Python

# Part 1:

**Exercise**

1. **Conditional Statements and Boolean Logic**

* Create a program that determines whether a given year is a leap year.
  + (A leap year is divisible by 4, but not by 100 unless also divisible by 400).

1. **Loops and Loop Controls**

* Write a Python program that prints numbers from 1 to 50 but:
  + Replaces multiples of 3 with "Fizz".
  + Replaces multiples of 5 with "Buzz".
  + Replaces multiples of both 3 and 5 with "FizzBuzz".

1. **List Manipulation Challenge:**

# Combines lists, loops, and string manipulation

def process\_names(name\_list):

"""

Take a list of names, capitalize first letter,

sort them and return as comma-separated string

"""

1. **Dictionary challenge**:

There are 3 inputs, but the problem statement is same (grades should be between 0&100, for anything else “warning” should be given):

Grades1 = { "Eve": "Not graded", "Frank": 88, "Grace": -5, "Henry": 105, "Ivy": 91 }

Grades2 = { "Eve": 95, "Frank": 88, "Grace": 5, "Henry": 105, "Ivy": 91 }

After taking the Grades, I want the final response to be

**Example output response:**

Warning: Invalid grade format for Eve: Not graded Warning: Invalid grade range for Grace: -5 Warning: Invalid grade range for Henry: 105

For the valid ones, I want the below response

{‘errors’:2, 'average': 89.5, 'highest': 91, 'lowest': 88, 'count': 2}

1. **Error-Handled Calculator:**

# Combines operators, functions, and error handling

def safe\_calculator(operation, num1, num2):

"""

Perform operation with comprehensive error handling

"""

1. **Set Operations Manager:**

# Combines sets, functions, and logical operators

def analyze\_sets(set1, set2):

"""

Return union, intersection, difference

Handle edge cases

"""

1. **Dictionaries and String Manipulation**  
   Create a dictionary that stores the names of 5 countries as keys and their capitals as values. Write a program that takes a country name as input and prints its capital. Use string methods to ensure the input is case-insensitive.
2. **Lambda Functions**

Implement a lambda function that:

Takes two numbers as input.

Returns their product if both are even, otherwise their sum.

1. **Word Frequency Counter**
   1. **Tasks Evaluated**: String manipulation, dictionaries, loops, functions.

**Description**:

* 1. Write a program that takes a paragraph of text as input and counts the frequency of each word.
  2. Use a dictionary to store the word counts.
  3. Print the top 5 most frequent words along with their counts.
  4. Use string methods to clean the input (e.g., remove punctuation and convert to lowercase).

1. **Basic Banking System**

💡 *A simple banking system where users can check balance, deposit, and withdraw money.*

* Create a dictionary where keys are user names and values are their account balances.
* Provide a menu for:
  + Checking balance
  + Depositing money
  + Withdrawing money (ensure enough funds are available)
* Ensure the program doesn't crash if the user enters invalid input (e.g., letters instead of numbers).

1. **Exercise: Local vs. Global Variables**

**Problem Statement**

Write a Python program that demonstrates the difference between local and global variables. The program should:

1. Define a global variable counter and initialize it to 0.
2. Create a function increment\_counter that:
   * Defines a local variable counter and increments it by 1.
   * Prints the value of the local counter.
3. Call the increment\_counter function multiple times and observe the output.
4. Modify the function to increment the global counter instead of the local one and observe the changes.

# Part 2

**1. Sum of Even Numbers (One-Liner)**

**Question:**  
Write a Python program that takes a list of integers and returns the sum of all even numbers using a one-liner for loop.

**Input:**

numbers = [3, 8, 12, 5, 9, 14, 7]

**Expected Output:**

34 # (8 + 12 + 14)

**2. Reverse a String (Without Slicing or Built-in Functions)**

**Question:**  
Write a Python function that reverses a string using only a for loop (without [::-1] or reversed()).

**Input:**

text = "Python"

**Expected Output:**

"nohtyP"

**3. Extract Words with More than 5 Letters (One-Liner)**

**Question:**  
Given a sentence, use a **list comprehension** to extract all words that are **longer than 5 letters**.

**Input:**

sentence = "Python is a powerful and versatile programming language"

**Expected Output:**

['Python', 'powerful', 'versatile', 'programming', 'language']

**4. Find the First Non-Repeating Character**

**Question:**  
Given a string, find the first character that **does not repeat** in the string using a for loop.

**Input:**

text = "swiss"

**Expected Output:**

"w"

**5. Count Word Frequency in a Sentence**

**Question:**  
Write a program that **counts the occurrences** of each word in a given sentence using **dictionary comprehension** and a for loop.

**Input:**

sentence = "python is fun and python is powerful"

**Expected Output:**

python

{'python': 2, 'is': 2, 'fun': 1, 'and': 1, 'powerful': 1}

**6. Generate a Fibonacci Sequence (While Loop)**

👉 **Question:**  
Write a function to generate the **first n Fibonacci numbers** using a while loop.

**Input:**

n = 7

**Expected Output:**

[0, 1, 1, 2, 3, 5, 8]

**7. Find Prime Numbers Using Nested Loops**

👉 **Question:**  
Write a **nested loop** that generates all prime numbers from **2 to 50**.

