Foundations of R Software

Lecture 22

Basics of Calculations

::::

Loops – for loop

Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

Control structures in R :

Control statements,

Functions,

Loops.

Control structures in R :

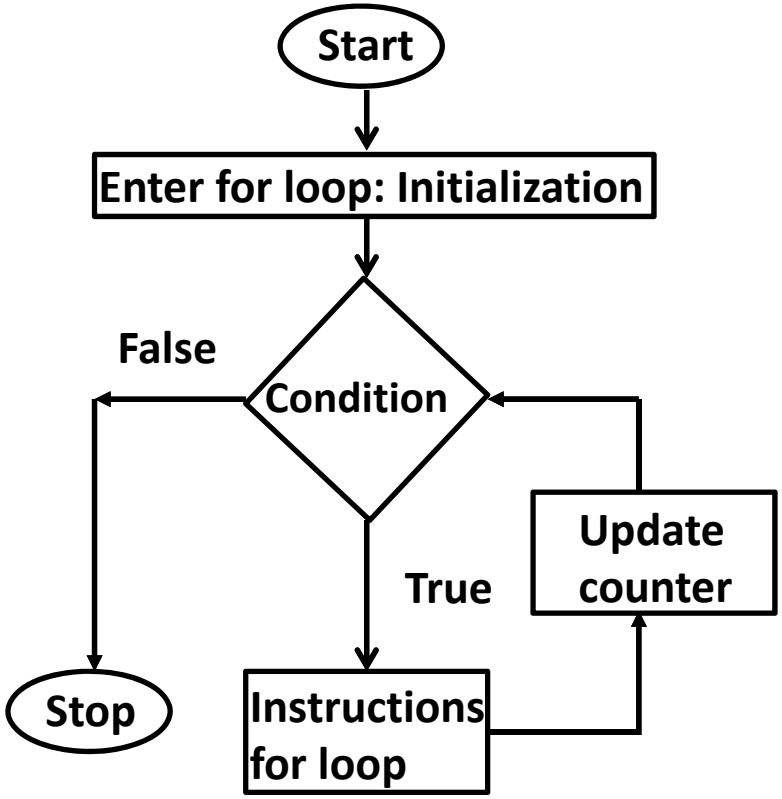
Loops

Repetitive commands are executed by loops

- **for loop**
- **while loop**
- **repeat loop**

1. The `for` loop

If the number of repetitions is known in advance (e.g. if all commands have to be executed for all cases *i* = 1,2,...,*n* in the data), a `for()` loop can be used.



1. The `for` loop

Syntax

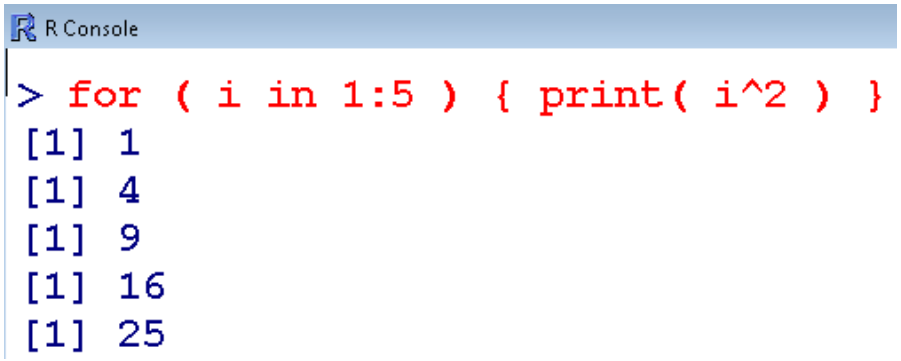
```
for (name in vector) {commands to be executed}
```

A variable with name `name` is sequentially set to all values, which contained in the vector vector.

All operations/commands are executed for all these values.

