Control Structures In R

Pavan Kumar A
Senior Project Engineer
Big Data Analytics Team
CDAC-KP

Control Structures

- These allow you to control the flow of execution of a script typically inside of a function. Common ones include:
  - if, else
  - for
  - While
  - break
  - Next

equal: ==
not equal: !=
greater/less than: > <
greater/less than or equal: >= <=
and: &
or: |
not: !

Control Structures

- Decision making is an important part of programming.
- This can be achieved in R programming using the conditional if...else statement.

- **If statement**
- Syntax of if statement

```
if (test_expression) {
        statement
}
```

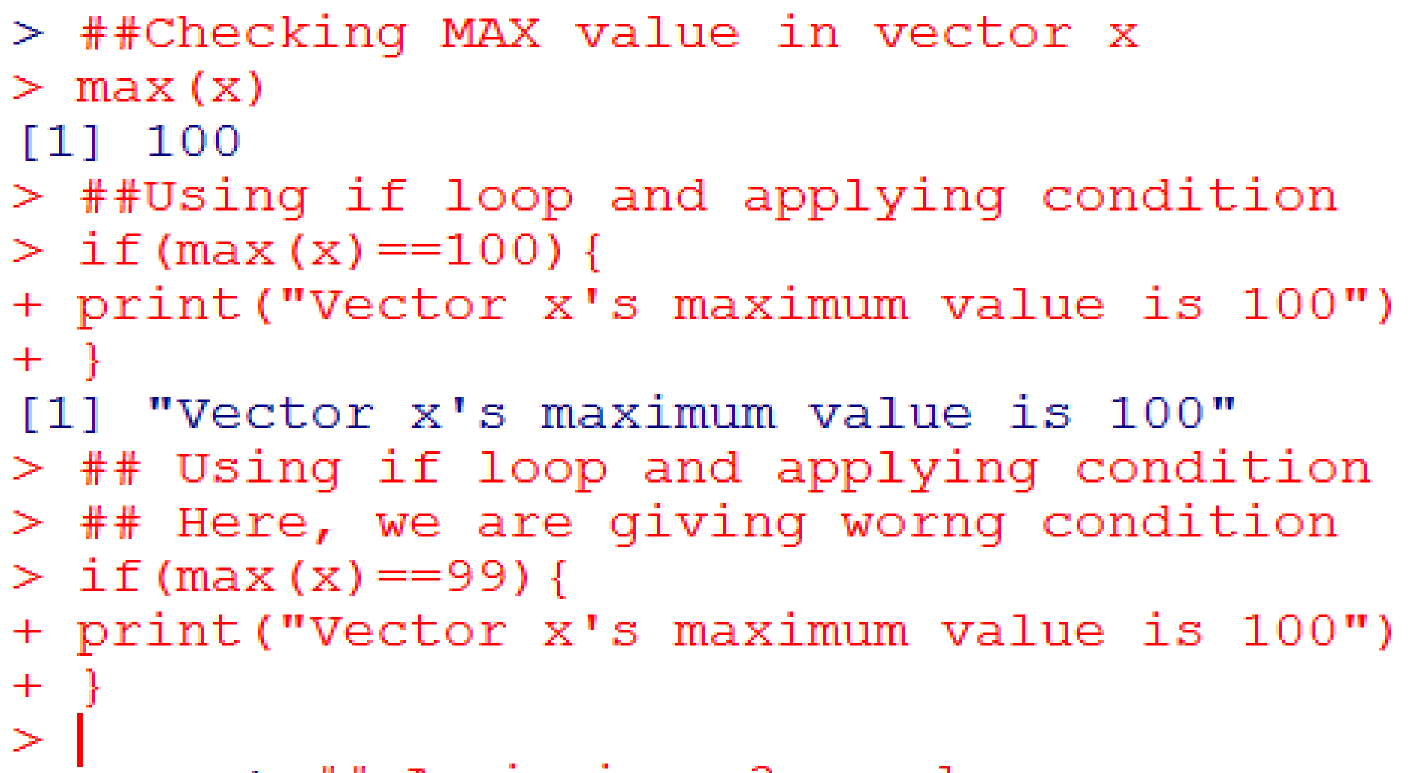
- If the `test_expression` is TRUE, the statement gets executed. But if it's FALSE, nothing happens.

