Rajalakshmi Engineering College

Name: Mythrayee V
Email: 241501125@rajalakshmi.edu.in
Roll no: 241501125
Phone: 9042130487
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 5_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Professor Adams needs to analyze student participation in three recent academic workshops. She has three sets of student IDs: the first set contains students who registered for the workshops, the second set contains students who actually attended, and the third set contains students who dropped out.

Professor Adams needs to determine which students who registered also attended, and then identify which of these students did not drop out.

Help Professor Adams identify the students who registered, attended, and did not drop out of the workshops.

Input Format

The first line of input consists of integers, representing the student IDs who registered for the workshops.

The second line consists of integers, representing the student IDs who attended the workshops.

The third line consists of integers, representing the student IDs who dropped out of the workshops.

Output Format

The first line of output displays the intersection of the first two sets, which shows the IDs of students who registered and attended.

The second line displays the result after removing student IDs that are in the third set (dropped out), showing the IDs of students who both attended and did not drop out.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 2 3
2 3 4
3 4 5
Output: {2, 3}
{2}

Answer

```
# You are using Python
reg = set(map(int,input().split()))
att = set(map(int,input().split()))
dro = set(map(int,input().split()))

ra = reg & att
fp = ra - dro
print(ra)
print(fp)
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

James is managing a list of inventory items in a warehouse. Each item is recorded as a tuple, where the first element is the item ID and the second element is a list of quantities available for that item. James needs to filter out all quantities that are above a certain threshold to find items that have a stock level above this limit.

Help James by writing a program to process these tuples, filter the quantities from all the available items, and display the results.

Note:

Use the filter() function to filter out the quantities greater than the specified threshold for each item's stock list.

Input Format

The first line of input consists of an integer N, representing the number of tuples.

The next N lines each contain a tuple in the format (ID, [quantity1, quantity2, ...]), where ID is an integer and the list contains integers.

The final line consists of an integer threshold, representing the quantity threshold.

Output Format

The output should be a single line displaying the filtered quantities, space-separated. Each quantity is strictly greater than the given threshold.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
(1, [1, 2])
(2, [3, 4])
2
Output: 3 4

Answer

```
# You are using Python
N = int(input())

items = []
for _ in range(N):
  item = eval(input())
  items.append(item)

threshold = int(input())

filtered_quantities = []

for item in items:
  id, quantities = item

  filtered = list(filter(lambda x: x > threshold, quantities))
  filtered_quantities.extend(filtered)

print(*filtered_quantities)
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Gowshik is working on a task that involves taking two lists of integers as input, finding the element-wise sum of the corresponding elements, and then creating a tuple containing the sum values.

Write a program to help Gowshik with this task.

Example:

Given list:

[1, 2, 3, 4]

[3, 5, 2, 1]

An element-wise sum of the said tuples: (4, 7, 5, 5)

Input Format

The first line of input consists of a single integer n, representing the length of the input lists.

The second line of input consists of n integers separated by commas, representing the elements of the first list.

The third line of input consists of n integers separated by commas, representing the elements of the second list.

Output Format

The output is a single line containing a tuple of integers separated by commas, representing the element-wise sum of the corresponding elements from the two input lists.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
1, 2, 3, 4
3, 5, 2, 1
Output: (4, 7, 5, 5)

Answer

```
# You are using Python
n = int(input())
l1 = list(map(int,input().split(',')))
l2 = list(map(int,input().split(',')))
result = tuple(a+b for a,b in zip(l1, l2))
print(result)
```

Status : Correct *Marks : 10/10*

4. Problem Statement

Liam is analyzing a list of product IDs from a recent sales report. He needs

to determine how frequently each product ID appears and calculate the following metrics:

Frequency of each product ID: A dictionary where the key is the product ID and the value is the number of times it appears.Total number of unique product IDs.Average frequency of product IDs: The average count of all product IDs.

Write a program to read the product IDs, compute these metrics, and output the results.

Example

Input:

6 //number of product ID

101

102

101

103

101

102 //product IDs

Output:

{101: 3, 102: 2, 103: 1}

Total Unique IDs: 3

Average Frequency: 2.00

Explanation:

Input 6 indicates that you will enter 6 product IDs.

A dictionary is created to track the frequency of each product ID.

Input 101: Added with a frequency of 1.

Input 102: Added with a frequency of 1.

Input 101: Frequency of 101 increased to 2.

Input 103: Added with a frequency of 1.

Input 101: Frequency of 101 increased to 3.

Input 102: Frequency of 102 increased to 2.

The dictionary now contains 3 unique IDs: 101, 102, and 103.

Total Unique is 3.

The average frequency is 2.00.

Input Format

The first line of input consists of an integer n, representing the number of product IDs.

The next n lines each contain a single integer, each representing a product ID.

Output Format

The first line of output displays the frequency dictionary, which maps each product ID to its count.

The second line displays the total number of unique product IDs, preceded by "Total Unique IDs: ".

The third line displays the average frequency of the product IDs. This is calculated by dividing the total number of occurrences of all product IDs by the total number of unique product IDs, rounded to two decimal places. It is preceded by "Average Frequency: ".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
101
102
101
103

101
102
Output: {101: 3, 102: 2, 103: 1}
Total Unique IDs: 3
Average Frequency: 2.00

Answer

```
# You are using Python
n = int(input())

frequency = {}
for _ in range(n):
    product_id = int(input())
    if product_id in frequency:
        frequency[product_id] += 1
    else:
        frequency[product_id] = 1


total_unique = len(frequency)
avg_frequency = sum(frequency.values()) / total_unique
print(frequency)
print(f"Total Unique IDs: {total_unique}")
print(f"Average Frequency: {avg_frequency:.2f}")
```

Status : Correct ***Marks : 10/10***

5. Problem Statement

Ella is analyzing the sales data for a new online shopping platform. She has a record of customer transactions where each customer's data includes their ID and a list of amounts spent on different items. Ella needs to determine the total amount spent by each customer and identify the highest single expenditure for each customer.

