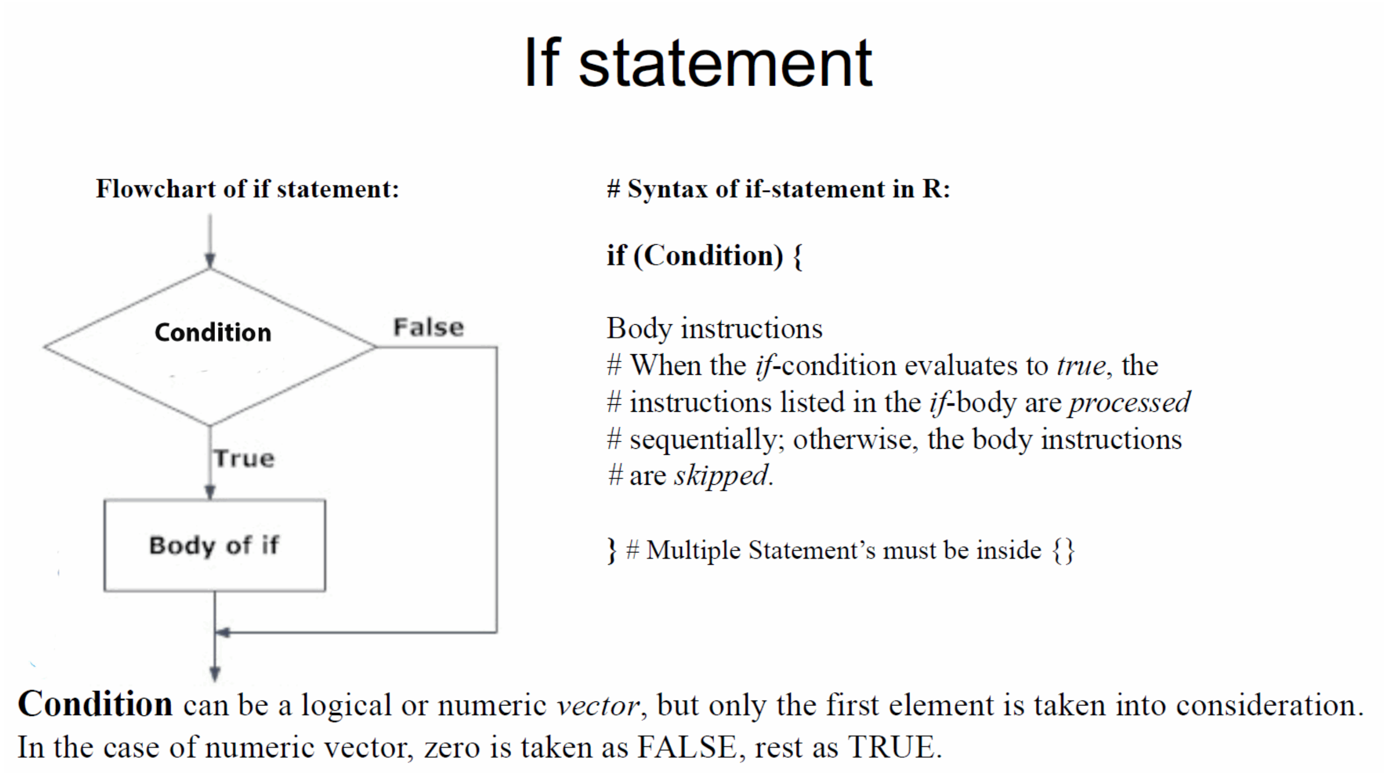
Code

R Notebook

R If … Else

Conditions and If Statements



If statement

# Syntax of if-statement in R:

if (Condition) {

Body instructions
When the *if*-condition evaluates to *true*, the
# instructions listed in the *if*-body are *processed*
# sequentially; otherwise, the body instructions
# are *skipped*.

} # Multiple Statement's must be inside {}

Condition can be a logical or numeric *vector*, but only the first element is taken into consideration. In the case of numeric vector, zero is taken as FALSE, rest as TRUE.