IF STATEMENT - EXAMPLES

- **Example of if statement**

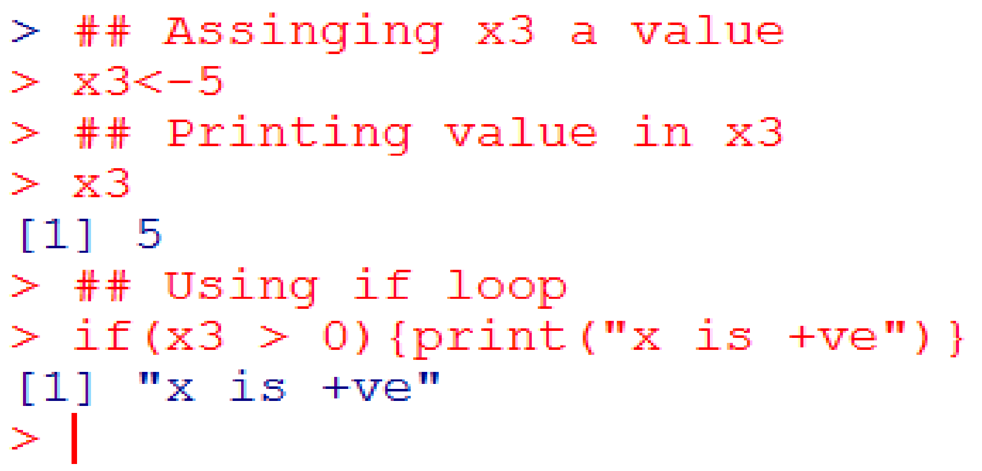
  Here, X is the numeric vector whose maximum value is 100.

  The same we are checking in if loop.



```
> ##Checking MAX value in vector x
> max(x)
[1] 100
> ##Using if loop and applying condition
> if(max(x)==100){
+ print("Vector x's maximum value is 100")
+ }
[1] "Vector x's maximum value is 100"
> ## Using if loop and applying condition
> ## Here, we are giving worng condition
> if(max(x)==99){
+ print("Vector x's maximum value is 100")
+ }
> |
```

When wrong condition is given , nothing is getting printed ion the screen



```
> ## Assinging x3 a value
> x3<-5
> ## Printing value in x3
> x3
[1] 5
> ## Using if loop
> if(x3 > 0){print("x is +ve")}
[1] "x is +ve"
> |
```

If-Else statement

- **If-else statement**: If the condition is true, if part is executed, or else part is executed
- The else part is optional and is evaluated `if test_expression` is FALSE
- It is important to note that else must be in the same line as the closing braces of the if statements
- **Syntax**

```
if (test_expression) {
       statement1
} else {
       statement2
 }
```