Your task is to write a program that computes these details and displays them in a dictionary.

Input Format

The first line of input consists of an integer n, representing the number of customers.

Each of the next n lines contains a numerical customer ID followed by integers representing the amounts spent on different items.

Output Format

The output displays a dictionary where the keys are customer IDs and the values are lists containing two integers: the total expenditure and the maximum single expenditure.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
101 100 150 200
102 50 75 100
Output: {101: [450, 200], 102: [225, 100]}

Answer

```
# You are using Python
n = int(input())
sd ={}
for _ in range(n):
    data = list(map(int,input().split()))
    cid = data[0]
    amt = data[1:]
    ts = sum(amt)
    ms = max(amt)
    sd[cid] = [ts, ms]
print(sd)
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Mythrayee V
Email: 241501125@rajalakshmi.edu.in
Roll no: 241501125
Phone: 9042130487
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 5_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 39

Section 1 : Coding

1. Problem Statement

Riley is analyzing DNA sequences and needs to determine which bases match at the same positions in two given DNA sequences. Each DNA sequence is represented as a tuple of integers, where each integer corresponds to a DNA base.

Your task is to write a program that compares these two sequences and identifies the bases that match at the same positions and print it.

Input Format

The first line of input consists of an integer n, representing the size of the first tuple.

The second line contains n space-separated integers, representing the elements of the first DNA sequence tuple.

The third line of input consists of an integer m, representing the size of the second tuple.

The fourth line contains m space-separated integers, representing the elements of the second DNA sequence tuple.

Output Format

The output is a space-separated integer of the matching bases at the same positions in both sequences.

Refer to the sample output for format specifications.

Sample Test Case

Input: 4
5 1 8 4
4
4 1 8 2

Output: 1 8

Answer

```
# You are using Python
# Read the size and elements of the first DNA sequence
n = int(input())
seq1 = tuple(map(int, input().split()))

# Read the size and elements of the second DNA sequence
m = int(input())
seq2 = tuple(map(int, input().split()))

# Find matching bases at the same positions
min_len = min(n, m)
matches = []

for i in range(min_len):
    if seq1[i] == seq2[i]:
        matches.append(str(seq1[i]))

# Print the result as space-separated values
```

```
print(' '.join(matches))
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Alex is working with grayscale pixel intensities from an old photo that has been scanned in a single row. To detect edges in the image, Alex needs to calculate the differences between each pair of consecutive pixel intensities.

Your task is to write a program that performs this calculation and returns the result as a tuple of differences.

Input Format

The first line of input contains an integer n, representing the number of pixel intensities.

The second line contains n space-separated integers representing the pixel intensities.

Output Format

The output displays a tuple containing the absolute differences between consecutive pixel intensities.

Refer to the sample output for format specifications.

Sample Test Case

Input: 5
200 100 20 80 10
Output: (100, 80, 60, 70)

Answer

```
# You are using Python
# Read the number of pixel intensities
n = int(input())
```

```
# Read the pixel intensities as a list of integers
pixels = list(map(int, input().split()))

# Calculate the absolute differences between consecutive pixels
differences = tuple(abs(pixels[i+1] - pixels[i]) for i in range(n - 1))

# Print the result as a tuple
print(differences)
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Riya owns a store and keeps track of item prices from two different suppliers using two separate dictionaries. He wants to compare these prices to identify any differences. Your task is to write a program that calculates the absolute difference in prices for items that are present in both dictionaries. For items that are unique to one dictionary (i.e., not present in the other), include them in the output dictionary with their original prices.

Help Riya to implement the above task using a dictionary.

Input Format

The first line of input consists of an integer n1, representing the number of items in the first dictionary.

The next n1 lines contain two integers

1. The first line contains the item (key), and
2. The second line contains the price (value).

The following line consists of an integer n2, representing the number of items in the second dictionary

The next n2 lines contain two integers

1. The first line contains the item (key), and
2. The second line contains the price (value).

Output Format

The output should display a dictionary that includes:

1. For items common to both dictionaries, the absolute difference between their prices.
2. For items that are unique to one dictionary, the original price from that dictionary.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
4
4
1
8
7

Output: {4: 4, 8: 7}

Answer

```
# Read number of items in the first dictionary
n1 = int(input())
dict1 = {}
order = []  # To preserve key order
for _ in range(n1):
    key = int(input())
    value = int(input())
    dict1[key] = value
    if key not in order:
        order.append(key)

# Read number of items in the second dictionary
n2 = int(input())
dict2 = {}
for _ in range(n2):
    key = int(input())
    value = int(input())
    dict2[key] = value
```

```
    if key not in dict1 and key not in order:
        order.append(key)

# Create result dictionary with ordered keys
result = {}
for key in order:
    if key in dict1 and key in dict2:
        result[key] = abs(dict1[key] - dict2[key])
    elif key in dict1:
        result[key] = dict1[key]
    else:
        result[key] = dict2[key]

# Print the result dictionary
print(result)
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Alex is tasked with managing the membership lists of several exclusive clubs. Each club has its own list of members, and Alex needs to determine the unique members who are part of exactly one club when considering all clubs together.

