STUDENT NAME	**A.PAVANI**	
STUDENT REGISTRATION NUMBER		**CLASS: AIML(B)**
PROGRAM	**UG**	**YEAR and TERM:** **1st year & 1st term**
SUBJECT NAME	**PROBLEM SOLVING WITH PYTHON.**	
NAME OF THE ASSESSMENT	**CONCEPT MAPPING**	
DATE OF SUBMISSION	**18-10-2025**	

CONCEPT MAPPING ON LOOPS

- SUMMATION AND FACTORIAL
  → CALCULATE SERIES, FACTORIALS, POWERS
- PRIME NUMBER CHECKING
  → LOOP THROUGH DIVISORS
- FIBONACCI SERIES GENERATION
  → REPETITIVE CALCULATION OF NEXT TERM
- PATTERN PRINTING
  → GENERATE PYRAMIDS, DIAMONDS, AND GRIDS

1. Summation, Factorials, and Powers.

Summation Logic:

- Problem: Sum numbers from 1 to N or a series like 1 + 2 + 3 + ... + N.
- Logic:

1. Initialize sum = 0.
2. Loop i from 1 to N.
3. Add i to sum.
4. Print sum.

Formula: sum = 1 + 2 + ... + N = N*(N+1)/2

Factorial Logic

- Problem: Calculate N! = 1 × 2 × 3 × ... × N.
- Logic:

1. Initialize fact = 1.
2. Loop i from 1 to N.
3. Multiply fact = fact * i.
4. Print fact.

Power Calculation

- Problem: Calculate A^B.
- Logic:
  1. Initialize result = 1.
  2. Loop i from 1 to B.
  3. Multiply result = result * A.
  4. Print result.

EX:

- Summation of first n numbers

```
def summation(n):
  sum = 0
  for i in range(1, n + 1):
    sum += i
  return sum
```

- Factorial of a number

```
def factorial(n):
  fact = 1
  for i in range(1, n + 1):
    fact *= i
  return fact
```

- Power calculation: base^exponent

```
def power(base, exponent):
  result = 1
```

```
    for i in range(exponent):
        result *= base
    return result
```

➢ Main program

```
num = int(input("Enter a number: "))
print(f" Summation of first {num} numbers is: {summation(num)}")
print(f" Factorial of {num} is: {factorial(num)}")
base = int(input("Enter base for power calculation: "))
exp = int(input("Enter exponent: "))
print(f"{base}^{exp} = {power(base, exp)}")
```

2. Prime Number Checking

- Problem: Check if a number N is prime.
- Logic:

1. If N <= 1 → not prime.
2. Loop i from 2 to sqrt(N).
3. If N % i == 0, it's not prime.
4. Otherwise, it's prime.

Reason: Divisors beyond sqrt(N) repeat.

EX:

➢ Function to check prime

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
```

➢ Main program

```
num = int(input("Enter a number: "))
if is_ prime (num):
  print(f"{num} is a prime number.")
else:
  print(f"{num} is not a prime number.")
```

3. Fibonacci Series Generation

- Problem: Generate first N Fibonacci numbers: 0, 1, 1, 2, 3, 5, ...
- Logic:

1. Initialize a = 0, b = 1.
2. Loop from 1 to N:

   Print a.

   next = a + b.

   Update a = b, b = next.

Alternative: Recursive calculation, but iterative is faster.

EX:

- Function to generate Fibonacci series up to n terms

```
def Fibonacci _series(n):
  a, b = 0, 1
  series = []
  for _ in range(n):
    series. append(a)
    a, b = b, a + b
  return series
```

- Main program

```
num = int(input("Enter the number of terms: "))
fib_ series = Fibonacci _series(num)
print(f" Fibonacci series up to {num} terms: {fib_ series}")
```

4. Pattern Printing

a) Pyramid Pattern

Example for N = 5:

```
    *
   ***
  *****
 *******
*********
```

Logic:

1. Loop i from 1 to N (rows).
2. Print (N-i) spaces.
3. Print (2*i-1) stars.

---

b) Diamond Pattern

Example for N = 3:

```
  *
 ***
*****
 ***
  *
```

Logic:

1. Print upper pyramid of height N.
2. Print inverted pyramid of height N-1.

---

c) Grid Pattern

Example for N = 3:

```
* * *
* * *
* * *
```

Logic:

1. Loop i from 1 to N.
2. Loop j from 1 to N.
3. Print * in inner loop.
4. Print newline after inner loop.