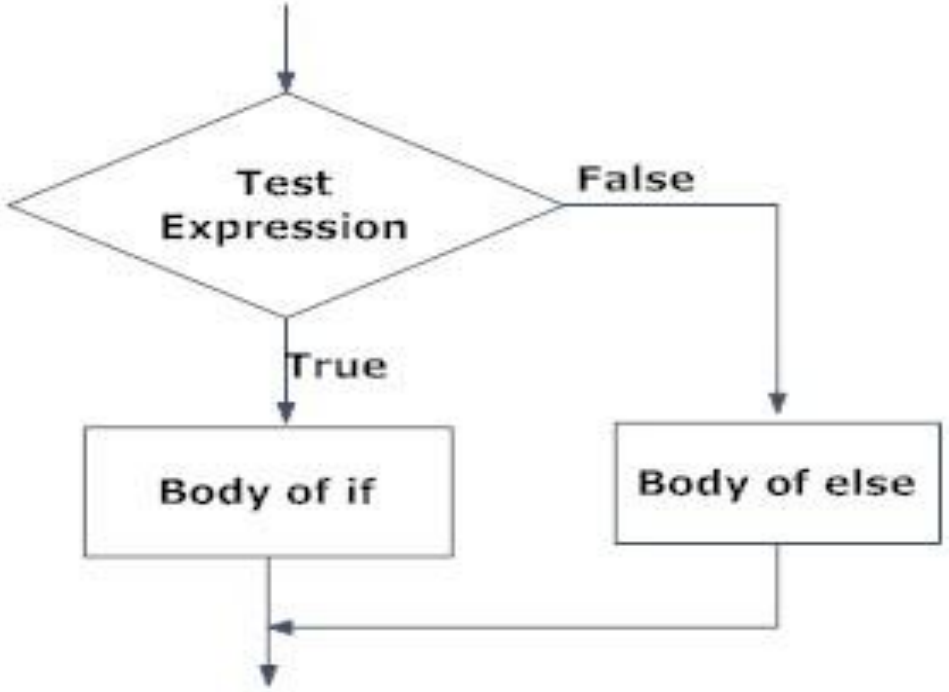
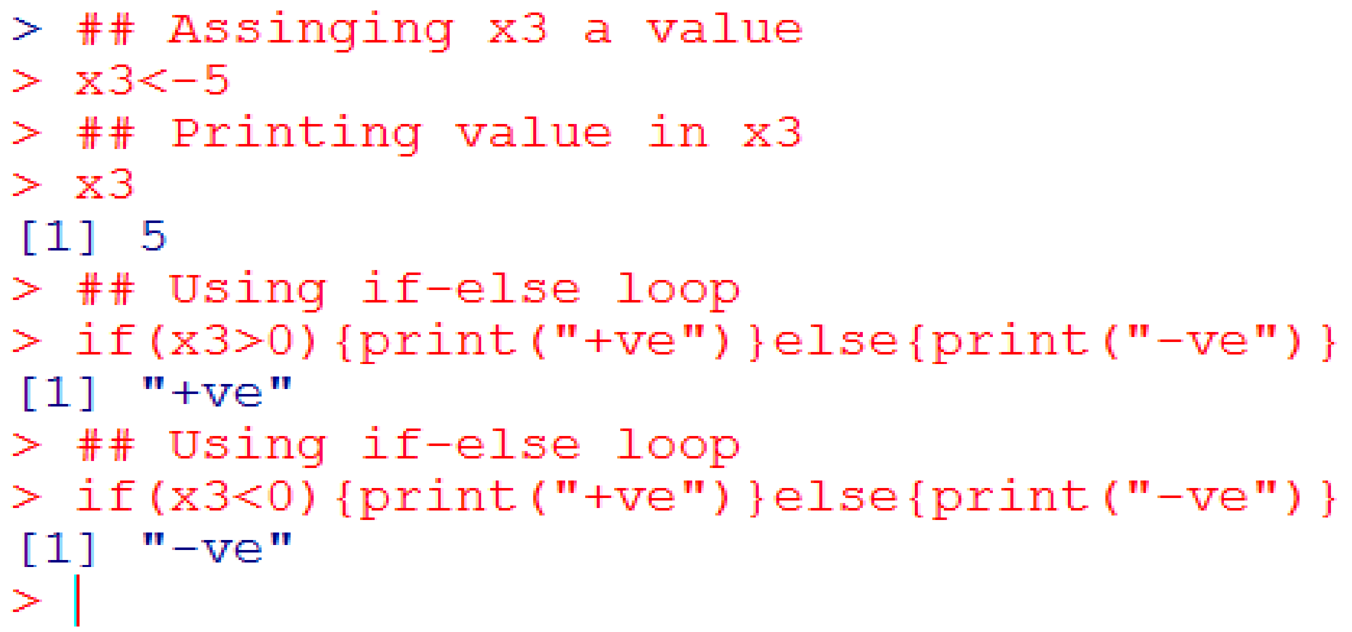


Fig: Operation of if...else statement

If-Else statement

- **Examples**
- Using if-else statement

- Another form of using if-else statement



```
> ## Assinging x3 a value
> x3<-5
> ## Printing value in x3