Your goal is to help Alex by writing a program that calculates the symmetric difference of membership lists from multiple clubs and then finds the total number of unique members.

Input Format

The first line of input consists of an integer k, representing the number of clubs.

The next k lines each contain a space-separated list of integers, where each integer represents a member's ID.

Output Format

The first line of output displays the symmetric difference of the membership lists as a set.

The second line displays the sum of the elements in this symmetric difference.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
1 2 3
2 3 4
5 6 7
Output: {1, 4, 5, 6, 7}
23

Answer

```
k = int(input())
sets = []

for _ in range(k):
    members = set(map(int, input().split()))
    sets.append(members)

# Calculate symmetric difference of all sets
result = sets[0]
for s in sets[1:]:
    result = result.symmetric_difference(s)

# Print the symmetric difference set
print(result)
# Print the sum of elements in the symmetric difference set
print(sum(result))
```

Status : Partially correct ***Marks : 9/10***

Rajalakshmi Engineering College

Name: Mythrayee V
Email: 241501125@rajalakshmi.edu.in
Roll no: 241501125
Phone: 9042130487
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 5_MCQ

Attempt : 1
Total Mark : 20
Marks Obtained : 14

Section 1 : MCQ

1. What is the output of the following?

```
set1 = {10, 20, 30, 40, 50}
set2 = {60, 70, 10, 30, 40, 80, 20, 50}
print(set1.issubset(set2))
print(set2.issuperset(set1))
```

Answer

TrueTrue

Status : Correct ***Marks : 1/1***

2. Fill in the code in order to get the following output.

Output:

Tuple: (1, 3, 4)

Max value: 4

t=(1,)

_____

print("Tuple:" ,t)
print("Max value:",_____)

Answer

1) t1=t+(3,4)2) max(t)

Status : Wrong ***Marks : 0/1***

3. Predict the output of the following Python program

init_tuple_a = 1, 2, 8
init_tuple_b = (1, 2, 7)
set1=set(init_tuple_b)
set2=set(init_tuple_a)
print (set1 | set2)
print (init_tuple_a | init_tuple_b)

Answer

{1, 2, 7, 8}TypeError: unsupported operand type

Status : Correct ***Marks : 1/1***

4. What will be the output?

a={'B':5,'A':9,'C':7}
print(sorted(a))

Answer

['A', 'B', 'C'].

Status : Correct ***Marks : 1/1***

5. What will be the output for the following code?

```
a=(1,2,3)
b=('A','B','C')
c=zip(a,b)

print(c)
print(tuple(c))
```

Answer

((1, 'A'), (2, 'B'), (3, 'C'))

Status : Correct ***Marks : 1/1***

6. Which of the statements about dictionary values is false?

Answer

Values of a dictionary must be unique

Status : Correct ***Marks : 1/1***

7. What is the output of the following code?

```
a={"a":1,"b":2,"c":3}
b=dict(zip(a.values(),a.keys()))
print(b)
```

Answer

{1: ‘a’, 2: ‘b’, 3: ‘c’}

Status : Correct ***Marks : 1/1***

8. Which of the following is a Python tuple?

Answer

[1, 2, 3]

Status : Wrong ***Marks : 0/1***

9. What is the output of the following code?

```
a=(1,2,(4,5))
b=(1,2,(3,4))
print(a<b)
```

Answer

Error, < operator is not valid for tuples

Status : Wrong ***Marks : 0/1***

10. What will be the output of the following program?

```
set1 = {1, 2, 3}
set2 = set1.copy()
set2.add(4)
print(set1)
```

Answer

{1,2,3,4}

Status : Wrong ***Marks : 0/1***

11. Which of the following isn't true about dictionary keys?

Answer

Keys must be integers

Status : Correct ***Marks : 1/1***

12. What will be the output for the following code?

```
t1 = (1, 2, 4, 3)
t2 = (1, 2, 3, 4)
print(t1 < t2)
```

Answer

False

Status : Correct ***Marks : 1/1***

13. What is the output of the below Python code?

```
list1 = [1, 2, 3]
list2 = [5, 6, 7]
list3 = [10, 11, 12]
set1 = set(list2)
set2 = set(list1)
set1.update(set2)
set1.update(list3)
print(set1)
```

Answer

{1, 2, 3, 5, 6, 7, 10, 11, 12}

Status : Correct ***Marks : 1/1***

14. If 'a' is a dictionary with some key-value pairs, what does a.popitem() do?

Answer

Removes an arbitrary element

Status : Correct ***Marks : 1/1***

15. What is the output of the following code?

```
a={1:"A",2:"B",3:"C"}
b=a.copy()
b[2]="D"
print(a)
```

Answer

{1: ‘A’, 2: ‘D’, 3: ‘C’}

Status : Wrong ***Marks : 0/1***

16. Suppose t = (1, 2, 4, 3), which of the following is incorrect?

Answer

t[3] = 45

Status : Correct *Marks : 1/1*

17. Set s1 = {1, 2, 4, 3} and s2 = {1, 5, 4, 6}, find s1 & s2, s1 - s2, s1 | s2 and s1 ^ s2.

Answer

s1&s2 = {1, 4}s1-s2 = {2, 3}s1^s2 = {2, 3, 5, 6}s1|s2 = {1, 2, 3, 4, 5, 6}

Status : Correct *Marks : 1/1*

18. What is the result of print(type({}) is set)?

Answer

False

Status : Correct *Marks : 1/1*

19. What will be the output of the following code?

```
a=(1,2,3,4)
print(sum(a,3))
```

Answer

13

Status : Correct *Marks : 1/1*

20. Which of the following statements is used to create an empty tuple?

Answer

[ ]

Status : Wrong

Marks : 0/1

Rajalakshmi Engineering College

Name: Mythrayee V
Email: 241501125@rajalakshmi.edu.in
Roll no: 241501125
Phone: 9042130487
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 5_PAH

Attempt : 1
Total Mark : 60
Marks Obtained : 60

Section 1 : Coding

1. Problem Statement

Maya wants to create a dictionary that maps each integer from 1 to a given number n to its square. She will use this dictionary to quickly reference the square of any number up to n.

