DAY–3

1.ARMSTRONG NUMBER

def is\_armstrong(n):

num\_str = str(n)

power = len(num\_str)

return n == sum(int(digit) \*\* power for digit in num\_str)

number = int(input("Enter a number: "))

if is\_armstrong(number):

print(f"{number} is an Armstrong number.")

else:

print(f"{number} is not an Armstrong number."

2.HAPPY NUMBER

def is\_happy(n):

def get\_sum\_of\_squares(x):

return sum(int(digit) \*\* 2 for digit in str(x))

seen = set()

while n != 1 and n not in seen:

seen.add(n)

n = get\_sum\_of\_squares(n)

return n == 1

number = int(input("Enter a number: "))

if is\_happy(number):

print(f"{number} is a happy number.")

else:

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

3.SIMPLE INTEREST

def simple\_interest(principal, rate, time):

return (principal \* rate \* time) / 100

p = float(input("Enter principal amount: "))

r = float(input("Enter annual interest rate (in %): "))

t = float(input("Enter time (in years): "))

interest = simple\_interest(p, r, t)

print(f"Simple Interest: {interest}")

4.FACTORIAL AND PRINT 1ST N FACTOR

def factors(n, count):

fact\_list = [i for i in range(1, n + 1) if n % i == 0]

return fact\_list[:count]

number = int(input("En5.SQUARE AND CUBE

def square\_and\_cube(n):

return n \*\* 2, n \*\* 3

number = float(input("Enter a decimal number: "))

square, cube = square\_and\_cube(number)

print(f"Square: {square}")

print(f"Cube: {cube}")

ter a number: "))

n\_factors = int(input("Enter the number of factors to display: "))

fact\_list = factors(number, n\_factors)

print(f"First {n\_factors} factors of {number}: {fact\_list}")

5.SQUARE AND CUBE

def square\_and\_cube(n):

return n \*\* 2, n \*\* 3

number = float(input("Enter a decimal number: "))

square, cube = square\_and\_cube(number)

print(f"Square: {square}")

print(f"Cube: {cube}")

6.a)BINARY TO DECIMAL

def binary\_to\_decimal(binary\_str):

return int(binary\_str, 2)

binary\_str = input("Enter a binary number: ")

decimal\_value = binary\_to\_decimal(binary\_str)

print(f"Decimal value: {decimal\_value}")

b)BINARY TO OCTAL

def binary\_to\_octal(binary\_str):

decimal\_value = int(binary\_str, 2)

return oct(decimal\_value)[2:]

binary\_str = input("Enter a binary number: ")

octal\_value = binary\_to\_octal(binary\_str)

print(f"Octal value: {octal\_value}")

7.ADD TWO BINARY STRING

def add\_binary(a, b):

return bin(int(a, 2) + int(b, 2))[2:]

a = "11"

b = "1"

result = add\_binary(a, b)

print(f"Output: {result}")

8.GREATEST OF GIVEN 3

def binary\_to\_decimal(binary\_str):

return int(binary\_str, 2)

def greatest\_binary(b1, b2, b3):

dec1, dec2, dec3 = map(binary\_to\_decimal, [b1, b2, b3])

return max(dec1, dec2, dec3)

b1 = input("Enter the first binary number: ")

b2 = input("Enter the second binary number: ")

b3 = input("Enter the third binary number: ")

greatest = greatest\_binary(b1, b2, b3)

print(f"Greatest number in decimal: {greatest}")

9.MATRIX MULTIPLICATION

def matrix\_multiply(A, B):

return [[sum(x \* y for x, y in zip(A\_row, B\_col)) for B\_col in zip(\*B)] for A\_row in A]

A = [[1, 2], [3, 4]]

B = [[5, 6], [7, 8]]

result = matrix\_multiply(A, B)

print("Product of matrices:")

for row in result:

print(row)

10.MATRIX ADDITION

def add\_matrices(A, B):

return [[A[i][j] + B[i][j] for j in range(len(A[0]))] for i in range(len(A))]

A = [[1, 2, 3], [4, 5, 6]]

B = [[7, 8, 9], [10, 11, 12]]

result = add\_matrices(A, B)

print("Sum of matrices:")

for row in result:

print(row)