**10211CS213- Python Programming**

**Task 4: Use various data types, List, Tuples and Dictionary in python programming**

**4.1** Given a matrix mat where every row is sorted in **strictly** **increasing** order, return the **smallest common element** in all rows.  If there is no common element, return -1.

**Aim**

To write a Python program that takes a sorted matrix (each row sorted in strictly increasing order) as input from the user and finds the smallest element that is present in all rows of the matrix. If no such element exists, the program should return -1.

Algorithm

1. Start
2. Input the number of rows (rows) and columns (cols) from the user.
3. Initialize an empty matrix mat.
4. For each row from 0 to rows - 1:
   * Read the row values from the user as space-separated integers.
   * Append the row to mat.
5. Define a function smallestCommonElement(mat) to:
   * Let rows = total number of rows, cols = total number of columns.
   * For each element num in the first row:
     1. Assume found = True.
     2. For each other row (starting from the 2nd row):
        + Perform binary search for num in that row:
          - Set left = 0, right = cols - 1.
          - While left <= right:

Find mid = (left + right) // 2.

If mat[row][mid] == num, break (element found in this row).

If mat[row][mid] < num, move to the right half (left = mid + 1).

Else, move to the left half (right = mid - 1).

* + - * If binary search finishes without finding the element, set found = False and break.
    1. If found is still True after checking all rows, return num (smallest common element).
  + If no element is found in all rows, return -1.

1. Call the function with mat as input.
2. Print the result.
3. Stop.

**Program**

def smallestCommonElement(mat):

rows = len(mat)

cols = len(mat[0])

# Check each number in first row

for num in mat[0]:

found = True

for r in range(1, rows):

# Binary search in each row

left, right = 0, cols - 1

while left <= right:

mid = (left + right) // 2

if mat[r][mid] == num:

break

elif mat[r][mid] < num:

left = mid + 1

else:

right = mid - 1

else: # executed if while loop doesn't break

found = False

break

if found:

return num

return -1

# --- Get matrix input from user ---

rows = int(input("Enter number of rows: "))

cols = int(input("Enter number of columns: "))

mat = []

print("Enter matrix values row by row (sorted in increasing order):")

for \_ in range(rows):

row = list(map(int, input().split()))

mat.append(row)

# Find and display smallest common element

result = smallestCommonElement(mat)

print("Smallest common element in all rows:", result)

Example 1:

Input:

4 5

1 2 3 4 5

2 4 5 8 10

3 5 7 9 11

1 3 5 7 9

 Output:

5

**4.2** Given an integer n, return an list of length n + 1 such that for each i (0 <= i <= n), ans[i] is the number of 1's in the binary representation of i.

**Aim**

To read a non-negative integer n from the user and generate a list of length n + 1 where each element contains the count of 1s in the binary representation of its index.

**Algorithm**

1. Start the program.
2. Read the integer n from the user.
3. Initialize an empty list ans.
4. Repeat for each integer i from 0 to n:
   * Convert i into its binary representation using bin(i).
   * Count the number of 1s in the binary string using .count('1').
   * Append this count to the list ans.
5. Display the list ans as the result.
6. End the program.

**Program**

def countBits(n):

ans = []

for i in range(n + 1):

ans.append(bin(i).count('1'))

return ans

# Get input from user

n = int(input("Enter a non-negative integer: "))

result = countBits(n)

print("Count of 1's for numbers from 0 to", n, ":", result)

**Example**

Enter a non-negative integer: 5

Count of 1's for numbers from 0 to 5 : [0, 1, 1, 2, 1, 2]

**4.3** You are given an integer tuple nums containing distinct numbers. Your task is to perform a sequence of operations on this tuple until it becomes empty. The operations are defined as follows:

1. If the first element of the tuple has the smallest value in the entire tuple, remove it.
2. Otherwise, move the first element to the end of the tuple.

You need to return an integer denoting the number of operations required to make the tuple empty.

### **Constraints**

* The input tuple nums contains distinct integers.
* The operations must be performed using tuples and sets to maintain immutability and efficiency.
* Your function should accept the tuple nums as input and return the total number of operations as an integer.

**Aim**

To simulate the given operations on a tuple of distinct integers and count the number of steps needed until the tuple is empty.

**Algorithm**

1. Start the program.
2. Read the tuple nums from the user (or predefined).
3. Initialize a counter operations = 0.
4. While nums is not empty:
   * If nums[0] is the minimum element in nums:
     + Remove it (nums = nums[1:]).
   * Else:
     + Move nums[0] to the end of the tuple.
   * Increment operations by 1.
5. Output the value of operations.

**Program**

def count\_operations(nums):

nums = list(nums) # Convert tuple to list for easier modification

operations = 0

while nums:

if nums[0] == min(nums):

nums.pop(0) # Remove first element

else:

nums.append(nums.pop(0)) # Move first element to end

operations += 1

return operations

# Get input from user

nums = tuple(map(int, input("Enter distinct integers separated by space: ").split()))

print("Number of operations:", count\_operations(nums))

**Example**

Enter distinct integers separated by space: 3 1 2

Number of operations: 5

**4.4** Given an array of integers nums containing n + 1 integers where each integer is in the range [1, n] inclusive. There is only **one repeated number** in nums, return *this repeated number*. Solve the problem using set.