Hide

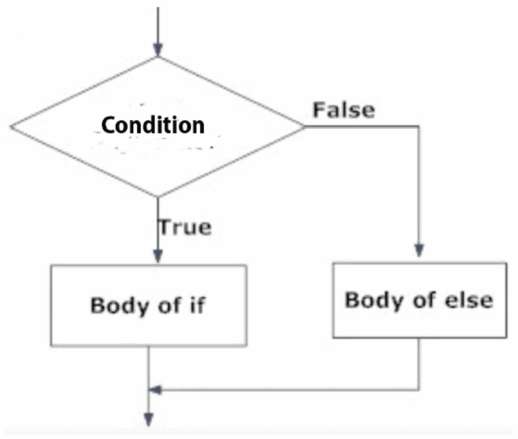
```
x = readline(prompt="Enter an integer number: ")



if(x>3) print("Positive")
if(x<7) print("negative")
```

Write the R script that take two integers as input and print out the larger number.

Hide

```
A = as.numeric(readline(prompt="Enter first numbera: "))
B = as.numeric(readline(prompt="Enter second number: "))
if(A > B) print(A)
if(B > A) print(B)
```

If…else statement



if (Condition) {

if-Body instructions

} else { # It is important to note that *else* must be in the same
line as the closing braces of the *if* statement.

else-Body instructions

}

Hide

```
x = as.integer(readline(prompt="Enter an integer number: "))
if(x>0) print("Positive") else print("Negative")
```

You can also run a condition in an `if` statement, which you will learn much more about in the if..else (https://www.w3schools.com/r/r_if_else.asp) chapter.

Hide

```
a = 200
b = 33

if (b > a) {
  print ("b is greater than a")
} else {
  print("b is not greater than a")
}
```

Hide

```
if (b > a) {
  (b*4)
} else {
  (b*2)
}
```

- Write the R script to:
  - Take an integer input
  - Take an integer threshold
  - Specify whether the input is larger, smaller or equal to the threshold?

Hide

```
A = as.integer(readline(prompt="Enter an integer number: "))
B = as.integer(readline(prompt="Enter an integer threshold: "))
if(A > B){
print(paste(A, "is greater than threshold ", B), quote=F)
} else if (A < B){
print(paste(A, "is less than threshold ", B), quote=F)
} else {
print (paste(A, "is equal to threshold ", B), quote=F)
}
```

Hide

```
a <- 200
b <- 33

if (b > a) {
  print("b is greater than a")
} else if (a == b) {
  print("a and b are equal")
} else {
  print("a is greater than b")
}
```

Nested If Statements

Hide

```
x <- 44

if (x > 10) {
  print("Above ten")
  if (x > 20) {
    print("and also above 20!")
  } else {
    print("but not above 20.")
  }
} else {
  print("below 10.")
}
```

AND

The & symbol (and) is a logical operator, and is used to combine conditional statements:

Hide

```
a <- 200
b <- 33
c <- 500

if (a > b & c > a){
  print("Both conditions are true")
}
```

OR

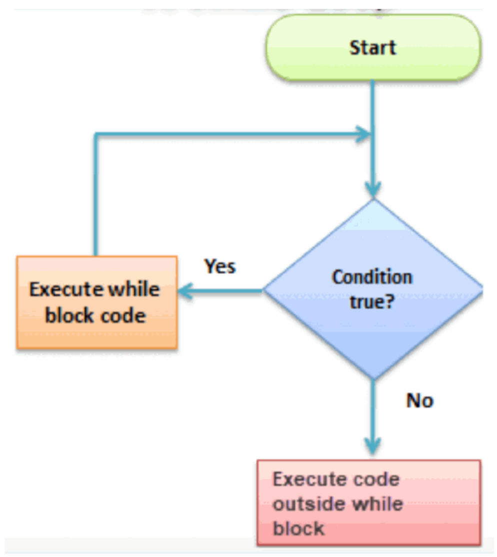
The `|` symbol (or) is a logical operator, and is used to combine conditional statements:

Hide

```
a <- 200
b <- 33
c <- 500

if (a > b | a > c){
  print("At least one of the conditions is true")
}
```

R While Loop

while loop



# Syntax of while loop in R:

while (Condition) {

Body instructions
# While loop start with the *condition*, and if the *condition* is
# True then instructions inside the while loop will be
# executed sequentially and keeps running
# until the *condition* is satisfied (TRUE)
# Remember to write a closing condition at some point
# otherwise the loop will go on indefinitely.
}

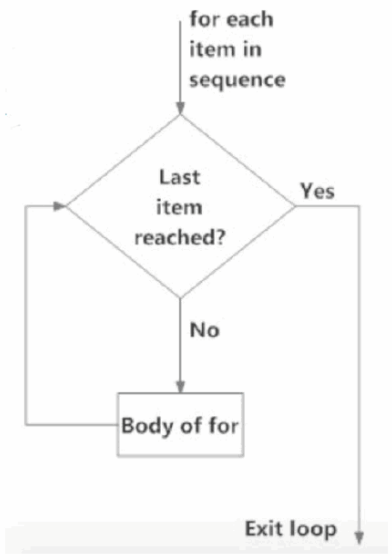
If the specified expression is false, it won't be executed at least once.

