Pertemuan 3 - Conditional Expressions

plirapli

2024-09-18

Conditional Expressions

Conditional expression adalah pernyataan untuk memeriksa kondisi dan menjalankan aksi tertentu berdasarkan apakah kondisi tersebut benar (TRUE) atau salah (FALSE).

```
# Kita di sini bakalan make dataset murders
library(dslabs)
data(murders)
```

Nampilin dataset murders

```
# View(murders) # <- Pas mau diknit di-comment dulu kodenya
```

IF-ELSE IF-ELSE

Struktur IF-ELSE digunakan untuk melakukan percabangan logika berdasarkan kondisi tertentu.

Contoh if else

```
nilai = 98

if (nilai >= 85) {
  print("A")
} else if (nilai >= 80 && nilai < 85) {
  print("B+")
} else if (nilai >= 75 && nilai < 80) {
  print("B")
} else if (nilai >= 70 && nilai < 75) {
  print("C+")
} else if (nilai >= 60 && nilai < 70) {
  print("C")
} else if (nilai >= 50 && nilai < 60) {
  print("D")
} else {
  print("bjir ngulang")
}
```

Contoh 1

```
## [1] "A"
```

Contoh 2 Pada contoh misalkan di sini kita mau mencari negara bagian mana yang punya tingkat pembunuhan kurang dari 0.5

```
# Buat data murder_rate
murder_rate = murders$total / murders$population * 100000

for (i in 1:nrow(murders)) {
  if (murder_rate[i] < 0.5) {
    print(murders$state[i])
  } else {
    print("-")
  }
}
```

```
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "New Hampshire"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
```

```
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "Vermont"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
## [1] "-"
```

IFELSE

Selain if else normal, di R juga ada fungsi ifelse. Ini mirip seperti versi pendek dari if else normal. Fungsi ini membutuhkan 3 argumen: kondisi, kondisi ketika true, dan kondisi ketika false (mirip ternary operator).

```
a = 0
ifelse(a > 0, "kurang dari 0", "lebih dari 0")
```

```
## [1] "lebih dari 0"
```

Fungsi ini biasanya digunakan untuk mengevaluasi data dengan tipe vektor. IFELSE akan memeriksa setiap elemen dari vektor. Jadi ga perlu looping.

Misalkan di sini kita mau ngecek populasi dari tiap state. Kalo populasi < 1jt, maka output "dikit", tapi kalo >= 1jt, output "banyak".

```
ifelse(murders$population < 1000000, "dikit", "banyak")
```

```
##  [1] "banyak" "dikit"  "banyak" "banyak" "banyak" "banyak" "banyak" "dikit"
##  [9] "dikit"  "banyak" "banyak" "banyak" "banyak" "banyak" "banyak" "banyak"
## [17] "banyak" "banyak" "banyak" "banyak" "banyak" "banyak" "banyak" "banyak"
## [25] "banyak" "banyak" "dikit"  "banyak" "banyak" "banyak" "banyak" "banyak"
## [33] "banyak" "banyak" "dikit"  "banyak" "banyak" "banyak" "banyak" "banyak"
## [41] "banyak" "dikit"  "banyak" "banyak" "banyak" "dikit"  "banyak" "banyak"
## [49] "banyak" "banyak" "dikit"
```

ANY & ALL

Fungsi ANY mengevaluasi vektor logika dan mengembalikan BENAR jika salah satu entri bernilai BENAR. Sedangkan fungsi ALL mengevaluasi vektor logika dan mengembalikan BENAR jika semua entri bernilai BENAR.

Misal di sini kita mau ngecek ada ga sih state yang total kasus pembunuhannya itu lebih dari 5000. Nah, kita bisa pake fungsi any(), kalo ada 1 state yang kasusnya >5000, dia bakalan bernilai true.

Beda dengan any, kalo all setiap elemen harus bernilai true supaya dia true.

```r
any(murders$total >= 5000)
```

```
## [1] FALSE
```

```r
all(murders$total >= 2)
```

```
## [1] TRUE
```

```r
# Silahkan coba coba untuk mengubah aturan dari logika IF-ELSE tersebut.
```

FUNCTION

Fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu. Fungsi memudahkan kita untuk menulis kode yang bersih, terstruktur, dan dapat digunakan kembali. R memiliki banyak fungsi bawaan (built-in), tetapi kita juga dapat membuat fungsi sendiri menggunakan sintaks function.

Membuat Fungsi di R

```r
nama_fungsi = function(argumen 1, argumen 2, ...) {
  # Kode yang akan dijalankan
  hasil = argumen 1 + argumen 2     # Contoh operasi
  return(hasil)                     # Mengembalikan hasil
}
```

Contoh bikin fungsi di R

```r
# Bikin fungsi
salam = function() {
  print("Praktikum DS Pertemuan 3")
}

# Memanggil fungsi
salam()
```

1. Fungsi Sederhana tanpa Argumen

```
## [1] "Praktikum DS Pertemuan 3"
```

```r
penjumlahan = function(a = 4, b) {
  hasil = a + b
  return(hasil)
}

# Memanggil fungsi dengan argumen
penjumlahan(3, 5)
```

2. Fungsi dengan Argumen

```
## [1] 8
```

```
pengurangan = function(a = 5, b){
  hasil = a - b
  return(hasil)
}

# Memanggil fungsi tanpa mengubah nilai default
pengurangan(, 10)
```

3. Fungsi dengan Nilai Default pada Argumen

```
## [1] -5
```

```
# Memanggil fungsi dengan nilai argumen yang berbeda
pengurangan(10, 3)
```

```
## [1] 7
```

```
# Fungsi Rata-Rata
mean(c(1,2,3,4,5))
```

