Python Programming

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 First '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("Enter 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)
```