Example 1:

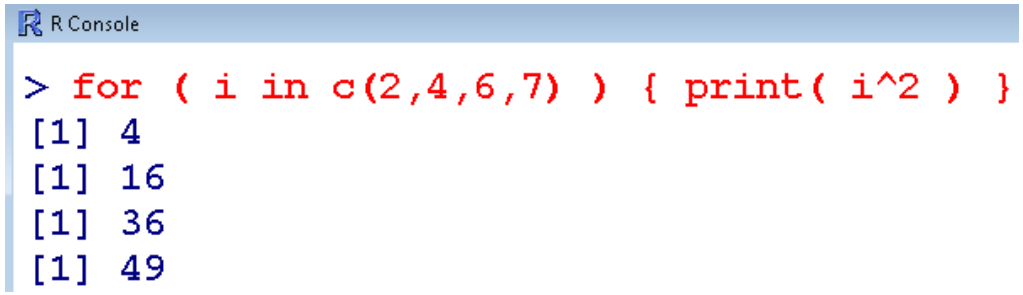
```
> for ( i in 1:5 ) { print( i^2 ) }
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```



Example 2:

Note: `print` **is a function to print the argument**

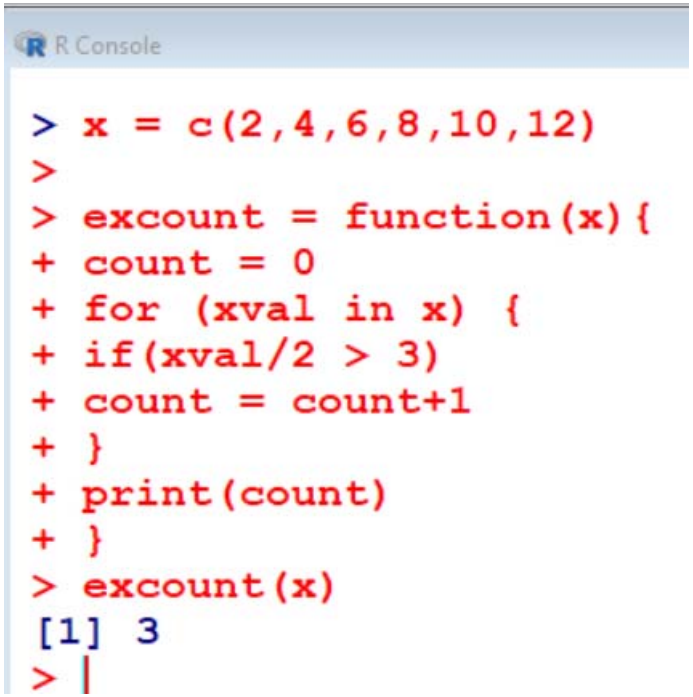
```
> for ( i in c(2,4,6,7) ) { print( i^2 ) }
[1] 4
[1] 16
[1] 36
[1] 49
```



Example 3:

```
x = c(2,4,6,8,10,12)

excount = function(x){
count = 0
for (xval in x) {
if(xval/2 > 3)
count = count+1
}
print(count)
}
excount(x)
[1] 3
```



Nested looping with `for` loop

We can have a loop inside a loop.

Example 4:

```
child = c("child1", "child2", "child3")
sweet = c("sweet1", "sweet2", "sweet3")


for (x in child) {
   for (y in sweet) {
      print(paste(x, y))
  }
}
```

Nested looping with `for` loop

```
> for (x in child) {
+    for (y in sweet) {
+       print(paste(x, y))
+   }
+ }
[1] "child1 sweet1"
[1] "child1 sweet2"
[1] "child1 sweet3"
[1] "child2 sweet1"
[1] "child2 sweet2"
[1] "child2 sweet3"
[1] "child3 sweet1"
[1] "child3 sweet2"
[1] "child3 sweet3"
```

Nested looping with `for` loop

R Console

```
> child = c("child1", "child2", "child3")
> sweet = c("sweet1", "sweet2", "sweet3")
>
> for (x in child) {
+    for (y in sweet) {
+       print(paste(x, y))
+    }
+ }
[1] "child1 sweet1"
[1] "child1 sweet2"
[1] "child1 sweet3"
[1] "child2 sweet1"
[1] "child2 sweet2"
[1] "child2 sweet3"
[1] "child3 sweet1"
[1] "child3 sweet2"
[1] "child3 sweet3"
>
```

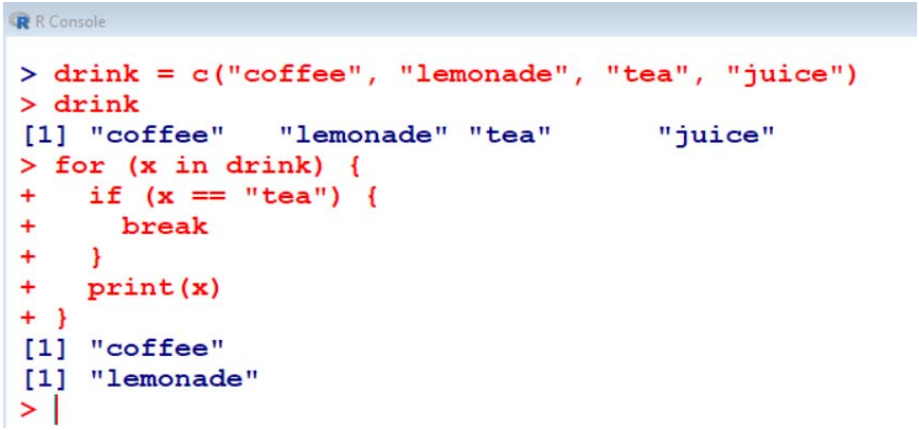
The break command

Using the `break` command, we can stop the loop before it has looped through all the items.