**Input:**

n = 20

**Expected Output:**

[2, 3, 5, 7, 11, 13, 17, 19]

**8. Print a Pattern Using Nested Loops**

👉 **Question:**  
Write a Python program to print the following pattern using **nested loops**:

markdown

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

**Input:**

rows = 5

**Expected Output:**

markdown

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

**9. Matrix Multiplication (Using Nested Loops)**

👉 **Question:**  
Write a function to perform **matrix multiplication** for two N x N matrices using **nested loops**.

**Input:**

A = [[1, 2],

[3, 4]]

B = [[5, 6],

[7, 8]]

**Expected Output:**

[[19, 22],

[43, 50]]

**10. Flatten a Nested List (Using Nested Loops in One Line)**

👉 **Question:**  
Given a **nested list** of integers, flatten it into a **single list** using a **nested for loop in one line**.

**Input:**

nested\_list = [[1, 2, 3], [4, 5], [6, 7, 8]]

**Expected Output:**

[1, 2, 3, 4, 5, 6, 7, 8]

# Part 3

**Dictionary Exercises**

**1. Access Dictionary Values Using Keys**

**Question:**  
Given a dictionary of student grades, return the grade for a specific student.

**Input:**

grades = {"Alice": 85, "Bob": 92, "Charlie": 78}

student = "Bob"

**Expected Output:**

92

**2. Update or Add a Key in a Dictionary**

**Question:**  
Modify a dictionary to update the grade of an existing student or add a new student.

**Input:**

grades = {"Alice": 85, "Bob": 92}

new\_entry = ("Charlie", 88) # (key, value)

**Expected Output:**

{"Alice": 85, "Bob": 92, "Charlie": 88}

**3. Remove a Key from a Dictionary Safely**

**Question:**  
Delete a given key from the dictionary safely without causing an error.

**Input:**

grades = {"Alice": 85, "Bob": 92, "Charlie": 78}

remove\_key = "Charlie"

**Expected Output:**

{"Alice": 85, "Bob": 92}

**4. Merge Two Dictionaries**

**Question:**  
Combine two dictionaries, ensuring that the second dictionary overwrites duplicate keys.

**Input:**

dict1 = {"Alice": 85, "Bob": 92}

dict2 = {"Charlie": 88, "Bob": 95}

**Expected Output:**

{"Alice": 85, "Bob": 95, "Charlie": 88}

**5. Count Occurrences of Characters in a String (Using Dictionary & Loop)**

**Question:**  
Write a function that counts how many times each character appears in a given string.

**Input:**

text = "banana"

**Expected Output:**

{'b': 1, 'a': 3, 'n': 2}

**6. Filter Dictionary Based on Values (Using Loop)**

**Question:**  
Filter out students who scored below 80 from a dictionary.

**Input:**

grades = {"Alice": 85, "Bob": 92, "Charlie": 78, "David": 65}

**Expected Output:**

{"Alice": 85, "Bob": 92}

**7. Invert a Dictionary (Keys Become Values & Vice Versa, Using Loop)**

**Question:**  
Swap keys and values in a dictionary.

**Input:**

grades = {"Alice": 85, "Bob": 92, "Charlie": 78}

**Expected Output:**

{85: "Alice", 92: "Bob", 78: "Charlie"}

# Part 4

**Loop Exercises**

**1. Spiral Matrix Traversal**

**Question:**  
Given a **N x N matrix**, print all its elements in a **spiral order** (clockwise).

**Input:**

matrix = [

[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12],

[13, 14, 15, 16]

]

**Expected Output:**

[1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10]

**2. Generate Pascal’s Triangle (N Rows)**

**Question:**  
Generate Pascal’s triangle up to **N rows**, where each number is the sum of the two numbers directly above it.

**Input:**

n = 5

**Expected Output:**

[

[1],

[1, 1],

[1, 2, 1],

[1, 3, 3, 1],

[1, 4, 6, 4, 1]

]

**3. Find Subarrays That Sum to Zero**

**Question:**  
Given a list of integers, find all **subarrays** (continuous segments) that sum to **zero**.

**Input:**

nums = [3, 4, -7, 1, 2, -1, -3, 6]

**Expected Output:**

[

[3, 4, -7],

[4, -7, 1, 2],

[-1, -3, 6]

]

# Part 5

**Dictionary Exercises**

**1. Group Anagrams**

**Question:**  
Given a list of words, **group anagrams together** using a dictionary.

**Input:**

words = ["listen", "silent", "enlist", "rat", "tar", "art", "evil", "vile", "live"]

**Expected Output:**

{

"eilnst": ["listen", "silent", "enlist"],

"art": ["rat", "tar", "art"],

"eilv": ["evil", "vile", "live"]

}

**2. Find Most Frequent Element in a List**

**Question:**  
Given a list, find the **most frequently occurring element** using a dictionary.

**Input:**

nums = [1, 3, 3, 4, 2, 1, 3, 2, 4, 2, 2]

**Expected Output:**

2

**3. Invert a Nested Dictionary**

**Question:**  
Invert a **nested dictionary**, swapping keys and values at all levels.

**Input:**

data = {

"a": {"x": 1, "y": 2},

"b": {"x": 3, "z": 4}

}

**Expected Output:**

{

"x": {"a": 1, "b": 3},

"y": {"a": 2},

"z": {"b": 4}

}