> x3
[1] 5
> ## Using if-else loop
> if(x3>0){print("+ve")}else{print("-ve")}
[1] "+ve"
> ## Using if-else loop
> if(x3<0){print("+ve")}else{print("-ve")}
[1] "-ve"
> 
```

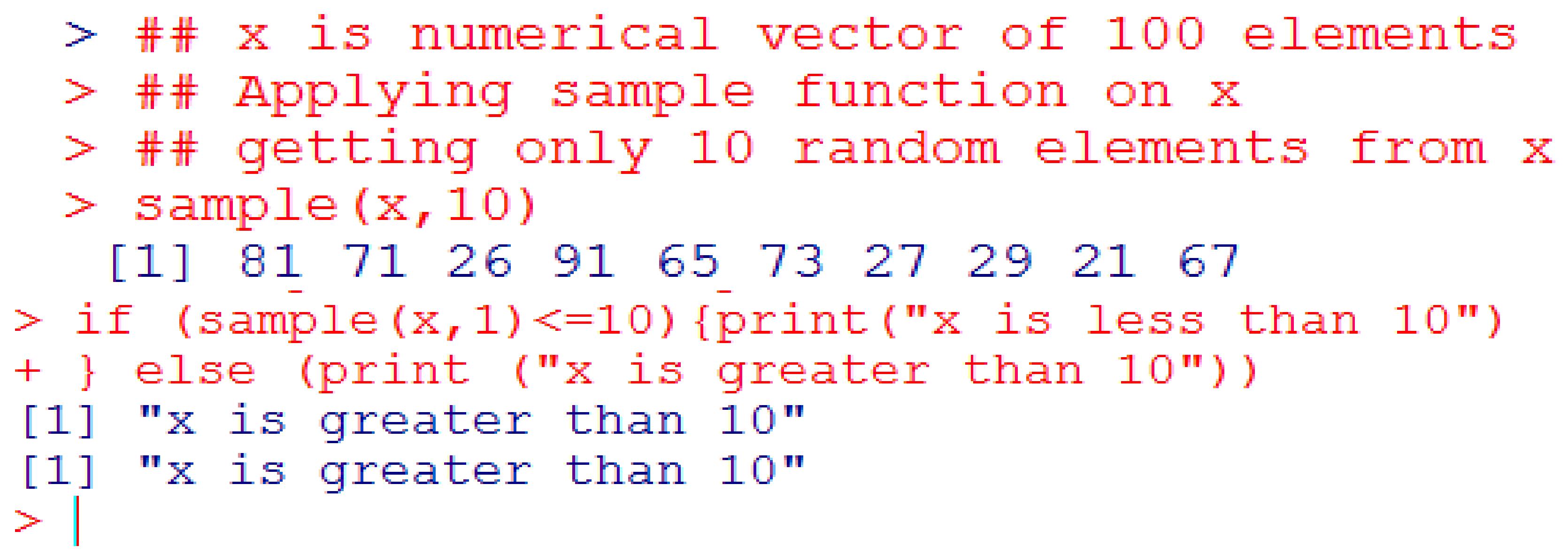
```
if(x > 0) print("Non-negative number") else print("Negative number")
```

