**Test Series 1**

1. What happens if you enter "abc" as input?

A) "Invalid input"

B) "You entered: abc"

C) No output, the program waits for valid input

D) None of the above

**try:**

**num = int(input("Enter a number: "))**

**print("You entered:", num)**

**except ValueError:**

**print("Invalid input")**

1. What is the output if you enter "-3" as input?

A) "Error: Not a positive integer"

B) No output

C) "Error: -3 is not a positive integer"

D) None of the above

**try:**

**num = int(input("Enter a positive integer: "))**

**if num <= 0:**

**raise ValueError("Not a positive integer")**

**except ValueError as e:**

**print("Error:", str(e))**

1. What is the output if you enter "0" as input?

A) "Result: Infinity"

B) "Result: 0"

C) "Result: 10.0"

D) None of the above

**try:**

**num = int(input("Enter a number: "))**

**result = 10 / num**

**except ValueError:**

**result = 0**

**except ZeroDivisionError:**

**result = "Infinity"**

**finally:**

**print("Result:", result)**

1. What is the output of the following code?

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

**squared\_values = [x \*\* 2 for x in numbers]**

**print(squared\_values)**

A) [1, 2, 3, 4, 5]

B) [1, 4, 9, 16, 25]

C) [2, 4, 6, 8, 10]

D) [1, 2, 3, 4, 5, 6]

1. What is the output of the following code?

**my\_dict = {'a': 1, 'b': 2, 'c': 3}**

**del my\_dict['b']**

**print(len(my\_dict))**

A) 2

B) 3

C) 1

D) 0

1. What is the output of this code?

**a = []**

**b = []**

**for i in range(1, 11):**

**if i % 2:**

**a.append(i)**

**for i in range(1, 11):**

**if i % 4:**

**b.append(i)**

**a = set(a)**

**b = set(b)**

**a.update(b)**

**print(a)**

A) {1, 2, 3, 5, 6, 7, 9, 10}

B) {1, 3, 5, 7, 9}

C) {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

D) {1, 4, 7}

1. What is the output of this code?

**value = 10**

**result = value \* 2 if value > 5 else value / 2**

**print(result)**

A) 5.0

B) 20

C) 10

D) 2.0

1. What is the output of this code?

**text = "Hello, World!"**

**print(text[::-1])**

A) "!dlroW ,olleH"

B) "Hello, World!"

C) "World, Hello!"

D) "!World ,Hello"

1. What is the output of this code?

A) 1

B) 3

C) 4

D) 5

**def my\_generator():**

**yield 1**

**yield 2**

**yield 3**

**gen = my\_generator()**

**print(next(gen) + next(gen))**

1. What is the output of this code?

A) 1.0

B) -1.0

C) 0.0

D) ValueError

**import math**

**result = math.sqrt(-1)**

**print(result)**

1. What is the output of this code?

A) 10

B) 15

C) 5

D) NameError

**def outer\_func():**

**x = 10**

**def inner\_func():**

**nonlocal x**

**x += 5**

**inner\_func()**

**return x**

**result = outer\_func()**

**print(result)**

1. What is the output of this code?

A) 30

B) TypeError

C) 10

D) 20

**class A:**

**def \_\_init\_\_(self, value):**

**self.value = value**

**a = A(10)**

**b = A(20)**

**print(a + b)**

1. What is the output of this code?

A) 5.0 Done

B) 5 Done

C) ZeroDivisionError Done

D) 10 Done

**def divide(x, y):**

**try:**

**result = x / y**

**except ZeroDivisionError as e:**

**result = str(e)**

**finally:**

**result += " Done"**

**return result**

**result = divide(10, 2)**

**print(result)**

1. What is the value of **y** printed by this code?

A) 0

B) 5

C) -5

D) None

**x = 5**

**y = x if x > 0 else 0**

**x = 0**

**print(y)**

1. What is the output of this code?

A) [1, 2, 3, 4]

B) [1, 2, 3, 5]

C) [2, 3, 5]

D) [1, 2, 3]