Help Maya generate this dictionary based on the input she provides.

Input Format

The input consists of an integer n, representing the highest number for which Maya wants to calculate the square.

Output Format

The output displays the generated dictionary where each key is an integer from 1 to n, and the corresponding value is its square.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
Output: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

Answer

```
# You are using Python
n = int(input())
sd = {i: i**2 for i in range(1,n+1)}
print(sd)
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Rishi is working on a program to manipulate a set of integers. The program should allow users to perform the following operations:

Find the maximum value in the set.Find the minimum value in the set.Remove a specific number from the set.

The program should handle these operations based on user input. If the user inputs an invalid operation choice, the program should indicate that the choice is invalid.

Input Format

The first line contains space-separated integers that will form the initial set. Each integer x is separated by a space.

The second line contains an integer ch, representing the user's choice:

- 1 to find the maximum value
- 2 to find the minimum value
- 3 to remove a specific number from the set

If ch is 3, the third line contains an integer n1, which is the number to be

removed from the set.

Output Format

The first line of output prints the original set in descending order.

For choice 1: Print the maximum value from the set.

For choice 2: Print the minimum value from the set.

For choice 3: Print the set after removing the specified number, in descending order.

For invalid choices: Print "Invalid choice".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 2 3 4 5
1
Output: {5, 4, 3, 2, 1}
5

Answer

```
# You are using Python
# Read the initial set of integers
input_set = set(map(int, input().split()))

# Read the user's choice
ch = int(input())

# Print the original set in descending order
print("{" + ", ".join(map(str, sorted(input_set, reverse=True))) + "}")

# Perform the operation based on the user's choice
if ch == 1:
    print(max(input_set))
elif ch == 2:
    print(min(input_set))
elif ch == 3:
```

```
    # Read the number to be removed
    n1 = int(input())
    input_set.discard(n1)  # discard avoids error if n1 is not in the set
    print("{" + ", ".join(map(str, sorted(input_set, reverse=True))) + "}")
else:
    print("Invalid choice")
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Mia is organizing a list of integers into a series of pairs for his new project. She wants to create pairs of consecutive integers from the list. The last integer should be paired with None to complete the series. The pairing happens as follows: ((Element 1, Element 2), (Element 2, Element 3)..... (Element n, None)).

Your task is to help Henry by writing a Python program that reads a list of integers, forms these pairs, and displays the result in tuple format.

Input Format

The first line of input consists of an integer n, representing the number of elements in the tuple.

The second line of input contains n space-separated integers, representing the elements of the tuple.

Output Format

The output displays a tuple containing pairs of consecutive integers from the input. The last integer in the tuple is paired with 'None'.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
5 10 15

Output: ((5, 10), (10, 15), (15, None))

Answer

```
# You are using Python
# Read the number of elements
n = int(input())

# Read the elements and convert them into a list of integers
elements = list(map(int, input().split()))

# Create the list of pairs
result = []
for i in range(n):
    if i < n - 1:
        result.append((elements[i], elements[i + 1]))
    else:
        result.append((elements[i], None))

# Convert list of pairs to tuple and print
print(tuple(result))
```

Status : Correct *Marks : 10/10*

4. Problem Statement

Tom wants to create a dictionary that lists the first n prime numbers, where each key represents the position of the prime number, and the value is the prime number itself.

Help Tom generate this dictionary based on the input she provides.

Input Format

The input consists of an integer n, representing the number of prime numbers Tom wants to generate.

Output Format

The output displays the generated dictionary where each key is an integer from 1 to n, and the corresponding value is the prime number.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4

Output: {1: 2, 2: 3, 3: 5, 4: 7}

Answer

```
# You are using Python
# Function to check if a number is prime
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

# Read input
n = int(input())

# Generate the first n prime numbers
prime_dict = {}
num = 2  # Start checking from 2
count = 1

while count <= n:
    if is_prime(num):
        prime_dict[count] = num
        count += 1
    num += 1

# Print the dictionary
print(prime_dict)
```

Status : Correct ***Marks : 10/10***

5. Problem Statement

Sophia is organizing a list of event IDs representing consecutive days of an

event. She needs to group these IDs into consecutive sequences. For example, if the IDs 3, 4, and 5 appear consecutively, they should be grouped.

Write a program that helps Sophia by reading the total number of event IDs and the IDs themselves, then display each group of consecutive IDs in tuple format.

Input Format

The first line of input consists of an integer n, representing the number of event IDs.

The next n lines contain integers representing the event IDs, where each integer corresponds to an event ID.

Output Format

The output should display each group of consecutive event IDs in a tuple format. Each group should be printed on a new line, and single event IDs should be displayed as a single-element tuple.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1
2
3
Output: (1, 2, 3)

Answer

```
# Read the number of event IDs
n = int(input())

# Read each event ID
ids = [int(input()) for _ in range(n)]

# Sort the IDs to group consecutive numbers
ids.sort()
```

```
# Group consecutive sequences
result = []
current_group = [ids[0]]

for i in range(1, n):
    if ids[i] == ids[i - 1] + 1:
        current_group.append(ids[i])
    else:
        result.append(current_group)
        current_group = [ids[i]]
result.append(current_group)  # Add the last group

# Print each group
for group in result:
    if len(group) == 1:
        print(f"({group[0]})")
    else:
        print(tuple(group))
```

Status : Correct ***Marks : 10/10***

6. Problem Statement

Jordan is creating a program to process a list of integers. The program should take a list of integers as input, remove any duplicate integers while preserving their original order, concatenate the remaining unique integers into a single string, and then print the result.