##This feature of R allows us to write construct as shown below

```
> x <- -5
> y <- if(x > 0) 5 else 6
> y [1] 6
```

If-Else statement

- Another example of if-else statement



```
    > ## x is numerical vector of 100 elements
    > ## Applying sample function on x
    > ## getting only 10 random elements from x
    > sample(x,10)
     [1] 81 71 26 91 65 73 27 29 21 67
> if (sample(x,1)<=10){print("x is less than 10")
+ } else (print ("x is greater than 10"))
[1] "x is greater than 10"
[1] "x is greater than 10"
> 
```

Nested If-Else statement

- We can nest as many if…else statement as we want as follows
- **Syntax of nested if…else statement**

```
if (test_expression1) {
statement1
} else if (test_expression2) {
statement2
} else if (test_expression3) {
statement3
} else
statement4
```

```
x <- 0
if (x < 0) {
print("Negative number")
}
else if (x > 0) {
print("Positive number")
} else
print("Zero")
```

- Only one statement will get executed depending upon the test_expressions.

For loop

- A for loop is used to iterate over a vector, in R programming.
- **Syntax**

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

- Here, `sequence` is a vector and `val` takes on each of its value during the loop. In each iteration, `statement` is evaluated

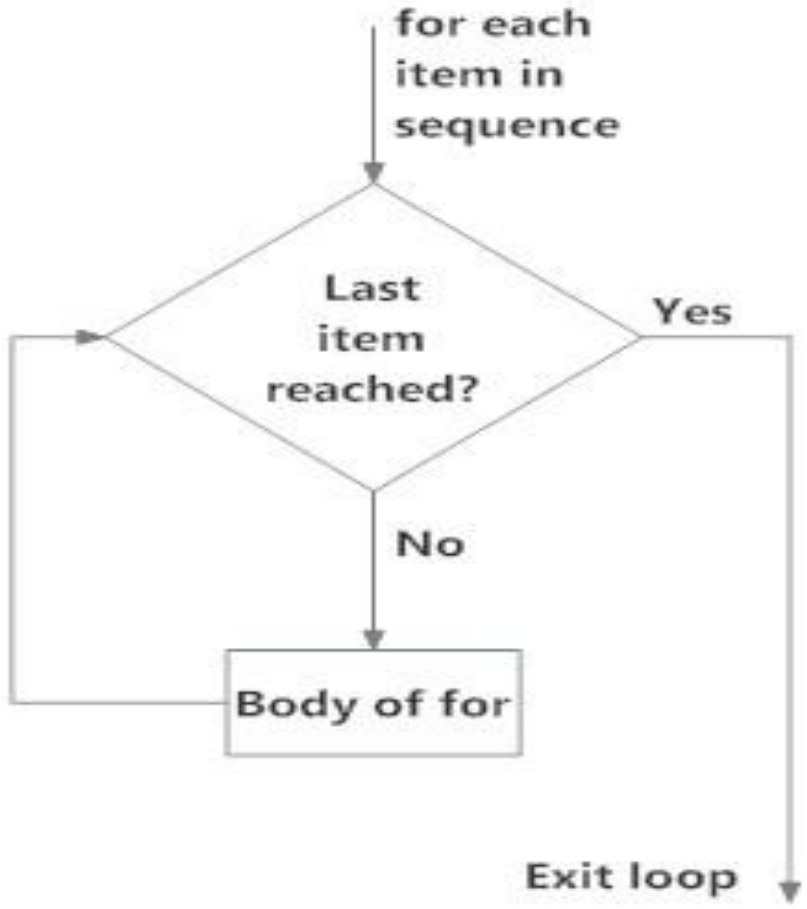
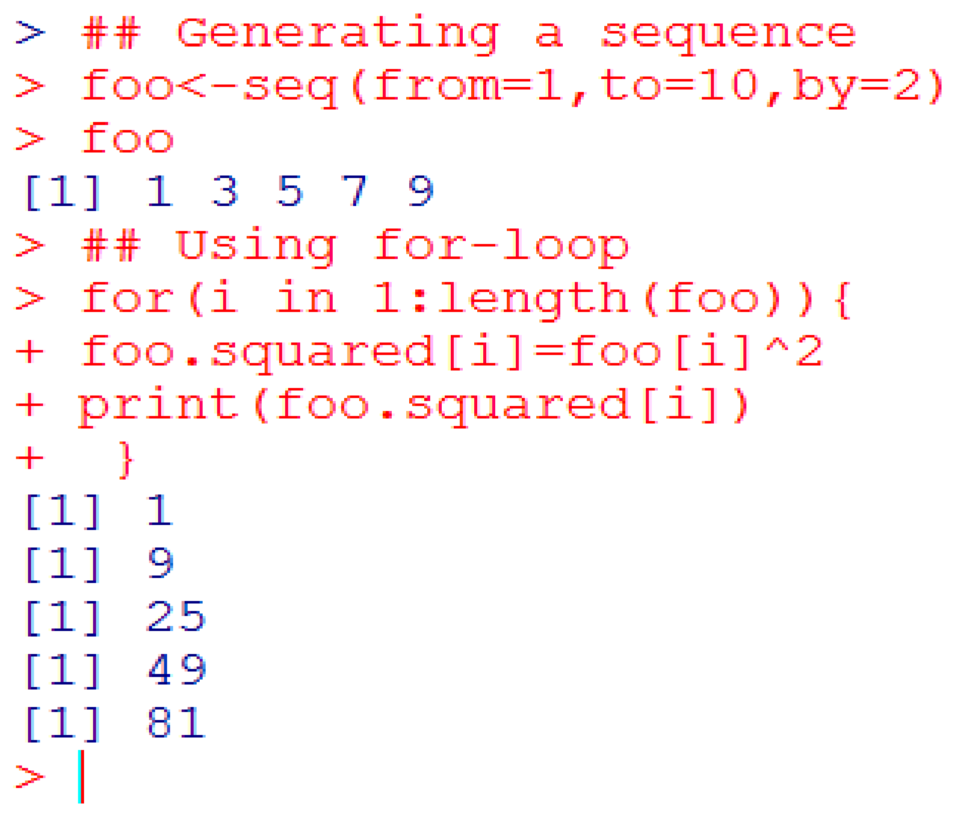


Fig: operation of for loop

For loop Example

- Example 1

```
foo = seq(1, 100, by=2)
foo.squared = NULL
for (i in 1:50) {
foo.squared[i] = foo[i]^2
print("foo.squared[i]")
}
```



```
> ## Generating a sequence
> foo<-seq(from=1,to=10,by=2)
> foo
[1] 1 3 5 7 9
> ## Using for-loop
> for(i in 1:length(foo)){
+ foo.squared[i]=foo[i]^2
+ print(foo.squared[i])
+   }
[1] 1
[1] 9
[1] 25
[1] 49
[1] 81
> 
```

For loop

- **Example of for loop**
- Below is an example to count the number of even numbers in a vector.

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

- In the above 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.

WHILE LOOP

- In R programming, while loops are used to loop until a specific condition is met.
- Syntax

```
while (test_expression) {
statement }
```

- Here, `test_expression` is evaluated and the body of the loop is entered if the result is TRUE.
- The statements inside the loop are executed and the flow returns to evaluate the `test_expression` again. This is repeated each time until `test_expression` evaluates to FALSE, in which case, the loop exits.

