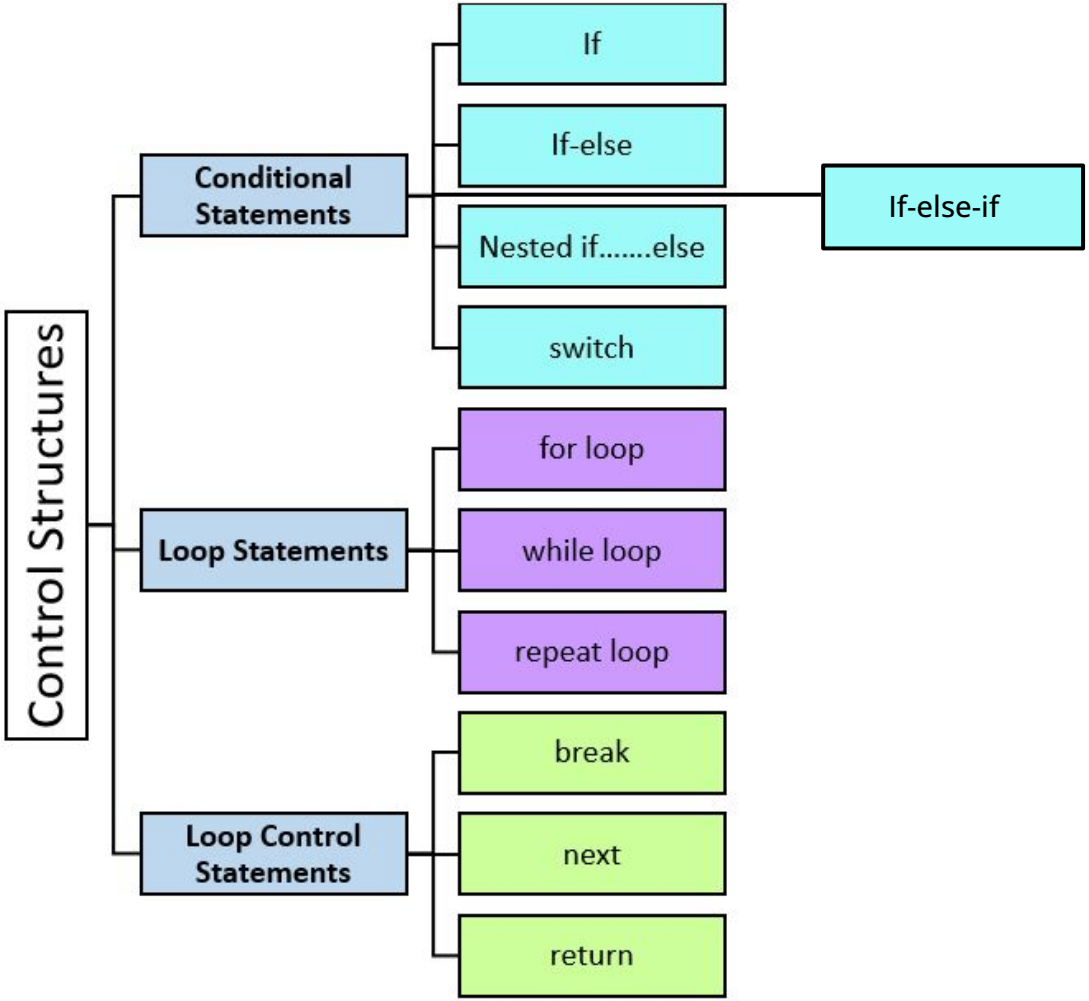
Module 1

Control Structures in R

Control structures allows us to control the flow of our programming

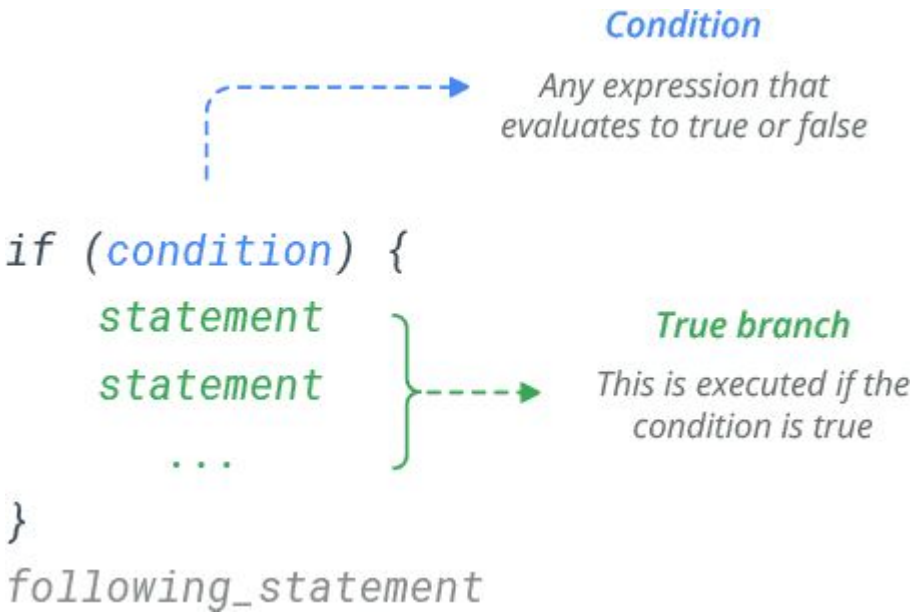


Conditional Statements

- **if Statement**: use it to execute a block of code, if a specified condition is true
- **else Statement**: use it to execute a block of code, if the same condition is false
- **else if Statement**: use it to specify a new condition to test, if the first condition is false
- **ifelse() Function**: use it when to check the condition for every element of a vector
- **Switch:** use it when to select values based only on a condition.

Simple if

Syntax



```r
x <- 7
y <- 5
if(x > y) {
  print("x is greater")
}
```

```
> x <- 7
> y <- 5
> if(x > y) {
+   print("x is greater")
+ }
[1] "x is greater"
```

Note: *In R, any non-zero value is considered TRUE, whereas a zero is considered FALSE*

If - Else

Syntax

```
if (condition) {
    statement
    statement
      ...
} else {
    statement
    statement
      ...
}
following_statement
```

True branch

This is executed if the condition is true

False branch

This is executed if the condition is false

```r
################ if else ##################
x <- 7
y <- 5
if(x > y) {
  print("x is greater")
} else {
  print("y is greater")
}
```

```
> x <- 7
> y <- 5
> if(x > y) {
+   print("x is greater")
+ } else {
+   print("y is greater")
+ }
[1] "x is greater"
```

If - Else If

Syntax

```
if (condition) {
   statement
   statement
      ...
} else if (condition) {
   statement
   statement
      ...
} else {
   statement
   statement
      ...
}
following_statement
```

First condition

This is executed if the first condition is true

New condition

A new condition to test if previous condition isn't true

False branch