Help Jordan in implementing the same.

Input Format

The input consists of space-separated integers representing the elements of the set.

Output Format

The output prints a single integer formed by concatenating the unique integers from the input in the order they appeared.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 11 11 33 50

Output: 113350

Answer

```
# You are using Python
# Read the input list of integers
input_list = list(map(int, input().split()))

# Remove duplicates while preserving order
unique_list = []
seen = set()
for num in input_list:
    if num not in seen:
        unique_list.append(num)
        seen.add(num)

# Concatenate the unique integers into a single string
result = ''.join(str(num) for num in unique_list)

# Print the result
print(result)
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Mythrayee V
Email: 241501125@rajalakshmi.edu.in
Roll no: 241501125
Phone: 9042130487
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Write a program that calculates the average of a list of integers. The program prompts the user to enter the length of the list (n) and each element of the list. It performs error handling to ensure that the length of the list is a non-negative integer and that each input element is a numeric value.

Input Format

The first line of the input is an integer n, representing the length of the list as a positive integer.

The second line of the input consists of an element of the list as an integer, separated by a new line.

Output Format

If the length of the list is not a positive integer or zero, the output displays "Error: The length of the list must be a non-negative integer."

If a non-numeric value is entered for the length of the list, the output displays "Error: You must enter a numeric value."

If a non-numeric value is entered for a list element, the output displays "Error: You must enter a numeric value."

If the inputs are valid, the program calculates and prints the average of the provided list of integers with two decimal places: "The average is: [average]".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: -2
1
2
Output: Error: The length of the list must be a non-negative integer.

Answer

```
# You are using Python
try:
    # Prompt for the length of the list
    n_str = input()

    # Error handling for non-numeric or empty input for n
    if not n_str.strip():
        print("Error: You must enter a numeric value.")
        exit()

    n = int(n_str)

except ValueError:
    # Catch ValueError if the input for n cannot be converted to an integer
    print("Error: You must enter a numeric value.")
```

```
    exit()

# Error handling for non-positive list length
if n <= 0:
    print("Error: The length of the list must be a non-negative integer.")
    exit()

numbers = []
# Loop to get each element of the list
for _ in range(n):
    try:
        element_str = input()

        # Error handling for non-numeric or empty input for list elements
        if not element_str.strip():
            print("Error: You must enter a numeric value.")
            exit()

        element = int(element_str)
        numbers.append(element)
    except ValueError:
        # Catch ValueError if a list element cannot be converted to an integer
        print("Error: You must enter a numeric value.")
        exit()

# Calculate and print the average if the list is not empty
if numbers:
    average = sum(numbers) / n
    print(f"The average is: {average:.2f}")
```

Status : Correct **Marks : 10/10**

2. Problem Statement

In a voting system, a person must be at least 18 years old to be eligible to vote. If a user enters an age below 18, the system should raise a user-defined exception indicating that they are not eligible to vote.

Input Format

The input contains a positive integer representing age.

Output Format

If the age is less than 18, the output displays "Not eligible to vote".

Otherwise, the output displays "Eligible to vote".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 18
Output: Eligible to vote

Answer

```
# You are using Python
class NotEligibleToVoteError(Exception):
    """Custom exception raised when a person is not eligible to vote."""
    pass

try:
    age = int(input())

    if age < 18:
        raise NotEligibleToVoteError
    else:
        print("Eligible to vote")

except NotEligibleToVoteError:
    print("Not eligible to vote")
except ValueError:
    # This block handles cases where the input is not an integer.
    # While the problem constraints state input will be a positive integer,
    # including this for robustness is a good practice.
    print("Invalid input. Please enter an integer for age.")
```

Status : Correct *Marks : 10/10*

3. Problem Statement

A retail store requires a program to calculate the total cost of purchasing a product based on its price and quantity. The program performs validation to ensure valid inputs and handles specific error conditions using exceptions:

Price Validation: If the price is zero or less, raise a ValueError with the message: "Invalid Price".Quantity Validation: If the quantity is zero or less, raise a ValueError with the message: "Invalid Quantity".Cost Threshold: If the total cost exceeds 1000, raise RuntimeError with the message: "Excessive Cost".

Input Format

The first line of input consists of a double value, representing the price of a product.

The second line consists of an integer, representing the quantity of the product.

Output Format

If the calculation is successful, print the total cost rounded to one decimal place.

If the price is zero or less prints "Invalid Price".

If the quantity is zero or less prints "Invalid Quantity".

If the total cost exceeds 1000, prints "Excessive Cost".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 20.0
5
Output: 100.0

Answer

```
# You are using Python
try:
    # Read the price as a floating-point number
    price = float(input())
```

```
    # Read the quantity as an integer
    quantity = int(input())