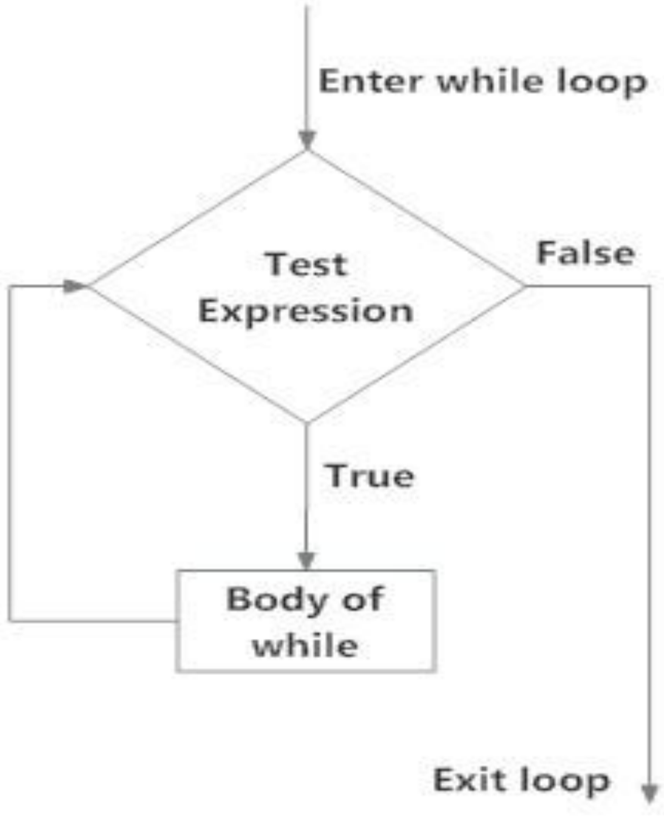
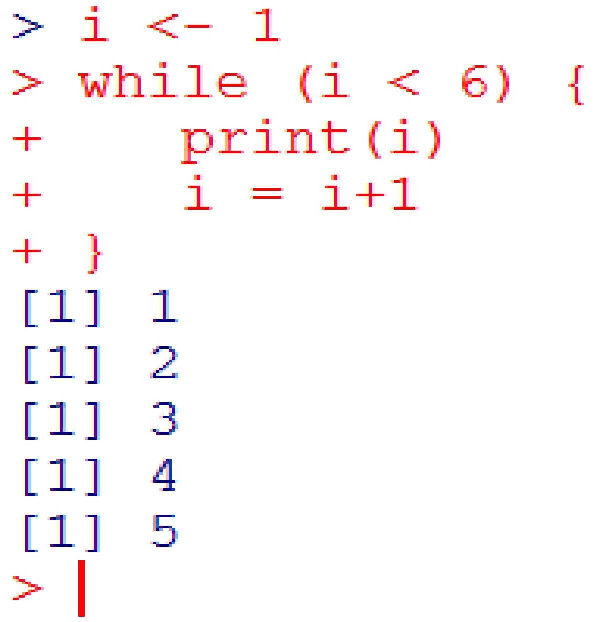


Fig: operation of while loop

While Loop

- **Example** : While loop

  - Here, i is initialized to 1 and the `test_expression` is i < 6 which evaluates to TRUE since 1 is less than 6.
  - So, the body of the loop is entered and i is printed and incremented. Incrementing i is important as this will eventually meet the exit condition. Failing to do so will result into an infinite loop.
  - In the next iteration, the value of i is 2 and the loop continues. This will continue until i takes the value 6.
  - The condition 6 < 6 will give FALSE and the loop finally exits.



```
> i <- 1
> while (i < 6) {
+     print(i)
+     i = i+1
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
> |
```

Break Statement

- A break statement is used inside a loop to stop the iterations and flow the control outside of the loop.

