Week\_03

230701195

2025-02-08

#Using if to Check a Number  
  
num <- -5  
if (num > 0) {  
 print("Positive")  
} else if (num < 0) {  
 print("Negative")  
} else {  
 print("Zero")  
}

## [1] "Negative"

#Using for Loop to Print a Sequence  
  
for (i in 1:10) {  
 print(i)  
}

## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 5  
## [1] 6  
## [1] 7  
## [1] 8  
## [1] 9  
## [1] 10

#Summing a Vector Using a for Loop  
  
vec <- c(1, 2, 3, 4, 5)  
sum <- 0  
for (val in vec) {  
 sum <- sum + val  
}  
print(sum)

## [1] 15

#Factorial Using while Loop  
  
num <- 5  
factorial <- 1  
while (num > 0) {  
 factorial <- factorial \* num  
 num <- num - 1  
}  
print(factorial)

## [1] 120

#Generate Fibonacci Series  
  
n <- 10  
fib <- numeric(n)  
fib[1] <- 0  
fib[2] <- 1  
for (i in 3:n) {  
 fib[i] <- fib[i-1] + fib[i-2]  
}  
print(fib)

## [1] 0 1 1 2 3 5 8 13 21 34

#Using repeat to Print a Countdown  
  
num <- 5  
repeat {  
 print(num)  
 num <- num - 1  
 if (num == 0) break  
}

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

#Using ifelse for Vectorized Conditions  
  
vec <- c(-2, 3, -1, 5, -7)  
result <- ifelse(vec < 0, 0, vec)  
print(result)

## [1] 0 3 0 5 0

#Nested for Loops for Matrix Multiplication  
  
A <- matrix(c(1, 2, 3, 4), nrow = 2)  
B <- matrix(c(2, 0, 1, 3), nrow = 2)  
C <- matrix(0, nrow = 2, ncol = 2)  
for (i in 1:2) {  
 for (j in 1:2) {  
 for (k in 1:2) {  
 C[i, j] <- C[i, j] + A[i, k] \* B[k, j]  
 }  
 }  
}  
print(C)

## [,1] [,2]  
## [1,] 2 10  
## [2,] 4 14

#Skipping Iterations Using next  
  
for (i in 1:10) {  
 if (i %% 2 == 0) next  
 print(i)  
}

## [1] 1  
## [1] 3  
## [1] 5  
## [1] 7  
## [1] 9

#Break Out of a Loop  
  
for (i in 1:10) {  
 if (i == 5) break  
 print(i)  
}

## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4

#Loop Over a List  
  
my\_list <- list(1, "R", TRUE, 3.14)  
for (item in my\_list) {  
 print(item)  
}

## [1] 1  
## [1] "R"  
## [1] TRUE  
## [1] 3.14

#Using apply for Matrix Operations  
  
mat <- matrix(1:9, nrow = 3)  
row\_sums <- apply(mat, 1, sum)  
print(row\_sums)

## [1] 12 15 18

#Generate Multiplication Table  
  
for (i in 1:5) {  
 for (j in 1:5) {  
 cat(i, "x", j, "=", i \* j, "\n")  
 }  
}

## 1 x 1 = 1   
## 1 x 2 = 2   
## 1 x 3 = 3   
## 1 x 4 = 4   
## 1 x 5 = 5   
## 2 x 1 = 2   
## 2 x 2 = 4   
## 2 x 3 = 6   
## 2 x 4 = 8   
## 2 x 5 = 10   
## 3 x 1 = 3   
## 3 x 2 = 6   
## 3 x 3 = 9   
## 3 x 4 = 12   
## 3 x 5 = 15   
## 4 x 1 = 4   
## 4 x 2 = 8   
## 4 x 3 = 12   
## 4 x 4 = 16   
## 4 x 5 = 20   
## 5 x 1 = 5   
## 5 x 2 = 10   
## 5 x 3 = 15   
## 5 x 4 = 20   
## 5 x 5 = 25

#Find Prime Numbers  
  
is\_prime <- function(n) {  
 if (n < 2) return(FALSE)  
 for (i in 2:sqrt(n)) {  
 if (n %% i == 0) return(FALSE)  
 }  
 return(TRUE)  
}  
for (i in 1:20) {  
 if (is\_prime(i)) print(i)  
}

## [1] 3  
## [1] 5  
## [1] 7  
## [1] 11  
## [1] 13  
## [1] 17  
## [1] 19

#Simulate Dice Rolls  
  
for (i in 1:10) {  
 roll <- sample(1:6, 1)  
 print(roll)  
}

## [1] 6  
## [1] 5  
## [1] 6  
## [1] 1  
## [1] 1  
## [1] 6  
## [1] 5  
## [1] 3  
## [1] 4  
## [1] 6

#Use while for Cumulative Sum  
  
sum <- 0  
i <- 1  
while (sum <= 50) {  
 sum <- sum + i  
 i <- i + 1  
}  
print(sum)

## [1] 55

#Filter Even Numbers Using if in a Loop  
  
nums <- 1:10  
evens <- c()  
for (num in nums) {  
 if (num %% 2 == 0) evens <- c(evens, num)  
}  
print(evens)

## [1] 2 4 6 8 10

#Compute Running Average  
  
vec <- c(10, 20, 30, 40, 50)  
running\_avg <- c()  
sum <- 0  
for (i in 1:length(vec)) {  
 sum <- sum + vec[i]  
 running\_avg <- c(running\_avg, sum / i)  
}  
print(running\_avg)

## [1] 10 15 20 25 30

#Using break in a Nested Loop  
  
found <- FALSE  
for (i in 1:5) {  
 for (j in 1:5) {  
 if (i \* j == 12) {  
 print(c(i, j))  
 found <- TRUE  
 break  
 }  
 }  
 if (found) break  
}

## [1] 3 4

#Use lapply to Apply a Function  
  
my\_list <- list(1, 2, 3, 4, 5)  
squared <- lapply(my\_list, function(x) x^2)  
print(squared)

## [[1]]  
## [1] 1  
##   
## [[2]]  
## [1] 4  
##   
## [[3]]  
## [1] 9  
##   
## [[4]]  
## [1] 16  
##   
## [[5]]  
## [1] 25