    # Validate price: Must be greater than zero
    if price <= 0:
        raise ValueError("Invalid Price")

    # Validate quantity: Must be greater than zero
    if quantity <= 0:
        raise ValueError("Invalid Quantity")

    # Calculate the total cost
    total_cost = price * quantity

    # Validate total cost: Must not exceed 1000
    if total_cost > 1000:
        raise RuntimeError("Excessive Cost")

    # If all validations pass, print the total cost formatted to one decimal place
    print(f"{total_cost:.1f}")

except ValueError as e:
    # Handle ValueError exceptions (e.g., non-numeric input, or invalid price/
quantity)
    if str(e) == "Invalid Price":
        print("Invalid Price")
    elif str(e) == "Invalid Quantity":
        print("Invalid Quantity")
    else:
        # This catches general ValueError during float() or int() conversion
        # if the input isn't purely numeric.
        print("Invalid input for price or quantity.")
except RuntimeError as e:
    # Handle RuntimeError exception for excessive total cost
    if str(e) == "Excessive Cost":
        print("Excessive Cost")
except Exception as e:
    # Catch any other unexpected errors
    print(f"An unexpected error occurred: {e}")
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Tara is a content manager who needs to perform case conversions for various pieces of text and save the results in a structured manner.

She requires a program to take a user's input string, save it in a file, and then retrieve and display the string in both upper-case and lower-case versions. Help her achieve this task efficiently.

File Name: text_file.txt

Input Format

The input consists of a single line containing a string provided by the user.

Output Format

The first line displays the original string read from the file in the format: "Original String: {original_string}".

The second line displays the upper-case version of the original string in the format: "Upper-Case String: {upper_case_string}".

The third line displays the lower-case version of the original string in the format: "Lower-Case String: {lower_case_string}".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: #SpecialSymBoLs1234

Output: Original String: #SpecialSymBoLs1234
Upper-Case String: #SPECIALSYMBOLS1234
Lower-Case String: #specialsymbols1234

Answer

```
# You are using Python
file_name = "text_file.txt"

# Get input string from the user
```

```
user_string = input()

# Save the string to the file
try:
    with open(file_name, 'w') as file:
        file.write(user_string)
except IOError:
    print(f"Error: Could not write to file {file_name}")
    exit()

# Read the string from the file and perform case conversions
try:
    with open(file_name, 'r') as file:
        original_string = file.read()
except IOError:
    print(f"Error: Could not read from file {file_name}")
    exit()

# Convert to upper-case and lower-case
upper_case_string = original_string.upper()
lower_case_string = original_string.lower()

# Display the results
print(f"Original String: {original_string}")
print(f"Upper-Case String: {upper_case_string}")
print(f"Lower-Case String: {lower_case_string}")
```

Status : Correct **Marks : 10/10**

5. Problem Statement

Sophie enjoys playing with words and wants to count the number of words in a sentence. She inputs a sentence, saves it to a file, and then reads it from the file to count the words.

Write a program to determine the number of words in the input sentence.

File Name: sentence_file.txt

Input Format

The input consists of a single line of text containing words separated by spaces.

Output Format

The output displays the count of words in the sentence.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: Four Words In This Sentence

Output: 5

Answer

```
# You are using Python
file_name = "sentence_file.txt"

# Get the input sentence from the user
sentence = input()

# Save the sentence to the file
try:
    with open(file_name, 'w') as file:
        file.write(sentence)
except IOError:
    # In a real-world scenario, you might log this error or provide more user feedback.
    # For this specific problem, we just need to ensure the program proceeds.
    pass # Or handle specific error if required by problem statement

# Read the sentence from the file
try:
    with open(file_name, 'r') as file:
        read_sentence = file.read()
except IOError:
    # Handle error if file cannot be read, though unlikely after a successful write
    print("Error: Could not read sentence from file.")
    exit()

# Count the words
# The split() method by default splits by any whitespace and handles multiple spaces
```

```
# and leading/trailing spaces correctly, resulting in an empty list if the string is
empty or only spaces.
words = read_sentence.split()
word_count = len(words)

# Print the word count
print(word_count)
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Mythrayee V
Email: 241501125@rajalakshmi.edu.in
Roll no: 241501125
Phone: 9042130487
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Alice is developing a program called "Name Sorter" that helps users organize and sort names alphabetically.

The program takes names as input from the user, saves them in a file, and then displays the names in sorted order.

File Name: sorted_names.txt.

Input Format

The input consists of multiple lines, each containing a name represented as a string.

To end the input and proceed with sorting, the user can enter 'q'.

Output Format

The output displays the names in alphabetical order, each name on a new line.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: Alice Smith
John Doe
Emma Johnson
q
Output: Alice Smith
Emma Johnson
John Doe

Answer

```
# You are using Python
file_name = "sorted_names.txt"
names = []

# Read names from the user until 'q' is entered
while True:
    try:
        name_input = input()
        if name_input == 'q':
            break
        names.append(name_input)
    except EOFError:
        # Handle end-of-file condition if input is redirected
        break

# Write names to the file, each on a new line
try:
    with open(file_name, 'w') as file:
        for name in names:
            file.write(name + '\n')
except IOError:
    print(f"Error: Could not write to file {file_name}")
    exit()
```

```
# Read names back from the file, sort them, and print
read_names = []
try:
    with open(file_name, 'r') as file:
        for line in file:
            read_names.append(line.strip()) # .strip() removes the newline character
except FileNotFoundError:
    print(f"Error: File '{file_name}' not found.")
    exit()
except IOError:
    print(f"Error: Could not read from file {file_name}")
    exit()

# Sort the names alphabetically
read_names.sort()

# Print the sorted names
for name in read_names:
    print(name)
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Alex is creating an account and needs to set up a password. The program prompts Alex to enter their name, mobile number, chosen username, and desired password. Password validation criteria include:

Length between 10 and 20 characters.At least one digit.At least one special character from !@#$%^&* set. Display "Valid Password" if criteria are met; otherwise, raise an exception with an appropriate error message.

