



Shri Vile Parle Kelavani Mandal's

# INSTITUTE OF TECHNOLOGY

## DHULE (M.S.)

### DEPARTMENT OF COMPUTER ENGINEERING

Subject : Cloud Computing Lab	Remark
Name : Mohammad Anas Aarif Baig Mirza	Roll No. : 40
Class : B.tech Final Year	Batch : B2
Expt. No. :07	Division: -
Date : 15/10/2025	Signature
Title : Demonstrate the use of map and reduce tasks.	

#### Aim:

To demonstrate the use of **map()** and **reduce()** functions in Python for performing data transformation and aggregation tasks.

#### Software / Tools Used:

- **Software:** Python (IDLE / VS Code / Jupyter Notebook / any IDE)
- **Version:** Python 3.8 or above
- **Operating System:** Windows / Linux / macOS
- **Theory:**

The **map()** and **reduce()** functions are part of Python's functional programming features. They help in applying operations over data collections efficiently.

1. **map(function, iterable)**
  - Applies a function to each element of the iterable.
  - Returns a map object (can be converted into a list).
  - Example:
    - `list(map(lambda x: x*x, [1,2,3]))`
    - $\rightarrow [2, 4, 6]$
2. **reduce(function, iterable)**
  - Performs a cumulative operation on all items in an iterable to reduce it to a single value.
  - Imported from the **functools** module.
  - Example:
    - `from functools import reduce`
    - `reduce(lambda a,b: a+b, [1,2,3,4])`
    - $\rightarrow 10$

Together, these mimic the **MapReduce** concept used in Big Data, where:

- **Map step:** transforms or filters data.

- **Reduce step:** aggregates results.
- 

### **Algorithm:**

1. Start the program.
2. Import the **reduce** function from the **functools** module.
3. Initialize a list of numbers.
4. Use **map()** to:
  - o Double each number.
  - o Convert each number to string.
5. Use **reduce()** to:
  - o Calculate the sum of all numbers.
  - o Calculate the product of all numbers.
6. Display all the results.
7. Stop the program.

### **Program:**

```
from functools import reduce

numbers = [5, 10, 15, 20, 25, 30]

def map_task(num):
    return num * 2

def reduce_task(a, b):
    return a + b

mapped = list(map(map_task, numbers))
total = reduce(reduce_task, mapped)

print("Original numbers:", numbers)
print("Mapped (doubled) numbers:", mapped)
print("Reduced (sum of doubled numbers):", total)
```

### **Output:**

After map (doubled): [4, 8, 12, 16, 20]  
 Numbers as strings: ['2', '4', '6', '8', '10']  
 Sum of all numbers: 30  
 Product of all numbers: 3840

The screenshot shows a Python development environment with a dark theme. The top bar includes standard icons for file operations (New, Open, Save), running the program, debugging, stopping, sharing, saving, and beautifying code. The main window displays a file named 'main.py' containing the following code:

```
1 from functools import reduce
2
3 numbers = [5, 10, 15, 20, 25, 30]
4
5 def map_task(num):
6     return num * 2
7
8 def reduce_task(a, b):
9     return a + b
10
11 mapped = list(map(map_task, numbers))
12 total = reduce(reduce_task, mapped)
13
14 print("Original numbers:", numbers)
15 print("Mapped (doubled) numbers:", mapped)
16 print("Reduced (sum of doubled numbers):", total)
17
```

The bottom panel shows the output of the program's execution:

```
Original numbers: [5, 10, 15, 20, 25, 30]
Mapped (doubled) numbers: [10, 20, 30, 40, 50, 60]
Reduced (sum of doubled numbers): 210

...Program finished with exit code 0
Press ENTER to exit console.
```

## Conclusion:

The practical demonstrates how the `map()` function performs element-wise operations, while `reduce()` performs cumulative aggregation.

These are powerful tools for functional programming and are conceptually similar to the **MapReduce model** used in distributed data processing systems like Hadoop