**my\_list = [1, 2, 3, 4]**

**new\_list = my\_list[1:3]**

**new\_list.append(5)**

**print(my\_list)**

1. What is the output of this code?

A) [1, 3, 5]

B) [2, 4]

C) [1, 2, 3, 4, 5]

D) [2, 4, 5]

**def is\_even(num):**

**return num % 2 == 0**

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

**even\_numbers = list(filter(is\_even, numbers))**

**print(even\_numbers)**

**ge="Hello"):**

**print(message, name)**

**greet("Alice")**

1. What is the output of this code?

A) 14

B) 11

C) 24

D) 8

**def add(a, b):**

**return a + b**

**def multiply(a, b):**

**return a \* b**

**result = add(3, multiply(2, 4))**

**print(result)**

1. What is the output of this code?

A) 5

B) 20

C) 120

D) 24

**def factorial(n):**

**if n == 0:**

**return 1**

**return n \* factorial(n - 1)**

**result = factorial(5)**

**print(result)**

1. What is the output of this code?

A) 2

B) 3

C) 4

D) 1

**my\_set = {1, 2, 3}**

**my\_set.add(4)**

**my\_set.remove(2)**

**print(len(my\_set))**

1. What is the output of this code?

A) True

B) False

C) Error

D) "racecar"

**def is\_palindrome(s):**

**return s == s[::-1]**

**result = is\_palindrome("racecar")**

**print(result)**

1. What is the output of this code?

A) "Hello, Alice"

B) "HELLO, ALICE"

C) "Hello, ALICE"

D) "hello, Alice"

**def uppercase\_decorator(func):**

**def inside(\*args, \*\*kwargs):**

**result = func(\*args, \*\*kwargs)**

**return result.upper()**

**return inside**

**@uppercase\_decorator**

**def greet(name):**

**return f"Hello, {name}"**

**print(greet("Alice"))**

1. What is the output of this code?

A) "Time taken: 2.0 seconds"

B) "Time taken: 0.0 seconds"

C) "Time taken: 1.0 seconds"

D) No output, program execution is delayed by 2 seconds

**def performance\_timer(func):**

**import time**

**def inside(\*args, \*\*kwargs):**

**start\_time = time.time()**

**result = func(\*args, \*\*kwargs)**

**end\_time = time.time()**

**print(f"Time taken: {end\_time - start\_time} seconds")**

**return result**

**return inside**

**@performance\_timer**

**def slow\_function():**

**time.sleep(2)**

**slow\_function()**

1. What is the output of this code?

A) "Arguments: 3, 4, Result: 7"

B) "Arguments: '3', '4', Result: 7"

C) "Arguments: 3, 4, Result: '7'"

D) Error, decorators cannot be applied to functions with arguments

**def log\_args\_and\_result(func):**

**def inside(\*args, \*\*kwargs):**

**arg\_str = ", ".join(map(repr, args))**

**result = func(\*args, \*\*kwargs)**

**print(f"Arguments: {arg\_str}, Result: {result}")**

**return result**

**return inside**

**@log\_args\_and\_result**

**def add(a, b):**

**return a + b**

**result = add(3, 4)**

1. What is the value of **result**?

A) 10

B) 55

C) 89

D) 144

**def memoize(func):**

**cache = {}**

**def inside(\*args):**

**if args not in cache:**

**cache[args] = func(\*args)**

**return cache[args]**

**return inside**

**@memoize**

**def fibonacci(n):**

**if n <= 1:**

**return n**

**else:**

**return fibonacci(n - 1) + fibonacci(n - 2)**

**result = fibonacci(10)**

1. What is the output of this code?

A) "Admin panel accessed."