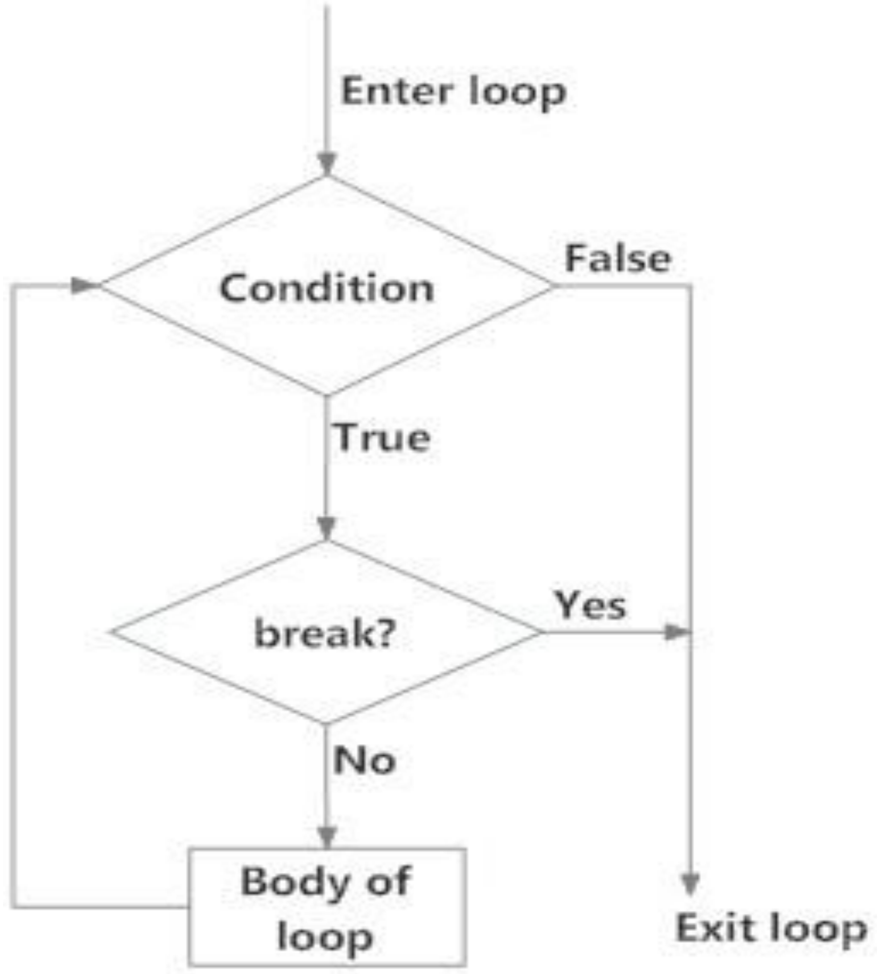
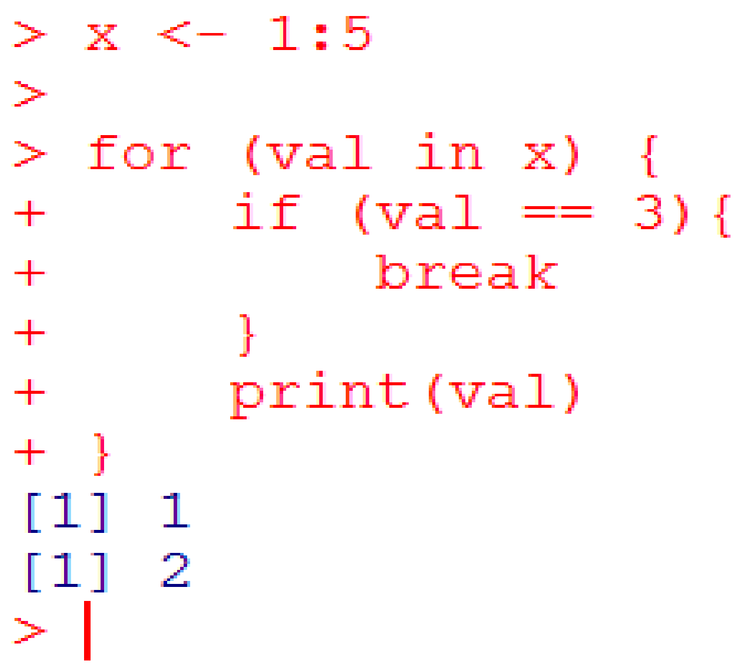


Fig: flowchart of break

Break Statement

- **Example:** break statement
  - In this example, we iterate over the vector x, which has consecutive numbers from 1 to 5.
  - Inside the for loop we have used a condition to break if the current value is equal to 3.
  - As we can see from the output, the loop terminates when it encounters the break statement.



```
> x <- 1:5
>
> for (val in x) {
+       if (val == 3){
+           break
+       }
+       print(val)
+ }
[1] 1
[1] 2
>
```

Next Statement

- A next statement is useful when we want to skip the current iteration of a loop without terminating it.
- On encountering next, the R parser skips further evaluation and starts next iteration of the loop.