This is executed if none of the conditions are true

```r
################## if else if ################
x <- 5
y <- 5
if(x > y) {
  print("x is greater")
} else if(x < y) {
  print("y is greater")
} else {
  print("x and y are equal")
}
```

```
> x <- 5
> y <- 5
> if(x > y) {
+   print("x is greater")
+ } else if(x < y) {
+   print("y is greater")
+ } else {
+   print("x and y are equal")
+ }
[1] "x and y are equal"
```

ifelse()

Syntax

- performs an elementwise if…else check on the vector and returns a result vector based on the conditions.

```
ifelse (condition, TrueVector, FalseVector)
```

Condition	*True branch*	*False branch*
Condition is checked for every element of a vector	*Select element from this if the condition is true*	*Select element from this if the condition is false*

```r
v <- c(1,2,3,4,5,6)
ifelse(v %% 2 == 0, "even", "odd")
```

```
> v <- c(1,2,3,4,5,6)
> ifelse(v %% 2 == 0, "even", "odd")
[1] "odd"  "even" "odd"  "even" "odd"  "even"
```

switch

Syntax

```
variable <- switch(expression, val1, val2, ...)
```

Expression

It can be either a number or a character

List of values

Value is selected based on the name or position

```r
x <- "c"
v <- switch(x, "a"="apple", "b"="banana", "c"="cherry")
v
```

```
> x <- "c"
> v <- switch(x, "a"="apple", "b"="banana", "c"="cherry")
> v
[1] "cherry"
```

```r
x <- 1
v <- switch(x, "apple", "banana", "cherry")
v
```

```
> x <- 1
> v <- switch(x, "apple", "banana", "cherry")
> v
[1] "apple"
>
```

for loop

Syntax

Var

It takes items from iterable one by one

Iterable

It's a collection of objects (like a vector, list etc.)

```
for (var in iterable) {
    statement
    statement
      ...
}
following_statement
```

Loop body

It is executed once for each item in iterable

```r
# Print 'Hello!' 3 times
for (x in 1:3) {
  print("Hello!")
}
```