B) "Access Denied"

C) "admin"

D) "admin panel accessed. Access Denied"

**def access\_control(required\_role):**

**def decorator(func):**

**def inside(user\_role):**

**if user\_role == required\_role:**

**return func(user\_role)**

**else:**

**return "Access Denied"**

**return inside**

**return decorator**

**@access\_control("admin")**

**def admin\_panel(user\_role):**

**return "Admin panel accessed."**

**result = admin\_panel("admin")**

1. What is the output of this code?

A) "Operation successful"

B) "Attempt 1: Operation failed"

C) "Attempt 1: Operation failed\n

Attempt 2: Operation failed"

D) "Max attempts reached"

**def retry(max\_attempts):**

**def decorator(func):**

**def inside(\*args, \*\*kwargs):**

**attempts = 0**

**while attempts < max\_attempts:**

**try:**

**return func(\*args, \*\*kwargs)**

**except Exception as e:**

**attempts += 1**

**print(f"Attempt {attempts}: {e}")**

**raise Exception("Max attempts reached")**

**return inside**

**return decorator**

**@retry(3)**

**def perform\_operation():**

**import random**

**if random.random() < 0.5:**

**raise ValueError("Operation failed")**

**return "Operation successful"**

**result = perform\_operation()**

**From the Below questions (attempt any 8)**

**Programs:**

1. **Problem Statement: Employee Performance Evaluation**

You are given a list of employees, and you need to evaluate their performance based on certain criteria. Each employee is assigned a set of scores for different criteria, and the overall performance score is calculated as the sum of these scores. However, there is a twist: the scores for each criterion are multiplied by a weightage factor.

**Input Specification:**

* employees: A list of dictionaries, where each dictionary represents an employee with the following keys:
* name: A string representing the employee's name (e.g., "John").
* scores: A dictionary where keys are criteria names (e.g., "Quality of Work," "Team Collaboration") and values are the scores for each criterion (e.g., {"Quality of Work": 90, "Team Collaboration": 80}).
* weights: A dictionary where keys are criteria names and values are the weightage factors for each criterion (e.g., {"Quality of Work": 0.6, "Team Collaboration": 0.4}).

Output Specification:

* Return a list of dictionaries, where each dictionary represents an employee with the following keys:
* name: The employee's name.
* performance\_score: The overall performance score after calculating the weighted sum of scores for each criterion.

Your task is to implement a function that takes the input list of employees and calculates their performance scores based on the provided criteria and weights.

1. **Problem Statement: Students Report**

Ms. Margaret is a teacher with a class of N students, and each student is marked for M subjects. She needs to determine which subject to ignore for each student in order to calculate their final grades. Ms. Margaret wants to ignore the subject that has the lowest class average.

**Input Specification:**

* N: An integer representing the number of students.
* M: An integer representing the number of subjects for each student.
* marks: A 2-D integer array of size N x M containing the marks of all students in each subject. Each row corresponds to a student, and each column corresponds to a subject.

**Output Specification:**

Return an integer array of size N containing the total marks for each student after deducting the score for the subject with the lowest class average.

**Here's an example input:**

employees = [

{"name": "John", "scores": {"Quality of Work": 90, "Team Collaboration": 80}, "weights": {"Quality of Work": 0.6, "Team Collaboration": 0.4}},

{"name": "Alice", "scores": {"Quality of Work": 85, "Team Collaboration": 95}, "weights": {"Quality of Work": 0.5, "Team Collaboration": 0.5}},

# Add more employees here...

]

**And the expected output would be a list of dictionaries containing each employee's name and performance score:**

[

{"name": "John", "performance\_score": 86.0},

{"name": "Alice", "performance\_score": 90.0},

# Output for other employees...

]

1. **Problem Statement: Sorting Employee Records**

You are given a list of employee records, where each record is a dictionary with the following keys:

* id: An integer representing the employee's ID.
* name: A string representing the employee's name.
* salary: An integer representing the employee's salary.

**Your task is to implement two functions:**

* sort\_employees\_by\_id(employees): This function takes a list of employee records and sorts them in ascending order of their id values. Return the sorted list of employee records.
* sort\_employees\_by\_salary(employees): This function takes a list of employee records and sorts them in descending order of their salary values. Return the sorted list of employee records.

