1. A number is considered Disarium when the sum of its digits, each raised to the power of their respective positions, equals the number itself.

def is\_ disarium (number):

digits = str(number)

total = 0

for index, digit in enumerate(digits, start=1):

total += int(digit) \*\* index

return total == number

num = 175

if is\_ disarium(num):

print(f"{num} is a Disarium number.")

else:

print(f"{num} is not a Disarium number.")

Output :

175 is a Disarium number.

1. Write a function to find the length of the longest palindromic subsequence in a given string.

def longest\_ palindromic\_ subsequence(s):

n = len(s)

dp = [[0 for \_ in range(n)] for \_ in range(n)]

for i in range(n):

dp[i][i] = 1

for length in range(2, n + 1):

for i in range(n - length + 1):

j = i + length - 1

if s[i] == s[j]:

dp[i][j] = 2 + dp[i + 1][j - 1] if length > 2 else 2

else:

dp[i][j] = max(dp[i + 1][j], dp[i][j - 1])

return dp[0][n - 1]

s = "bbabcbcab"

length = longest\_ palindromic\_ subsequence(s)

print(f"Length of longest palindromic subsequence: {length}")

Output:

Length of longest palindromic subsequence: 7

1. Write a function to compare version numbers.

def compare\_ versions(version1, version2):

v1\_parts = list(map(int, version1.split('.')))

v2\_parts = list(map(int, version2.split('.')))

max\_len = max(len(v1\_parts), len(v2\_parts))

v1\_parts.extend([0] \* (max\_len - len(v1\_parts)))

v2\_parts.extend([0] \* (max\_len - len(v2\_parts)))

for a, b in zip(v1\_parts, v2\_parts):

if a > b:

return 1

elif a < b:

return -1

return 0

print(compare\_ versions("1.01", "1.001"))

Output: 0

1. Write a function to convert a decimal number lying between 1 and 3999 to Roman numerals.

def int\_ to\_ roman(num):

val = [

1000, 900, 500, 400,

100, 90, 50, 40,

10, 9, 5, 4,

1

]

syms = [

"M", "CM", "D", "CD",

"C", "XC", "L", "XL",

"X", "IX", "V", "IV",

"I"

]

roman = ""

i = 0

while num > 0:

count = num // val[i]

roman += syms[i] \* count

num -= val[i] \* count

i += 1

return roman

print(int\_ to\_ roman(3))

# Output: III