**EXPERIMENT : 02**

**Python list comprehension with examples**

Python list comprehensions provide a concise way to create lists. They are often used as a more readable and expressive alternative to traditional loops. Here's a breakdown of Python list comprehensions with examples:

### **Basic List Comprehension Syntax:**

new\_list = [expression for item in iterable if condition]

**Example 1: Squaring Numbers**

numbers = [1, 2, 3, 4, 5]

squared = [num \*\* 2 for num in numbers]

print(squared)

# Output: [1, 4, 9, 16, 25]

**Example 2: Filtering Even Numbers**

numbers = [1, 2, 3, 4, 5]

even\_numbers = [num for num in numbers if num % 2 == 0]

print(even\_numbers)

# Output: [2, 4]

**Example 3: Creating a List of Tuples**

coordinates = [(x, y) for x in range(3) for y in range(2)]

print(coordinates)

# Output: [(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)]

**Example 4: Flattening a 2D List**

matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

flattened = [item for row in matrix for item in row]

print(flattened)

# Output: [1, 2, 3, 4, 5, 6, 7, 8, 9]

**Example 5: Converting to UpperCase**

words = ['hello', 'world', 'python']

upper\_case\_words = [word.upper() for word in words]

print(upper\_case\_words)

# Output: ['HELLO', 'WORLD', 'PYTHON']

**Example 6: Creating a Dictionary from Lists**

keys = ['a', 'b', 'c']

values = [1, 2, 3]

dictionary = {key: value for key, value in zip(keys, values)}

print(dictionary)

# Output: {'a': 1, 'b': 2, 'c': 3}