R Functions

Functions in R

An R function is created by using the keyword function.

The basic syntax of an R function definition is as follows:

function_name <- function(arg_1, arg_2, …) { # Function body

}

Function Components: The different parts of a function are: • Function Name: This is the actual name of the function. It is stored in R environment as an object with this name.

• Arguments: An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional; that is, a function may contain no arguments. Also arguments can have default values.

• Function Body: The function body contains a collection of statements that defines what the function does.

• Return Value: The return value of a function is the last expression in the function body to be evaluated.

User Defined Function

These are used for customization by the user. Repeated set of instructions are put in function to be used again It makes programs readable and easy to understand

An R function is created by using the keyword function.

User define function Without argument

Here function is created without any arguments. It will have process inside it. This function may or may not return any value

```
display = function(){
  print("This is a function ....")
}
display
```

```
## function(){
##   print("This is a function ....")
## }
```

```
display() # calling the function
```

```
## [1] "This is a function ...."
```

User define function With argument

Function can work better when user can some arguments

For example in following code function is taking two arguments. Here all arguments are compulsory.

```
add = function(a,b){
  # all functions can handle array / matrix/a single value/vector
  return(a+b)
}
add(23,45)
```

```
## [1] 68
```

```
add(c(3,4),c(6,7))
```

```
## [1]  9 11
```

```
add(3+8i,7+4i)
```

```
## [1] 10+12i
```

With Default argument

Default argument means a default value is given to argument.

If user doesn't pass some values then default value of that argument is taken

Default arguments should be on rightmost side

All other arguments are compulsory

```
pow = function(n,p=2){
  return(n^p)
}
pow(23)
```

```
## [1] 529
```

```
pow(34,5)
```

```
## [1] 45435424
```

Calling a Function with Argument Values (by position and by name)

The arguments to a function call can be supplied in the same sequence as defined in the function or they can be supplied in a different sequence but assigned to the names of the arguments.

```
# Create a function with arguments.
new.function <- function(a,b,c) {
result <- a*b+c
print(result)
}


# Call the function by position of arguments.
new.function(5,3,11)
```

```
## [1] 26
```

```
# Call the function by names of the arguments.
new.function(a=11,b=5,c=3)
```

```
## [1] 58
```

Lazy Evaluation of Function

When functions are not checked for syntax errors like in compiled languages.

here syntax will give error when it is executed

R is interpreted language, so line by line code is checked and executed

In following code line 196, only one value is passed when 2 are expected, still doesn't give error because inside the function 2nd value is not used. So even if second value is not present it doesn't matter!

```
square = function(a,b){
  print(a^2)
}
square(2)
```

```
## [1] 4
```

```
square(2,56)
```

```
## [1] 4
```

Built in Functions

Simple examples of in-built functions are seq(), mean(), max(), sum(x)and paste(…) etc. They are directly called by user written programs. You can refer most widely used R functions on following link,

https://cran.r-project.org/doc/contrib/Short-refcard.pdf (https://cran.r-project.org/doc/contrib/Short-refcard.pdf)

```
# Create a sequence of numbers from 32 to 44.
print(seq(32,44))
```

```
##  [1] 32 33 34 35 36 37 38 39 40 41 42 43 44
```

```
# Find mean of numbers from 25 to 82.
print(mean(25:82))
```

```
## [1] 53.5
```

```
# Find sum of numbers frm 41 to 68.
print(sum(41:68))
```

```
## [1] 1526
```

```
x = c(45,-34,23,11.33)
abs(x)
```

```
## [1] 45.00 34.00 23.00 11.33
```

```
x = c(1,4,6)
log(x)
```

```
## [1] 0.000000 1.386294 1.791759
```

```
x = c(1,4,6)
exp(x) # base is e .. e^no
```

```
## [1]   2.718282  54.598150 403.428793
```

```
x = c(1,4,6)
sqrt(x)
```

```
## [1] 1.00000 2.00000 2.44949
```

```
x = c(1,4,6)
factorial(x)
```

```
## [1]   1  24 720
```

```
x = c(1,4,6)
sin(x)
```

```
## [1]  0.8414710 -0.7568025 -0.2794155
```

```
x = c(1,4,6)
cos(x)
```

```
## [1]  0.5403023 -0.6536436  0.9601703
```

```
x = c(1,0,-5,10,-34,0,0,100,-90)
sign(x)
```

```
## [1]  1  0 -1  1 -1  0  0  1 -1
```

```
x=c(10,-20,-30,80,-90,40,20.1,0)
sign(x)
```

```
## [1]  1 -1 -1  1 -1  1  1  0
```

Match Function match returns a vector of the positions of (first) matches of its first argument in its second.

%in% internally uses match

```
x=c(10,20,30,10,20)
y=c(20,20,40)
match(x,y)
```

```
## [1] NA  1 NA NA  1
```

```
match(y,x)
```

```
## [1]  2  2 NA
```

```
ux <- unique(x)
ux
```

```
## [1] 10 20 30
```

```
match(x, ux)
```

```
## [1] 1 2 3 1 2
```

```
x=c(10:30)
y=c(10,20,60)
x
```

```
##  [1] 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

```
y
```

```
## [1] 10 20 60
```

```
match(y,x)
```

```
## [1]  1 11 NA
```

```
x=c(10,20,30,40,50)
y=c(20,4,60,50,30)
x
```

```
## [1] 10 20 30 40 50
```

```
y
```

```
## [1] 20  4 60 50 30
```

```
match(x,y)
```

```
## [1] NA  1  5 NA  4
```

Tabulate Function

tabulate() function takes the integer-valued vector bin and counts the number of times each integer occurs in it

```
tabulate(c(-5,0,3,5,4,5,4,10))
```

```
##  [1] 0 0 1 2 2 0 0 0 0 1
```

```
tabulate(c(3,5,4,5,4,100),nbins=4)
```

```
## [1] 0 0 1 2
```

```
tabulate(c(3,5,4,8),nbins=4)
```

```
## [1] 0 0 1 1
```

```
tabulate(c(3,5,4,8),nbins=10)
```

```
##  [1] 0 0 1 1 1 0 0 1 0 0
```

Example usage of match and tabulate to find frequency of values in given vector

```
# given a vector x
x=c(10,300,30,10,300)
x
```

```
## [1]  10 300  30  10 300
```

```
#unique values in x
ux <- unique(x)
ux
```

```
## [1]  10 300  30
```

```
# matched indexes for unique values in x
match(x, ux)
```

```
## [1] 1 2 3 1 2
```

```
#element wise fequency of x
tabulate(match(x, ux))
```

```
## [1] 2 2 1
```

NOTE ABOUT OPERATORS

Every operator is a function in R programming

```
'+'(90,45)
```

```
## [1] 135
```

```
'*'(34,56)
```

```
## [1] 1904
```