PES UNIVERSITY, BENGALURU

Perseverance | Excellence | Service

UE25CS151A – PYTHON FOR COMPUTATIONAL PROBLEM SOLVING LAB MANUAL

WEEK 13

TOPICS:

Creation of User Defined Modules and its usages.

OBEJCTIVE:

The objective of this lab is to enable students to:

1. **Understand the concept of user-defined modules in Python** and how modular programming improves code organization, reusability, and maintainability.
2. **Create Python modules** containing functions and logic that can be reused across multiple programs.
3. **Import and use modules** correctly using different import styles (import module, from module import function).
4. **Apply modular design principles** to solve normal and algorithmic (Leet Code-style) problems.
5. **Develop problem-solving skills** by implementing clean, structured solutions across multiple files while maintaining separation of logic.

Problem Statement 1:

Create a module **math_utils.py** that contains the following functions:

- add(a, b) – returns a + b
- subtract(a, b) – returns a - b
- multiply(a, b) – returns a * b
- divide(a, b) – returns a / b (assume b is not zero for this lab)

Write a main program q1_main.py that:

1. Imports the functions from math_utils
2. Reads two integers and an operator (+, -, *, /) from the user
3. Uses the appropriate function from the module
4. Prints the result

Expected Output:

Enter first number: 45
Enter second number: 90
Enter operator (+, -, *, /): +
Result: 135

Enter first number: 87
Enter second number: 32
Enter operator (+, -, *, /): /
Result: 2.71875

Problem Statement 2:

Create a module **freq_utils.py** that contains:

char_frequency(s) – returns a dictionary with each character and its frequency
most_frequent_char(s) – returns the character that occurs maximum times

Write a program **q2_main.py** that:

- Reads a string
- Uses the module functions to display character frequency and most frequent character

Expected Output:

Enter a string: engineering
Frequencies: {'e': 3, 'n': 2, 'g': 2, 'i': 2, 'r': 1}
Most frequent character: e

Problem Statement 3:

Create a module **subject_utils.py** with:

- subject_mean(subject_marks) – returns average score using NumPy
- above_average(subject_marks) – returns a list of marks above mean

Write **q3_main.py** to:

1. Read marks of n students in one subject
2. Display mean
3. Display all marks above mean

Expected Output:

Enter marks: 50 60 70 80 90
Mean: 70.0
Above average: 80 90

Problem Statement 4:

Create a module **pair_utils.py** with a function:

- count_pairs(nums) – counts how many pairs (i, j) exist such that:
  - i < j
  - nums[i] < nums[j]

Write a main file **q4_main.py** that reads a list and prints the count.

Expected Output:
Input: 4 1 5 2 6
Output: 6
(Pairs: (4,5), (4,6), (1,5), (1,2), (1,6), (5,6)). So Output is 6

Practice Problems:

1. Write a module rotate_utils.py containing:
   - rotate_right(nums, k) – rotates the list right by k positions
     Example:

   nums = [1,2,3,4,5], k=2 → [4,5,1,2,3]

Write q5_main.py that:

1. Reads a list
2. Reads k
3. Calls rotate_right
4. Prints rotated list

Expected Output:

Enter numbers: 10 20 30 40 50
Enter k: 3
Rotated List: 30 40 50 10 20

2. Create a module named **cipher_module.py** with a function caesar_encrypt(text, shift). This function implements a basic Caesar cipher (shift cipher). It takes a plaintext string ***text*** and an integer ***shift*** value. It should return a new string where every alphabetical character in the original string is shifted forward by the given ***shift*** amount.

Constraints:

- The shift should wrap around the alphabet
  - (e.g., 'z' shifted by 1 becomes 'a').
- Case sensitivity must be preserved

    - (e.g., 'A' shifted by 1 becomes 'B', not 'b').
- Non-alphabetical characters (spaces, numbers, punctuation) should remain unchanged.
- Assume ***shift*** is a non-negative integer.

Example:

- Input: text = "Hello Z",
- shift = 1
- Output: "Ifmmp A"

The best code is not the one that works, but the one that's easy to understand.