**Here's an example input:**

employees = [

{"id": 101, "name": "John", "salary": 60000},

{"id": 103, "name": "Alice", "salary": 75000},

{"id": 102, "name": "Bob", "salary": 65000},

# Add more employee records here...

]

**After calling sort\_employees\_by\_id(employees), the list should be sorted by id:**

[

{"id": 101, "name": "John", "salary": 60000},

{"id": 102, "name": "Bob", "salary": 65000},

{"id": 103, "name": "Alice", "salary": 75000},

# Sorted by id in ascending order

]

**After calling sort\_employees\_by\_salary(employees), the list should be sorted by salary in descending order:**

[

{"id": 103, "name": "Alice", "salary": 75000},

{"id": 102, "name": "Bob", "salary": 65000},

{"id": 101, "name": "John", "salary": 60000},

# Sorted by salary in descending order

]

1. **Problem Statement: Find the Kth Number from a Set of Prime Numbers**

Dylan has a collection of prime numbers, and he wants to create a new number by multiplying these prime numbers together. He can use any prime number from his collection an infinite number of times.

Your task is to help Dylan find and return the Kth number that he can create using his set of prime numbers.

**Input Specification:**

- `N`: An integer representing the number of prime numbers in Dylan's collection.

- `prime\_numbers`: An integer array representing the prime numbers that Dylan has in his collection.

- `K`: An integer representing the position of the number Dylan wants to find (Kth number).

**Output Specification:**

Return the Kth number that Dylan can create from his set of prime numbers based on the given input.

For example, if Dylan has 3 prime numbers `[2, 3, 5]` and he wants to find the 4th number he can make, the expected output would be 10 (since 2 \* 5 = 10).

1. **Circular Picnic Game**

As part of a fun school outing, a teacher decided to take her students on a picnic game. She made the children sit in a circle and gave each of them a card with a number, which could be positive or negative. The game involved each child taking turns to speak out their number in clockwise order around the circle until every child had a turn.

As a member of the group, your goal is to win the game by finding the continuous maximum sum of numbers spoken.

**Input Specification:**

- `input1`: The number of children (n).

- `input2`: An array representing the numbers spoken by each child in clockwise order.

**Output Specification:**

Return the continuous maximum sum of numbers spoken during the game.

**Example:**

For instance, if there are 5 children and the array of numbers they speak is [10, -4, 1, 3, 3], the continuous maximum sum is 13, achieved by adding the numbers 10, 1, 3, and 3 together while ignoring the -4.

1. **Maximum Subarray Sum**

You are given an array of integers, and your task is to find the maximum possible sum of a contiguous subarray within the given array.

Write a function `max\_subarray\_sum` to solve this problem. The function should take two arguments:

- `arr`: A list of integers.

- `n`: The number of elements in the array.

The function should return the maximum subarray sum.

**Input:**

- An integer `n` (1 <= n <= 10^6), representing the number of elements in the array.

- A list of integers `arr` (-10^6 <= arr[i] <= 10^6), where 0 <= i < n.

**Output:**

- An integer, the maximum subarray sum.

**Example:**

python

n = 9

arr = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

result = max\_subarray\_sum(arr, n)

print(result) # This should output 6, which is the maximum sum of [4, -1, 2, 1].

**Note:**

In the example provided, the maximum sum is obtained by selecting the subarray [4, -1, 2, 1], which gives a sum of 6.

1. **Longest Common Subsequence**

Write a function `longest\_common\_subsequence` that takes two strings as input and returns the length of their longest common subsequence.

**Input:**

- Two strings `s1` and `s2` (1 <= len(s1), len(s2) <= 1000).

**Output:**

- An integer, the length of the longest common subsequence.

**Example:**

python

s1 = "AGGTAB"

s2 = "GXTXAYB"

result = longest\_common\_subsequence(s1, s2)

print(result) # This should output 4, as the longest common subsequence is "GTAB".

ANS:

def lcs(X, Y, m, n):

if m == 0 or n == 0:

return 0;

elif X[m-1] == Y[n-1]:

return 1 + lcs(X, Y, m-1, n-1);

else:

return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));

# Driver program to test the above function

X = "AGGTAB"

Y = "GXTXAYB"

print ("Length of LCS is ", lcs(X, Y, len(X), len(Y)))

1. **Matrix Spiral Order**

Given an `m x n` matrix, write a function `spiral\_order` to return all elements of the matrix in spiral order.