Hide

```
i = 1
while (i < 6) {
print(i)
i = i+1
}
```

R For Loop

for loop



for (val in sequence)
{# Here, sequence is a vector and val takes on each of its value during the loop.
statement # In each iteration, statement is evaluated.
}

In the below example, the loop iterates 7 times as the vector x has 7 elements.
In each iteration, val takes on the value of corresponding element of x.
We have used a counter to count the number of even numbers in x. We can see that x contains 3 even numbers.

x <- c(2,5,3,9,8,11,6)
count <- 0
for (val in x) {
if(val %% 2 == 0) count = count+1
}
print(count)

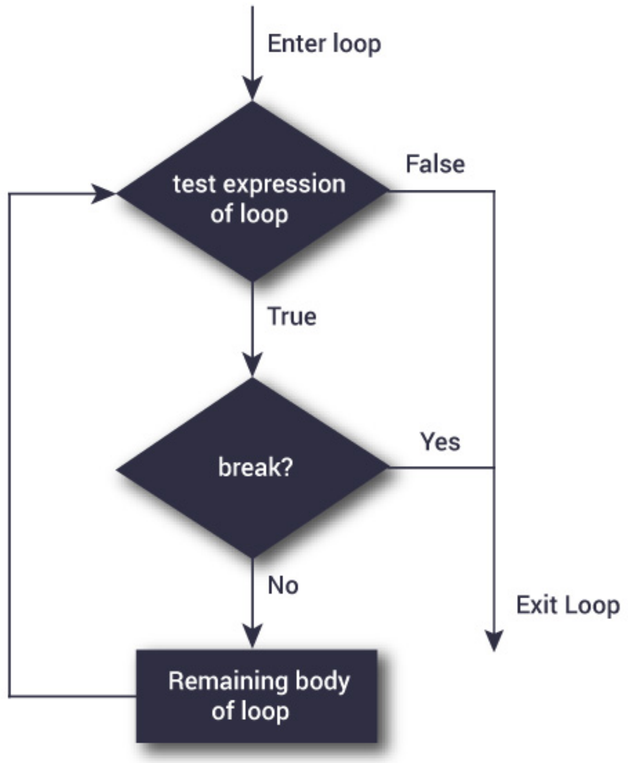
Hide

```
for (val in sequence)
{
statement
}
```

Hide

```
x = c(2,5,3,9,8,11,6,11)
count = 0
for (val in x) {
if(val %% 2 == 0)
count = count+1
}
print(count)
```

break statement



Hide

```
x <- 1:5
for (val in x) {
if (val == 3){
break
}
print(val)
}
```

Function

Hide

```
my_function <- function() {
  print("Hello World!")
}
my_function() # call the function named my_function
```

Hide

```r
novin = function(x,y) {
  paste(x,"hasani",y, "bagheri")
}







novin("zahra","hoseen")
```

Hide

```r
bmi = function(ghad,vazn) {
  x=vazn/(ghad^2)
  print(paste(x,"your bmi"))
}





bmi(1.60,65)
```

Hide

```r
my_function = function(fname, lname) {
  paste(fname, lname)
}

my_function("Peter", "Griffin")
```

Default Parameter Value

The following example shows how to use a default parameter value.

If we call the function without an argument, it uses the default value:

Hide

```r
my_function <- function(country = "Norway") {
  paste("I am from", country)
}

my_function("Sweden")
my_function("India")
my_function() # will get the default value, which is Norway
my_function("USA")
```

Return Values

To let a function return a result, use the `return()` function:

Hide

```
my_function <- function(x) {
  return (5 * x)
}

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

Hide

```
v=c(5,6,8,9,14)
rescale <- function(v) { # Rescales a vector, v, to lie in the range 0 to 1.
L <- min(v)
H <- max(v)
result <- (v - L) / (H - L)
return(result)
}
rescale(v)
```

END(1-2)