Input Format

The first line of the input consists of the name as a string.

The second line of the input consists of the mobile number as a string.

The third line of the input consists of the username as a string.

The fourth line of the input consists of the password as a string.

Output Format

If the password is valid (meets all the criteria), it will print "Valid Password"

If the password is weak (fails any one or more criteria), it will print an error message accordingly.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: John
9874563210
john
john1#nhoj
Output: Valid Password

Answer

-

Status : Skipped ***Marks : 0/10***

3. Problem Statement

Write a program to read the Register Number and Mobile Number of a student. Create user-defined exception and handle the following:

If the Register Number does not contain exactly 9 characters in the specified format(2 numbers followed by 3 characters followed by 4 numbers) or if the Mobile Number does not contain exactly 10 characters, throw an IllegalArgumentException. If the Mobile Number contains any character other than a digit, raise a NumberFormatException.If the Register Number contains any character other than digits and alphabets, throw a NoSuchElementException.If they are valid, print the message ‘valid’ or else print an Invalid message.

Input Format

The first line of the input consists of a string representing the Register number.

The second line of the input consists of a string representing the Mobile number.

Output Format

The output should display any one of the following messages:

If both numbers are valid, print "Valid".

If an exception is raised, print "Invalid with exception message: ", followed by the specific exception message.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 19ABC1001
9949596920
Output: Valid

Answer

```
# You are using Python
import re

# Define user-defined exceptions
class IllegalArgumentException(Exception):
    """Custom exception for illegal or inappropriate arguments."""
    pass

class NumberFormatException(Exception):
    """Custom exception for invalid number format."""
    pass

class NoSuchElementException(Exception):
    """Custom exception for when an element is not found or invalid character type."""
    pass

def validate_numbers(reg_num, mobile_num):
    """
```

```
    Validates register number and mobile number based on specified criteria.
    Raises custom exceptions for invalid formats or data.
    """

    # 1. Mobile Number Validation
    # Check if Mobile Number contains exactly 10 characters
    if len(mobile_num) != 10:
        raise IllegalArgumentException("Mobile Number should have exactly 10
characters.")

    # Check if Mobile Number contains any character other than a digit
    if not mobile_num.isdigit():
        raise NumberFormatException("Mobile Number should only contain digits.")

    # 2. Register Number Validation
    # Check if Register Number contains exactly 9 characters
    if len(reg_num) != 9:
        raise IllegalArgumentException("Register Number should have exactly 9
characters.")

    # Check if Register Number contains any character other than digits and
alphabets
    # This covers cases like special symbols
    if not reg_num.isalnum():
        raise NoSuchElementException("Register Number should only contain
alphanumeric characters.")

    # Check the specific format: 2 numbers, 3 characters, 4 numbers
    # Using regex for robust format checking
    # \d{2} - exactly 2 digits
    # [a-zA-Z]{3} - exactly 3 alphabetic characters (case-insensitive)
    # \d{4} - exactly 4 digits
    reg_num_pattern = r"^\d{2}[a-zA-Z]{3}\d{4}$"
    if not re.match(reg_num_pattern, reg_num):
        raise IllegalArgumentException("Register Number should have the format: 2
numbers, 3 characters, and 4 numbers.")

    # If all checks pass
    return True

# Main program execution
if __name__ == "__main__":
```

```
    register_number = input()
    mobile_number = input()

    try:
        if validate_numbers(register_number, mobile_number):
            print("Valid")
    except (IllegalArgumentException, NumberFormatException,
NoSuchElementException) as e:
        print(f"Invalid with exception message: {e}")
    except Exception as e:
        # Catch any other unexpected exceptions
        print(f"Invalid with exception message: An unexpected error occurred - {e}")
```

Status : Correct **Marks : 10/10**

4. Problem Statement

A shopkeeper is recording the daily sales of an item for N days, where the price of the item remains the same for all days. Write a program to calculate the total sales for each day and save them in a file named sales.txt that can store the data for a maximum of 30 days. Then, read the file and display the total earnings for each day.

Note: Total Earnings for each day = Number of Items sold in that day × Price of the item.

Input Format

The first line of input consists of an integer N, representing the number of days.

The second line of input consists of N space-separated integers representing the number of items sold each day.

The third line of input consists of an integer M, representing the price of the item that is common for all N days.

Output Format

If the number of days entered exceeds 30 (N > 30), the output prints "Exceeding limit!" and terminates.

Otherwise, the code reads the contents of the file and displays the total earnings

for each day on separate lines.

Contents of the file: The total earnings for N days, with each day's earnings appearing on a separate line.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
5 10 5 0
20

Output: 100
200
100
0

Answer

```
# You are using Python
file_name = "sales.txt"
MAX_DAYS = 30

try:
    # Read the number of days
    N = int(input())
except ValueError:
    # Handle cases where N is not a valid integer
    print("Invalid input for number of days.")
    exit()

# Check if N exceeds the limit
if N > MAX_DAYS:
    print("Exceeding limit!")
    exit()

try:
    # Read the space-separated number of items sold each day
```

```
    items_sold_str = input().split()
    items_sold_per_day = [int(item) for item in items_sold_str]
except ValueError:
    # Handle cases where items sold are not valid integers
    print("Invalid input for items sold per day.")
    exit()
except IndexError:
    # Handle cases where the input line for items sold is empty
    print("Invalid input for items sold per day.")
    exit()

# Ensure the number of items matches N
if len(items_sold_per_day) != N:
    # This scenario is not explicitly covered by output, but good for robustness
    print("Mismatch between number of days and items sold provided.")
    exit()

try:
    # Read the price of the item
    M = int(input())
except ValueError:
    # Handle cases where M is not a valid integer
    print("Invalid input for item price.")
    exit()

# Calculate total sales for each day
daily_earnings = []
for items_sold in items_sold_per_day:
    earnings = items_sold * M
    daily_earnings.append(earnings)

# Save the daily earnings to sales.txt
try:
    with open(file_name, 'w') as file:
        for earnings in daily_earnings:
            file.write(str(earnings) + '\n')
except IOError:
    print(f"Error: Could not write to file {file_name}")
    exit()

# Read the file and display the total earnings for each day
try:
```

```
    with open(file_name, 'r') as file:
        for line in file:
            print(line.strip()) # .strip() removes the newline character read from the
file
except FileNotFoundError:
    print(f"Error: File '{file_name}' not found.")
    exit()
except IOError:
    print(f"Error: Could not read from file {file_name}")
    exit()
```

