Short Topic Summary: for + if/else

2026-01-12

Short Topic Summary: for + if/else

1. For

```
items <- c(1,2,3)

for (i in items) {
  # operation to be performed on each item
}
```

items → a list / vector (anything that contains multiple values)

i → represents a single value from the list

The loop executes the following logic:

"Take each element inside items, one by one, and run the code inside { }."

```
leaf_lengths <- c(3.2, 5.1, 4.0)
```

c() → stands for combine and is used to create vectors.

Here, we combine three numbers into a single vector.

This vector is named leaf_lengths.

```
for (u in leaf_lengths) {
  print(u)
}
```

```
## [1] 3.2
## [1] 5.1
## [1] 4
```

Take each leaf length one by one and print it to the screen.

First iteration → first element

Second iteration → second element

Third iteration → third element

The loop continues until the list ends.4

```
results <- c()
```

We create an empty vector (list).

We will use this vector to store results.

Daily life analogy:

We took an empty basket that currently contains nothing.

```
for (u in leaf_lengths) {
results <- c(results, u * 2)
print(results)
}
```

```
## [1] 6.4
## [1]  6.4 10.2
## [1]  6.4 10.2  8.0
```

u selects each element in the yaprak_uzunluklari vector one by one, multiplies it by 2, and places the result into the sonuclar vector.

Take each leaf length, multiply it by 2, and put the result into the basket.

The for loop takes each value in the list one by one. sonuclar stores the outputs produced from these values.

In this way, we can automatically process large amounts of biological data.

2. if / else

if / else allows us to tell the computer:

"If this condition is true, do this. Otherwise, do that."

Example: If it is raining → take an umbrella Otherwise → do not take one

```
#if (condition) {
  # code to be executed
#}
```

condition → must be TRUE or FALSE

If the condition is true, the code inside { } runs. If it is false → nothing happens.

```
leaf_length <- 6

if (leaf_length > 5) {
print("Long leaf")
}
```

```
## [1] "Long leaf"
```

leaf_length > 5 → TRUE

Since it is TRUE, the text is printed.

If the length were 4:

The condition would be FALSE

Nothing would be printed

```
"if (condition) {
  # if true
} else {
  # if false
}"
```

```
## [1] "if (condition) {\n  # if true\n} else {\n  # if false\n}"
```

If the first condition is true, that code runs. Otherwise, the other code runs.

```r
leaf_length <- 4

if (leaf_length > 5) {
print("Long leaf")
} else {
print("Short leaf")
}
```

```
## [1] "Short leaf"
```

$4 > 5 \rightarrow$ FALSE

The else part runs.

"Short leaf" is printed.

When using if + else, the computer always makes a decision.

```r
if (leaf_length < 3) {
print("Very short")
} else if (leaf_length < 6) {
print("Medium")
} else {
print("Long")
}
```

```
## [1] "Medium"
```

If very short $\rightarrow$ small Else if medium $\rightarrow$ medium Else $\rightarrow$ long

The computer:

Checks the first condition

If true, stops

If false, moves to the next condition

Continues until the end

3. Using for together with if / else

for $\rightarrow$ takes each value one by one

if / else $\rightarrow$ makes a decision about that value

Classifying leaf lengths

```r
leaves <- c(2.5, 4.8, 6.1)

for (u in leaves) {
  if (u > 5) {
    print("Long leaf")
  } else {
    print("Short leaf")
  }
}
```

```
## [1] "Short leaf"
## [1] "Short leaf"
## [1] "Long leaf"
```

Take the leaf Ask whether it is long Print the result based on the answer Move on to the next leaf This structure represents the basic logic of biological data analysis.

Mini Example: GC% and Motif Analysis in DNA Sequences

Example 1: Checking GC Content in a DNA Sequence

We have a DNA sequence. For each nucleotide, we will print whether it forms a GC strong bond or an AT weak bond.

```
DNA_sequence <- c("A", "G", "C", "T", "G")

# Take each nucleotide one by one
for (nukleotit in DNA_sequence) {

  # If the nucleotide is G or C
  if (nukleotit == "G" | nukleotit == "C") {
    print(paste(nukleotit, ": GC Strong Bond"))
  } else {
    print(paste(nukleotit, ": AT Weak Bond"))
  }
}
```

```
## [1] "A : AT Weak Bond"
## [1] "G : GC Strong Bond"
## [1] "C : GC Strong Bond"
## [1] "T : AT Weak Bond"
## [1] "G : GC Strong Bond"
```

Example 2: Simple Motif Search (On a Vector)

We have 4 different DNA regions. Let's identify the regions that contain "ATG" (start codon).

```
regions <- c("CCG", "ATG", "TTA", "ATG")

for (i in 1:length(regions)) {

  if (regions[i] == "ATG") {
    print(paste(i, ". region contains a start codon!"))
  } else {
    print(paste(i, ". region is clean."))
  }
}
```

```
## [1] "1 . region is clean."
## [1] "2 . region contains a start codon!"
## [1] "3 . region is clean."
## [1] "4 . region contains a start codon!"
```

Example 3: GC Ratio Calculation

Let's count how many G and C bases are in a sequence and calculate the ratio.

```
dna <- c("G", "C", "A", "T", "G", "G", "A")
gc_number <- 0 # Create an empty basket/counter
```

```r
for (n in dna) {
  if (n == "G" | n == "C") {
  gc_number <- gc_number + 1 # Increase the counter by 1 each time
  }
}

total_length <- length(dna)
oran <- (gc_number / total_length) * 100

print(paste("Total GC Count:", gc_number))
```

```
## [1] "Total GC Count: 4"
```

Example 4: Multiple Conditions (Temperature Control – Enzyme Activity)

Let's check the behavior of an enzyme at different temperatures.

```r
temperatures <- c(10, 37, 55, 90)

for (t in temperatures) {
  if (t < 20) {
    print(paste(t, "degrees: Enzyme is very slow."))
  } else if (t >= 20 & t <= 45) {
    print(paste(t, "degrees: Optimum working temperature."))
  } else {
    print(paste(t, "degrees: Enzyme structure may be damaged (Denaturation)."))
  }
}
```

```
## [1] "10 degrees: Enzyme is very slow."
## [1] "37 degrees: Optimum working temperature."
## [1] "55 degrees: Enzyme structure may be damaged (Denaturation)."
## [1] "90 degrees: Enzyme structure may be damaged (Denaturation)."
```