Coding Examples

1. Summing Numbers with for Loop

sum <- 0

for (i in 1:10) {

sum <- sum + i

}

print(sum)

output:

sum<-0

> for(i in 1:10)

+ {

+ sum<-sum+i

+ }

> print(sum)

[1] 55

2. Factorial Calculation with while Loop

num <- 5

factorial <- 1

while (num > 1) {

factorial <- factorial \* num

num <- num - 1

}

print(factorial)

output:

num<-5

> factorial<-1

> while(num>1)

+ {

+ factorial<-factorial\*num

+ num<-num-1

+ }

> print(factorial)

[1] 120

3. Finding Fibonacci Numbers with repeat Loop

fib <- numeric(10)

fib[1] <- 0

fib[2] <- 1

i <- 3

repeat {

fib[i] <- fib[i-1] + fib[i-2]

if (i == 10) {

break

}

i <- i + 1

}

print(fib)

output:

|  |
| --- |
| fib<-numeric(10)  > fib[1]<-0  > fib[2]<-1  > i<-3  > repeat{  + fib[i]<-fib[i-1]+fib[i-2]  + if(i==10)  + {  + break  + }  + i<-i+1  + }  > print(fib)  [1] 0 1 1 2 3 5 8 13 21 34 |
|  |
| |  | | --- | |  | |

Exercises

1. Exercise 1: Sum of Even Numbers

o Task: Write a for loop to calculate the sum of even numbers from 1 to 20.

o Expected Output:

sum\_even <- 0

for (i in 1:20) {

if (i %% 2 == 0) {

sum\_even <- sum\_even + i

}

}

print(sum\_even) # Output should be 110

output:

|  |
| --- |
| > sum\_even<-0  > for(i in 1:20)  + {  + if(i%%2==0)  + {  + sum\_even<-sum\_even +i  + }  + }  > print(sum\_even)  [1] 110 |
|  |
| |  | | --- | |  | |

2. Exercise 2: Prime Number Checker

o Task: Write a while loop to check if a given number is prime.

o Expected Output:

num <- 29

is\_prime <- TRUE

i <- 2

while (i <= sqrt(num)) {

if (num %% i == 0) {

is\_prime <- FALSE

break

}

i <- i + 1

}

if (is\_prime) {

print(paste(num, "is a prime number"))

} else {

print(paste(num, "is not a prime number"))

}

Output:

|  |
| --- |
| num<-29  > is\_prime<-TRUE  > i<-2  > while(i<=sqrt(num))  + {  + if(num%%i==0)  + {  + is\_prime<-FALSE  + break  + }  + i<-i+1  + }  > if(is\_prime)  + {  + print(paste(num,"is a prime number"))  + }else  + {  + print(paste(num,"is not a prime number"))  + }  [1] "29 is a prime number" |
|  |
| |  | | --- | | > | |

3. Exercise 3: Collatz Sequence

o Task: Use a repeat loop to generate the Collatz sequence for a given number. The

Collatz sequence is defined as follows: start with any positive integer n. Then

each term is obtained from the previous term as follows: if the previous term is

even, the next term is one half of the previous term. If the previous term is odd,

the next term is 3 times the previous term plus 1. The sequence ends when it

reaches 1.

o Expected Output:

num <- 13

repeat {

print(num)

if (num == 1) {

break

} else if (num %% 2 == 0) {

num <- num / 2

} else {

num <- 3 \* num + 1

}

}

Output:

|  |
| --- |
| num<-13  > repeat{  + print(num)  + if(num==1)  + {  + break  + }  + else if(num%%2==0)  + {  + num<-num/2  + }  + else  + {  + num<-3\*num+1  + }  + }  [1] 13  [1] 40  [1] 20  [1] 10  [1] 5  [1] 16  [1] 8  [1] 4  [1] 2  [1] 1 |

4. Exercise 4: Finding the Maximum Value in a Vector

o Task: Write a for loop to find the maximum value in a given numeric vector.

o Expected Output:

vec <- c(3, 5, 2, 8, 1, 9, 4)

max\_val <- vec[1]

for (i in vec) {

if (i > max\_val) {

max\_val <- i

}

}

print(max\_val) # Output should be 9

output:

|  |
| --- |
| vec<-c(3,5,2,8,1,9,4)  > max\_val<-vec[1]  > for(i in vec)  + {  + if(i>max\_val)  + {  + max\_val<-i  + }  + }  > print(max\_val)  [1] 9 |
|  |
| |  | | --- | | > | |