**Input:**

- A list of lists representing an `m x n` matrix (1 <= m, n <= 100).

- The matrix elements are integers.

**Output:**

- A list of integers, representing the elements of the matrix in spiral order.

**Example:**

python

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

result = spiral\_order(matrix)

print(result) # This should output [1, 2, 3, 6, 9, 8, 7, 4, 5].

ANS:

def spiral\_order(matrix):

if not matrix:

return []

result = []

top, bottom, left, right = 0, len(matrix) - 1, 0, len(matrix[0]) - 1

while top <= bottom and left <= right:

# Traverse from left to right (top boundary)

for j in range(left, right + 1):

result.append(matrix[top][j])

top += 1

# Traverse from top to bottom (right boundary)

for i in range(top, bottom + 1):

result.append(matrix[i][right])

right -= 1

# Check if there are more rows and columns to traverse

if top <= bottom:

# Traverse from right to left (bottom boundary)

for j in range(right, left - 1, -1):

result.append(matrix[bottom][j])

bottom -= 1

if left <= right:

# Traverse from bottom to top (left boundary)

for i in range(bottom, top - 1, -1):

result.append(matrix[i][left])

left += 1

return result

1. **Sudoku Solver**

Write a function `solve\_sudoku` that solves a Sudoku puzzle.

**Input:**

- A 9x9 Sudoku board represented by a list of lists.

- The empty cells are represented by '0', and filled cells have digits from '1' to '9'.

**Output:**

- A solved Sudoku board.

**Example:**

python

sudoku\_board = [

[5, 3, 0, 0, 7, 0, 0, 0, 0],

[6, 0, 0, 1, 9, 5, 0, 0, 0],

[0, 9, 8, 0, 0, 0, 0, 6, 0],

[8, 0, 0, 0, 6, 0, 0, 0, 3],

[4, 0, 0, 8, 0, 3, 0, 0, 1],

[7, 0, 0, 0, 2, 0, 0, 0, 6],

[0, 6, 0, 0, 0, 0, 2, 8, 0],

[0, 0, 0, 4, 1, 9, 0, 0, 5],

[0, 0, 0, 0, 8, 0, 0, 7, 9]

]

solve\_sudoku(sudoku\_board)

# The modified sudoku\_board should be the solved Sudoku puzzle.

ANS:

def solve\_sudoku(board):

def is\_valid(num, row, col):

# Check if 'num' is valid in the current cell

for i in range(9):

if board[i][col] == num or board[row][i] == num:

return False

# Check if 'num' is valid in the 3x3 subgrid

start\_row, start\_col = 3 \* (row // 3), 3 \* (col // 3)

for i in range(start\_row, start\_row + 3):

for j in range(start\_col, start\_col + 3):

if board[i][j] == num:

return False

return True

def backtrack():

for row in range(9):

for col in range(9):

if board[row][col] == '0':

for num in map(str, range(1, 10)):

if is\_valid(num, row, col):

board[row][col] = num

if backtrack():

return True

board[row][col] = '0' # If not a valid solution, backtrack

return False # If no valid number can be placed, backtrack

return True # All cells filled successfully

backtrack()

return board

1. Word Ladder

Given two words, `beginWord` and `endWord`, and a word list, write a function `find\_ladders` to find all the shortest transformation sequences from `beginWord` to `endWord`, such that:

1. Only one letter can be changed at a time.

2. Each transformed word must exist in the word list.

**Input:**

- `beginWord`: A string (1 <= len(beginWord) <= 10).

- `endWord`: A string (1 <= len(endWord) <= 10).

- `wordList`: A list of strings, representing the word list.

**Output:**

- A list of lists, each representing a transformation sequence.

**Example:**

python

beginWord = "hit"

endWord = "cog"

wordList = ["hot", "dot", "dog", "lot", "log", "cog"]

result = find\_ladders(beginWord, endWord, wordList)

# The result should contain all shortest transformation sequences from "hit" to "cog".