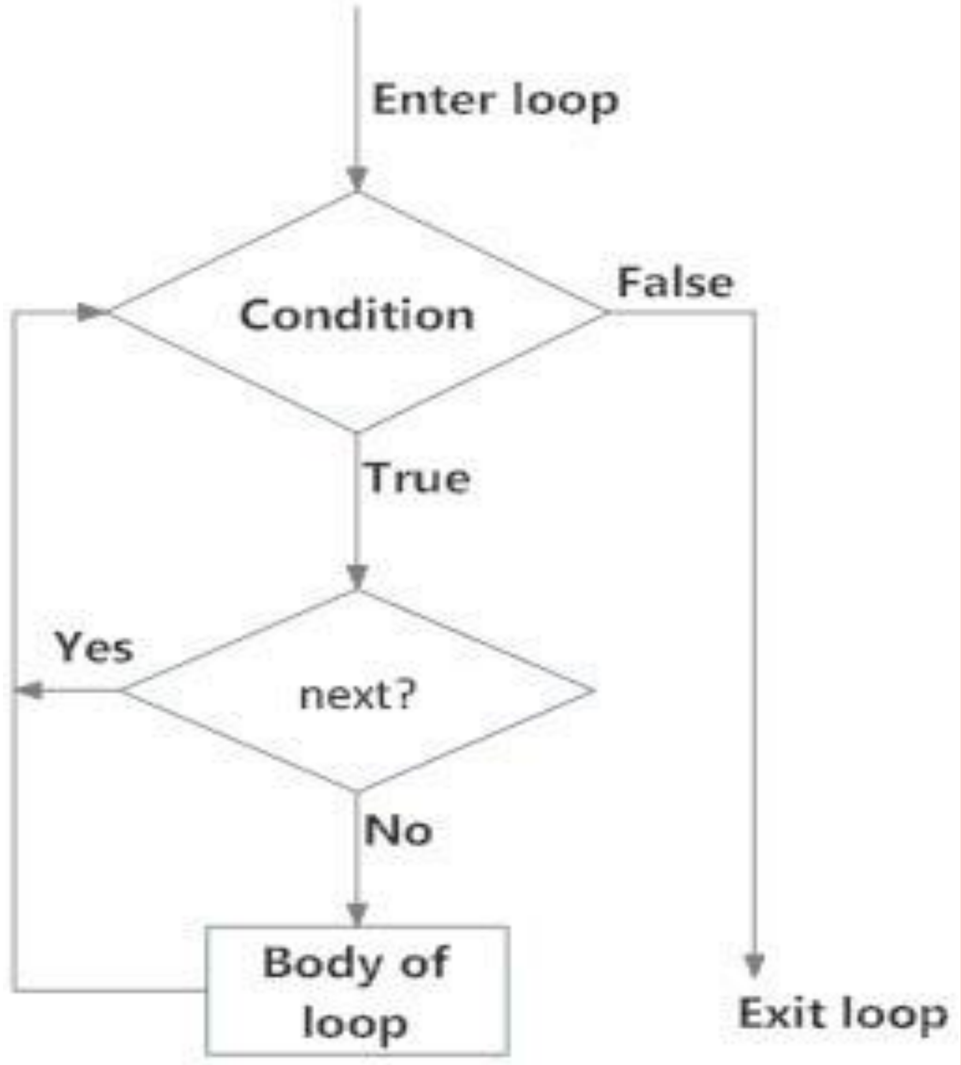
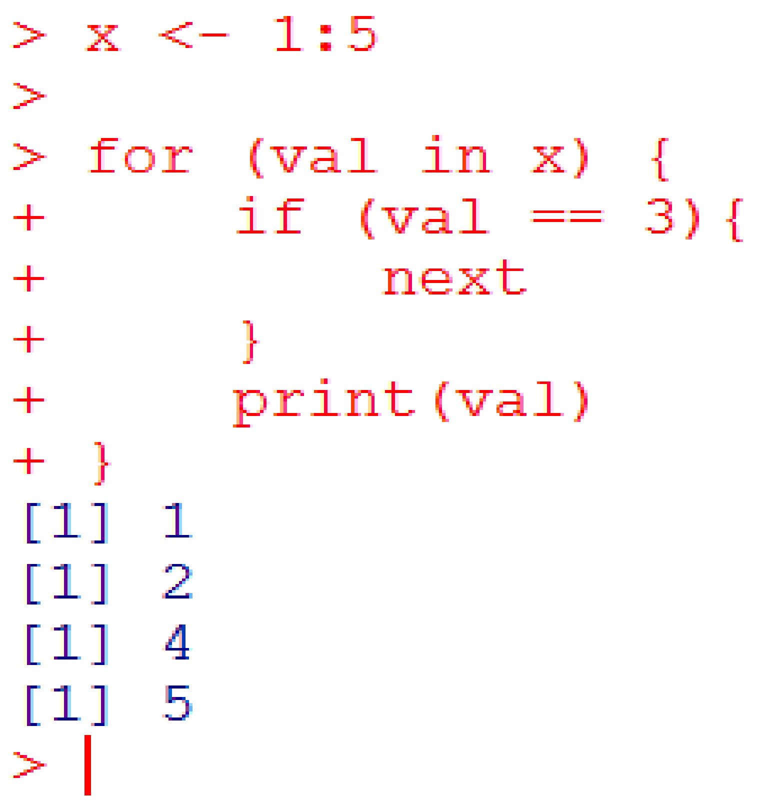


Fig: flowchart of next

Next Statement

- **Example:** next statement
  - In the above example, we use the next statement inside a condition to check if the value is equal to 3.
  - If the value is equal to 3, the current evaluation stops (value is not printed) but the loop continues with the next iteration.
  - The output reflects this situation.



```
> x <- 1:5
>
> for (val in x) {
+     if (val == 3){
+         next
+     }
+     print(val)
+ }
[1] 1
[1] 2
[1] 4
[1] 5
> 
```

THANK YOU !!!