**Week 3 - Task 2: Recursion & Efficiency**

### **User-Generated Prompt for AI (Example)**

"I am working with a dataset that includes hierarchical relationships, such as a tree structure or nested categories. I need a Python script that:\*

* **Generates a recursive function to calculate the factorial of a given number.**
* **Creates a recursive function to compute Fibonacci numbers.**
* **Implements a recursive depth-first search (DFS) algorithm for traversing a tree structure.**
* **Ensures that the recursive functions include base cases for proper termination.** Can you generate an efficient Python script for these recursive tasks?"\*

### **AI-Generated Python Code for Task 2**

# Recursive function to calculate factorial

def factorial(n):

if n == 0 or n == 1:

return 1

return n \* factorial(n - 1)

# Recursive function to compute Fibonacci numbers

def fibonacci(n):

if n == 0:

return 0

elif n == 1:

return 1

return fibonacci(n - 1) + fibonacci(n - 2)

# Recursive Depth-First Search (DFS) for a tree structure

class TreeNode:

def \_\_init\_\_(self, value):

self.value = value

self.children = []

def add\_child(self, child):

self.children.append(child)

def dfs\_recursive(node, visited=None):

if visited is None:

visited = set()

if node.value in visited:

return

print(node.value)

visited.add(node.value)

for child in node.children:

dfs\_recursive(child, visited)

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

print("Factorial of 5:", factorial(5))

print("Fibonacci of 6:", fibonacci(6))

# Creating a sample tree

root = TreeNode("A")

nodeB = TreeNode("B")

nodeC = TreeNode("C")

nodeD = TreeNode("D")

root.add\_child(nodeB)

root.add\_child(nodeC)

nodeB.add\_child(nodeD)

print("DFS Traversal:")

dfs\_recursive(root)

**Week 3 - Task 2: Recursion & Efficiency (R Version)**

### **User-Generated Prompt for AI (Example)**

"I am working with a dataset that includes hierarchical relationships, such as a tree structure or nested categories. I need an R script that:\*

* **Generates a recursive function to calculate the factorial of a given number.**
* **Creates a recursive function to compute Fibonacci numbers.**
* **Implements a recursive depth-first search (DFS) algorithm for traversing a tree structure.**
* **Ensures that the recursive functions include base cases for proper termination.** Can you generate an efficient R script for these recursive tasks?"\*

### **AI-Generated R Code for Task 2**

# Recursive function to calculate factorial

factorial\_recursive <- function(n) {

if (n == 0 || n == 1) {

return(1)

}

return(n \* factorial\_recursive(n - 1))

}

# Recursive function to compute Fibonacci numbers

fibonacci\_recursive <- function(n) {

if (n == 0) {

return(0)

} else if (n == 1) {

return(1)

}

return(fibonacci\_recursive(n - 1) + fibonacci\_recursive(n - 2))

}

# Define a tree structure

TreeNode <- function(value) {

list(

value = value,

children = list(),

add\_child = function(self, child) {

self$children <- append(self$children, list(child))

}

)

}

# Recursive Depth-First Search (DFS) for a tree structure

dfs\_recursive <- function(node, visited = NULL) {

if (is.null(visited)) {

visited <- list()

}

if (node$value %in% visited) {

return()

}

print(node$value)

visited <- append(visited, node$value)

for (child in node$children) {

dfs\_recursive(child, visited)

}

}

# Example usage

print(paste("Factorial of 5:", factorial\_recursive(5)))

print(paste("Fibonacci of 6:", fibonacci\_recursive(6)))

# Creating a sample tree

root <- TreeNode("A")

nodeB <- TreeNode("B")

nodeC <- TreeNode("C")

nodeD <- TreeNode("D")

root$add\_child(root, nodeB)

root$add\_child(root, nodeC)

nodeB$add\_child(nodeB, nodeD)

print("DFS Traversal:")

dfs\_recursive(root)