Practical No. 4 Date: / / 201

Aim: Program to check given number is Prime number using function.

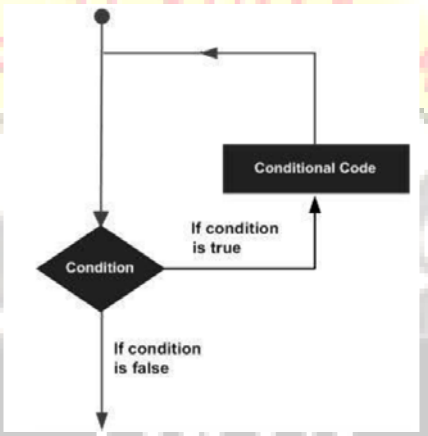
Objectives:

- To study R loops & functions.
- Implement a program to check given number as Prime number.

Theory:

R Loops

There may be a situation when you need to execute a block of code several number of times. A loop statement allows us to execute a statement or group of statements multiple times.



R provides the following types of looping statements.

Loop Type	Description
repeat	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
while	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
for	Like a while statement, except that it tests the condition at the end of the loop body.

repeat Loop

Syntax

```
repeat{
      // statement(s) to be executed.
      if(condition){
            break
      }
}
```

Example

```
V <- c(“Hello”, “Loop”)
count <- 2
repeat{
      print(V)
      count <- count+1
      if(count > 5){
            break
      }
}
```

Output

```
[1] “Hello Loop”
[1] “Hello Loop”
[1] “Hello Loop”
[1] “Hello Loop”
```

while Loop

Syntax

```
while(test_expression){
      // statement(s) to be executed.
}
```

Example

```
V <- c(“Hello”, “while Loop”)
count <- 2
while(count < 7){
      print(V)
      count = count+1
}
```

Output

```
[1] “Hello while Loop”
[1] “Hello while Loop”
[1] “Hello while Loop”
[1] “Hello while Loop”
[1] “Hello while Loop”
```

for Loop

Syntax

```
for(value in vector){
    // statement(s) to be executed.
}
```

Example

```
V <- LETTERS[1:4]
for(i in V){
    print(i)
}
```

Output

```
[1] “A”
[1] “B”
[1] “C”
[1] “D”
```

R Loop Control Statements

Loop control statements change execution from its normal sequence. R supports the following control statements.

Statement Type	Description
break	Terminates the loop statement and transfers execution to the statement immediately following the loop.
Next	Skip the current iteration of a loop without terminating it.

break Statement

Example

```
V <- c(“Hello”, “Loop”)
count <- 2
repeat{
    print(V)
    count <- count+1
    if(count > 5){
        break
    }
}
```

Output

```
[1] “Hello Loop”
[1] “Hello Loop”
[1] “Hello Loop”
[1] “Hello Loop”
```

next Statement

Example

```
V <- LETTERS[1:5]
for(i in V){
      if(i == “D”){
            next
      }
      print(i)
}
```

```
Output
[1] “A”
[1] “B”
[1] “C”
[1] “E”
```

R Functions

A function is a set of statements organized together to perform a specific task. In R, a function is an object so the R interpreter is able to pass control to the function, along with arguments (if any) to accomplish the action(s). After action(s) finished, it returns control to the interpreter as well as any result which may be stored in other objects.

Syntax

```
function_name <- function(arg#1, arg#2, …){
      #Function Body
}
```

- Function Name: Actual name of the function, stored as an object.
- Argument(s): A placeholder, used to pass value(s) to function, optional, can have default values.
- Function Body: collection of statements that defines what the function does.
- Return Value: last expression in the function body to be evaluated.

Types

R functions can be in-built or user-defined. The in-built functions can be directly called in the program without defining them first.
Example,

```
print(seq(32,44)) # Create a sequence of numbers from 32 to 44.
print(mean(25:82)) # Find mean of numbers from 25 to 82.
print(sum(41:68)) # Find sum of numbers frm 41 to 68.
Output,
[1] 32 33 34 35 36 37 38 39 40 41 42 43 44
[1] 53.5
[1] 1526
```

The user-defined functions are specific to what a user wants and once created they can be used like the in-built functions.
Example,

```
# Create a function to print squares of numbers in sequence.
sq_function <- function(a) {
      for(i in 1:a) {
            b <- i^2
            print(b)
      }
}
```

R Function Calling

Without An Argument

Example,

```
# Create a function to print squares of numbers in sequence.
sq_function <- function() {
      for(i in 1:5) {
            print(i^2)
      }
}
# Call the function without supplying arguments.
sq_function()
```

Output,

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```

With Argument (By Position & By Name)

Example,

```
# Create a function to perform arithmetic operation.
arith_function <- function(a,b,c) {
      result <- a*b+c
      print(result)
}
# Call the function by position of arguments.
arith_function(5,3,11)
# Call the function by name of arguments.
arith_function(a=11,b=5,c=3)
Output,
[1] 26
[1] 58
```

With Default Argument

Example,

```
# Create a function to perform multiplication.
mul_function <- function(a=3,b=6) {
      result <- a*b
      print(result)
}
# Call the function with default arguments.
mul_function()
# Call the function by new values to arguments.
mul_function(9,5)
```

Output,

```
[1] 18
[1] 45
```

Algorithm

1. Start.
2. Read input number.
3. Initialize loop and flag variables.
4. Using any loop, divide the given number with loop variable till it is 1 less than given number.
5. If at any point, remainder is zero, display “Given number is not Prime Number”
6. Else, display “Given number is Prime Number”
7. Stop.