**Aim**

To find the single repeated integer in a list where integers are in the range [1, n] and the list has n + 1 elements, using a set for efficient lookup.

**Algorithm**

1. Start the program.
2. Read the list nums from the user.
3. Create an empty set seen.
4. Iterate through each element num in nums:
   * If num is already in seen, return num as the duplicate.
   * Otherwise, add num to seen.
5. End.

**Program**

def find\_duplicate(nums):

seen = set()

for num in nums:

if num in seen:

return num

seen.add(num)

# Get input from user

nums = list(map(int, input("Enter numbers separated by space: ").split()))

duplicate = find\_duplicate(nums)

print("The repeated number is:", duplicate)

**Example**

Enter numbers separated by space: 3 1 3 4 2

The repeated number is: 3

4.5 . Create a student dictionary  for n students with the student name as key and their test mark assignment mark and lab mark as values. Do the following computations and display the result.

1.Identify the student with the  highest average score

2.Identify the student who as the highest Assignment marks

3.Identify the student with the Lowest lab marks

4.Identify the student with the lowest average score

Note:

If more than one student has the same score display all the student names

**Aim**

To write a Python program that stores student details in a dictionary with the student name as the key and their **test mark**, **assignment mark**, and **lab mark** as values. The program will:

1. Identify the student(s) with the **highest average score**.
2. Identify the student(s) with the **highest assignment marks**.
3. Identify the student(s) with the **lowest lab marks**.
4. Identify the student(s) with the **lowest average score**.  
   If more than one student has the same score, all such student names will be displayed.

**Algorithm**

1. **Start**
2. Read the number of students n.
3. Initialize an empty dictionary students.
4. For each student from 1 to n:
   * Read the student's name.
   * Read the **test mark**, **assignment mark**, and **lab mark**.
   * Store the values as a list in the dictionary with the student name as the key.
5. Define a function average(marks) that returns the average of the given list of marks.
6. **Find the highest average score**:
   * Calculate the average for each student.
   * Find the maximum average (max\_avg).
   * Create a list of all students whose average equals max\_avg.
7. **Find the highest assignment marks**:
   * Get the assignment mark (index 1 of the list) for each student.
   * Find the maximum assignment mark (max\_assignment).
   * Create a list of all students whose assignment mark equals max\_assignment.
8. **Find the lowest lab marks**:
   * Get the lab mark (index 2 of the list) for each student.
   * Find the minimum lab mark (min\_lab).
   * Create a list of all students whose lab mark equals min\_lab.
9. **Find the lowest average score**:
   * Calculate the average for each student.
   * Find the minimum average (min\_avg).
   * Create a list of all students whose average equals min\_avg.
10. **Display**:
    * Highest average score and corresponding student(s).
    * Highest assignment marks and corresponding student(s).
    * Lowest lab marks and corresponding student(s).
    * Lowest average score and corresponding student(s).
11. **Stop**

students = {}

n = int(input("Enter number of students: "))

for \_ in range(n):

name = input("\nEnter student name: ")

test = float(input("Enter Test mark: "))

assignment = float(input("Enter Assignment mark: "))

lab = float(input("Enter Lab mark: "))

students[name] = [test, assignment, lab]

# Function to calculate average

def average(marks):

return sum(marks) / len(marks)

# 1. Highest average score

max\_avg = max(average(marks) for marks in students.values())

highest\_avg\_students = [name for name, marks in students.items() if average(marks) == max\_avg]

# 2. Highest assignment marks

max\_assignment = max(marks[1] for marks in students.values())

highest\_assignment\_students = [name for name, marks in students.items() if marks[1] == max\_assignment]

# 3. Lowest lab marks

min\_lab = min(marks[2] for marks in students.values())

lowest\_lab\_students = [name for name, marks in students.items() if marks[2] == min\_lab]

# 4. Lowest average score

min\_avg = min(average(marks) for marks in students.values())

lowest\_avg\_students = [name for name, marks in students.items() if average(marks) == min\_avg]

# Display results

print("\n--- Results ---")

print("Highest average score:", highest\_avg\_students, "with average =", max\_avg)

print("Highest assignment marks:", highest\_assignment\_students, "with marks =", max\_assignment)

print("Lowest lab marks:", lowest\_lab\_students, "with marks =", min\_lab)

print("Lowest average score:", lowest\_avg\_students, "with average =", min\_avg)

Example

Enter number of students: 3

Enter student name: Alice

Enter Test mark: 85

Enter Assignment mark: 90

Enter Lab mark: 80

Enter student name: Bob

Enter Test mark: 78

Enter Assignment mark: 92

Enter Lab mark: 70

Enter student name: Charlie

Enter Test mark: 85

Enter Assignment mark: 85

Enter Lab mark: 85

**Output**-

Highest average score: ['Charlie'] with average = 85.0

Highest assignment marks: ['Bob'] with marks = 92.0

Lowest lab marks: ['Bob'] with marks = 70.0

Lowest average score: ['Bob'] with average = 80.0