nrow = 3, ncol = 5)

#make string matrix
  sample2_matrix <- matrix(c("apple", "banana", "cherry", "cucumber"), nrow = 2, ncol = 2)
  sample2_matrix

#You can access the items by using [ ] brackets.
#The first number "1" in the bracket specifies the row-position,
#while the second number "2" specifies the column-position:
  sample_matrix
  sample_matrix[1,2]#You can access the items by using [ ] brackets
  sample_matrix[2,]
  sample_matrix[,2]

#More than one row can be accessed if you use the c() function
  sample_matrix[c(1,2),]

# The rownames()function helps you to change the name of rows
  sample2_matrix
  rownames(sample2_matrix)<-c("Name1", "Name2")
  sample2_matrix

  rownames(sample2_matrix)[1]<-c("Name0")
  sample2_matrix

#The colnames () function to change the name of coluumns
  sample2_matrix
  colnames(sample2_matrix)<-c("Name1", "Name2")
  sample2_matrix
```

```
#Use the c() function to remove rows and columns in a Matrix
  sample_matrix
  sample3_matrix <- sample_matrix[-c(1), -c(1)]
  sample3_matrix

#To find out if a specified item is present in a matrix, use the %in% operator
  sample2_matrix
  "banana" %in% sample2_matrix
  "dragon" %in% sample2_matrix

# Use the dim() function to find the number of rows and columns in a Matrix
  dim(sample2_matrix)

#Use the length() function to find the dimension of a Matrix
  length(sample2_matrix)

#==============================List=========================
#To create a list, use the list() function.
#A list in R can contain many different data types inside it.
#A list is a collection of data which is ordered and changeable.

  # List of strings
  samplelist<- list("apple", "banana", "cherry")

  samplelist

#You can access the list items by referring to its index number, inside brackets
  samplelist2[2]

#To change the value of a specific item, refer to the index number
  samplelist[2]<- "cucumber"
  samplelist

# To find out how many items a list has, use the length() function
  length(samplelist)

#To add an item to the end of the list, use the append() function
  samplelist
  append(samplelist, "Figs")

  samplelist<- append(samplelist, "Grapefruit")
  samplelist

#To find out if a specified item is present in a list, use the %in% operator
  "apple" %in% samplelist

#To remove list items
  sample2list <- samplelist[-1]

#specify a range of indexes by specifying where to start and where
#to end the range, by using the : operator:
  (samplelist)[2:4]

#You can use the c() function, which combines two elements together
  list1 <- list("a", "b", "c")
  list2 <- list(1,2,3)
  list3 <- c(list1,list2)

  list3

#Array can be have more than two dimensions
# An array with one dimension with values ranging from 1 to 24
  samplearray <- c(2:19)
  samplearray
```

```
# An array with more than one dimension
  multiarray <- array(c(2:19), dim = c(4, 3, 2))
  multiarray

#You can access the array elements by referring to the index position, using [] brackets.
# array[row position, column position, matrix level]
  multiarray[2, 3, 2]

#Use the length() function to find the dimension of an array
  length(multiarray)

#Factors are used to categorize data like (male and female)
#To create a factor, use the factor() function
  gender <- factor(c("Male", "Female", "Male", "Male", "Female", "Male", "Female",
"Female"))

  gender
#To only print the levels, use the levels() function
  levels(gender)

#Use the length() function to find out how many items there are in the factor
  length(gender)

#To access the items in a factor, refer to the index number, using [] brackets
  gender[5]

#To access the items in a factor, refer to the index number, using [] brackets
  gender[5]<-"Male"
  gender
```