```
drink = c("coffee", "lemonade", "tea", "juice")


for (x in drink) {
  if (x == "tea") {
    break
  }
  print(x)
}
```

```
[1] "coffee"
[1] "lemonade"
```



The next command

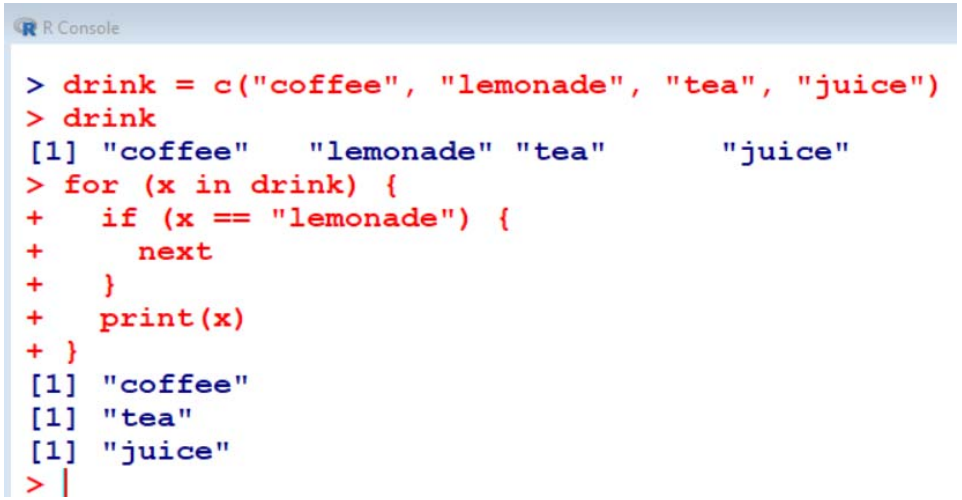
Using the `next` command, we can skip an iteration without terminating the loop.

Suppose we want to skip lemonade.

```
drink = c("coffee", "lemonade", "tea", "juice")

for (x in drink) {
  if (x == "lemonade") {
    next
  }
  print(x)
}
```

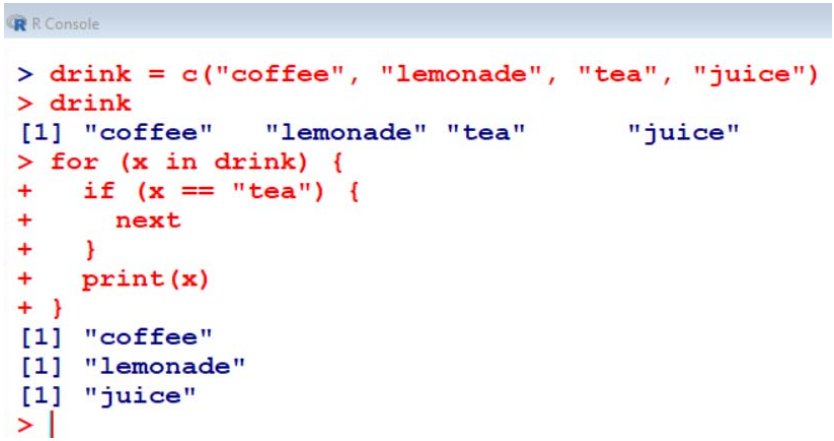
```
[1] "coffee"
[1] "tea"
[1] "juice"
```



The next command

```
drink = c("coffee", "lemonade", "tea", "juice")
```

Suppose we want to skip tea.

```
for (x in drink) {
  if (x == "tea") {
    next
  }
  print(x)
}
```

```
[1] "coffee"
[1] "lemonade"
[1] "juice"
```