4. Fungsi bawaan pada R

```
## [1] 3
```

```
# Fungsi menghitung Total
sum(c(1,2,3,4,5))
```

```
## [1] 15
```

```
# Fungsi mengurutkan data
sort(c(2,5,1,4,3))
```

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

```
# dan masih banyak lagi...
```

Perulangan (FOR LOOP)

Struktur For Loop R

```
for (variable in sequence) {
  Kode yang akan dijalankan pada setiap iterasi
}
```

Contoh For Loop

```
angka = c(1, 2, 3, 4, 5)

for (i in angka) {
  print(i)
}
```

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

```
for (i in 1:5) {
  print(i)
}
```

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

Penjelasan:
- **for (i in 1:5)**: i adalah variabel yang akan berubah nilainya dari 1 s/d 5, satu per satu.
- **print(i)**: Perintah yang dijalankan di setiap iterasi loop. print(i) akan menampilkan nilai dari 1 s/d 5.

Contoh dalam Data Frame

```
data_siswa = data.frame(
  nama = c("Sugab", "Awginc", "Desyan", "Bagus"),
  matematika = c(85, 70, 90, 65),
  bahasa_inggris = c(80, 60, 88, 75)
)

print(data_siswa)
```

```
##     nama matematika bahasa_inggris
## 1  Sugab         85             80
## 2 Awginc         70             60
## 3 Desyan         90             88
## 4  Bagus         65             75
```

```
# Bikin 2 kolom baru, rata_rata dan status yang awalnya diisi dengan NA (Not Available), artinya masih
data_siswa$rata_rata = NA
```

```r
data_siswa$status = NA

# Loop ini akan berjalan untuk setiap baris dalam dataframe.
for (i in 1:nrow(data_siswa)) {
  # Menghtiung rerata nilai tiap siswa
  data_siswa$rata_rata[i] = (data_siswa$matematika[i] + data_siswa$bahasa_inggris[i]) / 2

  if (data_siswa$rata_rata[i] >= 75) {
    data_siswa$status[i] = "Lulus"
  } else {
    data_siswa$status[i] = "Tidak Lulus"
  }
}

data_siswa
```

Menghitung Nilai Rata-Rata dan Status

```
##     nama matematika bahasa_inggris rata_rata      status
## 1  Sugab         85             80      82.5       Lulus
## 2 Awginc         70             60      65.0 Tidak Lulus
## 3 Desyan         90             88      89.0       Lulus
## 4  Bagus         65             75      70.0 Tidak Lulus
```

SAPPLY

sapply() adalah salah satu fungsi yang digunakan untuk menerapkan operasi atau fungsi tertentu pada elemen-elemen dari sebuah objek seperti vektor, list, atau data frame.

Sintkas dasar sapply()

`sapply(X, FUN, ...)`

Fungsi sapply() mengambil tiga argumen utama:

- X: Data (vektor, list, atau dataframe) yang ingin kamu iterasi.
- FUN: Fungsi yang ingin kamu terapkan pada setiap elemen data.
- ...: Argumen tambahan opsional untuk fungsi yang diterapkan.

Contoh penggunaaan sapply()

Contoh 1 Misalkan kita punya vektor yang isinya angka dan ingin menghitung akar kuadrat dari setiap elemen. Nah, daripada harus nge-looping, kita bisa pake fungsi sapply().

```r
# Bikin vektor
angka = c(81, 25, 36, 121, 100, 4, 64, 16)
```

```r
# Kalo pake looping
# Membuat vektor kosong untuk menyimpan hasil
hasil_akar_kuadrat = numeric(length(angka))
```

```r
for (i in 1:length(angka)) {
  hasil_akar_kuadrat[i] = sqrt(angka[i])
}
hasil_akar_kuadrat
```

```
## [1]  9  5  6 11 10  2  8  4
```

```r
# Kalo pake sapply()
sapply(angka, sqrt)
```

```
## [1]  9  5  6 11 10  2  8  4
```

Contoh 2 Misalnya, kita ingin menerapkan fungsi untuk mengubah semua string dalam suatu vector menjadi huruf kapital.

```r
# Bikin vektor yang berisi string
buah_buahan <- c("apel", "pisang", "jeruk", "mangga")
buah_buahan
```

```
## [1] "apel"   "pisang" "jeruk"  "mangga"
```

```r
# Kalo pake looping
# Membuat vektor kosong untuk menyimpan hasil
buah_buahan_kapital = buah_buahan

# Menggunakan for loop untuk mengubah setiap string menjadi huruf kapital
for (i in 1:length(buah_buahan)) {
  buah_buahan_kapital[i] <- toupper(buah_buahan[i])
}

# Menampilkan hasil
class(buah_buahan_kapital)
```

```
## [1] "character"
```

Penjelasan kalo pake looping:
- **for (i in 1:length(buah_buahan))**: Loop berjalan sebanyak panjang vektor, yang dihitung dengan fungsi length(). - **toupper(buah_buahan[i])**: Setiap elemen string diubah menjadi huruf kapital menggunakan fungsi toupper(). - **buah_buahan_kapital[i]** = . . . : Hasil dari fungsi toupper() disimpan kembali dalam vektor buah_buahan_kapital.

```r
# Kalo pake fungsi sapply()
sapply(buah_buahan, toupper)
```

```
##     apel   pisang    jeruk   mangga
##   "APEL" "PISANG"  "JERUK" "MANGGA"
```