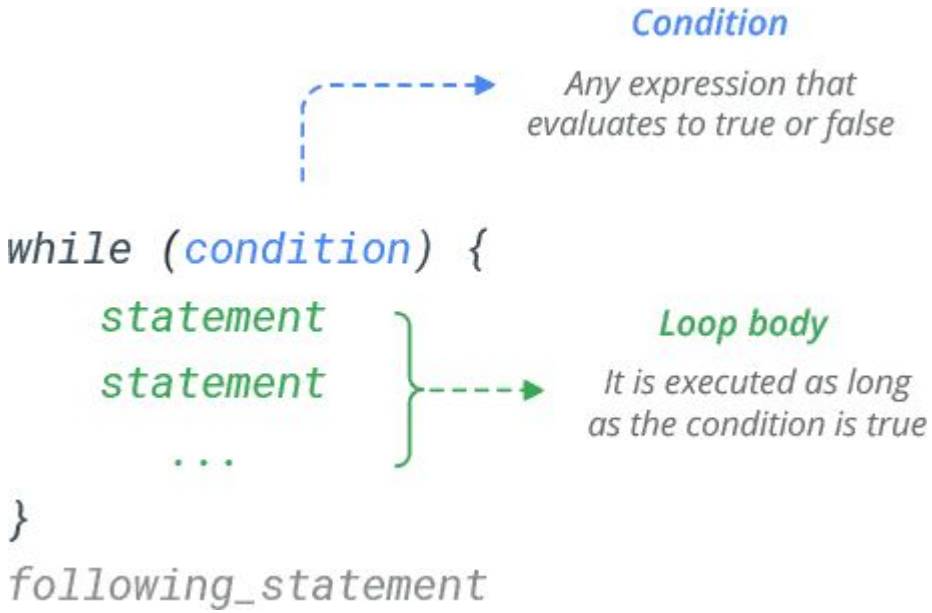
```
> # Print 'Hello!' 3 times
> for (x in 1:3) {
+   print("Hello!")
+ }
[1] "Hello!"
[1] "Hello!"
[1] "Hello!"
```

```r
for(x in 1:3) {
  for(y in 1:2) {
    print(paste(x, y))
  }
}
```

```
> for(x in 1:3) {
+   for(y in 1:2) {
+     print(paste(x, y))
+   }
+ }
[1] "1 1"
[1] "1 2"
[1] "2 1"
[1] "2 2"
[1] "3 1"
[1] "3 2"
>
```

while loop

Syntax



```r
################### while loop ##########
x <- 5
while (x) {
  print(x)
  x <- x - 1
}
```

```
> ################### while loop ##################
> x <- 5
> while (x) {
+   print(x)
+   x <- x - 1
+ }
[1] 5
[1] 4
[1] 3
[1] 2
[1] 1
```

repeat loop

Syntax

```
repeat {
    statement
    statement
      ...
}
following_statement
```

Loop body

Repeat this block of code indefinitely

- A repeat loop without a break statement results into an infinite/endless loop.
- However, you can put a condition explicitly inside the body of the loop and use the `break` statement to exit the loop.

```r
repeat {
  print("Press Esc to stop me!")
}
```

```
R 4.3.1 · ~/
[1] "Press Esc to stop me!"
[1] "Press Esc to stop me!"
[1] "Press Esc to stop me!"
[1] "Press Esc to stop me!"
[1] "Press Esc to stop me!"
[1] "Press Esc to stop me!"
[1] "Press Esc to stop me!"
```

```r
x <- 1
repeat {
  print(x)
  if (x > 4)
    break
  x <- x + 1
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

break - used to terminate the execution of a loop

Console Terminal

```
> for(i in 1:5) {                    # for-loop with break
+
+   if(i == 4) {
+     break
+   }
+
+   print(paste("This is step", i))
+ }
[1] "This is step 1"
[1] "This is step 2"
[1] "This is step 3"
>
```

Break at Step 4

Note :

- can only be used inside a loop.
- can be used only once per loop.
- will only terminate the innermost loop. If the break statement is inside a nested loop, it will only terminate the innermost loop and the outer loop will continue to execute.

next - to skip the current iteration of a loop without terminating it.

Console Terminal

```
> for(i in 1:5) {                    # for-loop with next
+
+   if(i == 4) {
+     next
+   }
+
+   print(paste("This is step", i))
+ }
[1] "This is step 1"
[1] "This is step 2"
[1] "This is step 3"
[1] "This is step 5"
>
```

Step 4 was skipped

Note:

In R language continue statement is referred to as the next statement.

return

```
########### return ########################
  for (i in 1:10) {
    if (i == 5) {
      return()
    }
    print(i)
  }
```

```
[1] 1
[1] 2
[1] 3
[1] 4
>
```

- can only be used inside a function.
- can only be used once per loop.
- can be used to return any value, including a vector, a list, or a data frame.

Thank You