Status : Correct **Marks : 10/10**

Rajalakshmi Engineering College

Name: Mythrayee V
Email: 241501125@rajalakshmi.edu.in
Roll no: 241501125
Phone: 9042130487
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_MCQ

Attempt : 1
Total Mark : 20
Marks Obtained : 18

Section 1 : MCQ

1. Which clause is used to clean up resources, such as closing files in Python?

Answer

finally

Status : Correct ***Marks : 1/1***

2. Fill in the blanks in the following code of writing data in binary files.

```
import _____ (1)
rec=[]
while True:
    rn=int(input("Enter"))
    nm=input("Enter")
```

```
    temp=[rn, nm]
    rec.append(temp)
    ch=input("Enter choice (y/N)")
    if ch.upper=="N":
        break
f.open("stud.dat","_____")(2)
_____.dump(rec,f)(3)
_____.close()(4)
```

Answer

(pickle,wb,pickle,f)

Status : Correct ***Marks : 1/1***

3. How do you create a user-defined exception in Python?

Answer

By creating a new class that inherits from the Exception class

Status : Correct ***Marks : 1/1***

4. What is the output of the following code?

```
class MyError(Exception):
    pass

try:
    raise MyError("Something went wrong")
except MyError as e:
    print(e)
```

Answer

Something went wrong

Status : Correct ***Marks : 1/1***

5. What happens if an exception is not caught in the except clause?

Answer

The program will display a traceback error and stop execution

Status : Correct ***Marks : 1/1***

6. What is the default value of reference_point in the following code?

file_object.seek(offset [,reference_point])

Answer

0

Status : Correct ***Marks : 1/1***

7. Fill the code to in order to read file from the current position.

Assuming exp.txt file has following 3 lines, consider current file position is beginning of 2nd line

Meri,25

John,21

Raj,20

Ouptput:

['John,21\n','Raj,20\n']

```
f = open("exp.txt", "w+")
_____(1)
print _____(2)
```

Answer

1) f.seek(0, 1)2) f.readlines()

Status : Correct ***Marks : 1/1***

8. Fill in the code in order to get the following output:

Output:

Name of the file: ex.txt

```
fo = open(_____(1), "wb")
print("Name of the file: ",_____)(2)
```

Answer

1) "ex.txt"2) fo.name

Status : Correct ***Marks : 1/1***

9. What is the output of the following code?

```
try:
    x = "hello" + 5
except TypeError:
    print("Type Error occurred")
finally:
    print("This will always execute")
```

Answer

Type Error occurredThis will always execute

Status : Correct ***Marks : 1/1***

10. What will be the output of the following Python code?

```
# Predefined lines to simulate the file content
lines = [
    "This is 1st line",
    "This is 2nd line",
    "This is 3rd line",
    "This is 4th line",
    "This is 5th line"
]

print("Name of the file: foo.txt")
```

```
# Print the first 5 lines from the predefined list
for index in range(5):
    line = lines[index]
    print("Line No %d - %s" % (index + 1, line.strip()))
```

Answer

Syntax Error

Status : Wrong ***Marks : 0/1***

11. Match the following:

a) f.seek(5,1) i) Move file pointer five characters behind from the current position

b) f.seek(-5,1) ii) Move file pointer to the end of a file

c) f.seek(0,2) iii) Move file pointer five characters ahead from the current position

d) f.seek(0) iv) Move file pointer to the beginning of a file

Answer

a-iii, b-i, c-ii, d-iv

Status : Correct ***Marks : 1/1***

12. Which of the following is true about

fp.seek(10,1)

Answer

Move file pointer ten characters ahead from the current position

Status : Correct ***Marks : 1/1***

13. What happens if no arguments are passed to the seek function?

Answer

error

Status : Wrong ***Marks : 0/1***

14. What will be the output of the following Python code?

```
f = None
for i in range (5):
    with open("data.txt", "w") as f:
        if i > 2:
            break
print(f.closed)
```

Answer

True

Status : Correct ***Marks : 1/1***

15. What is the output of the following code?

```
try:
    x = 1 / 0
except ZeroDivisionError:
    print("Caught division by zero error")
finally:
    print("Executed")
```

Answer

Caught division by zero errorExecuted

Status : Correct ***Marks : 1/1***

16. What is the difference between r+ and w+ modes?

Answer

in r+ the pointer is initially placed at the beginning of the file and the pointer is at the end for w+

Status : Correct *Marks : 1/1*

17. What is the correct way to raise an exception in Python?

Answer

raise Exception()

Status : Correct *Marks : 1/1*

18. What is the purpose of the except clause in Python?

Answer

To handle exceptions during code execution

Status : Correct *Marks : 1/1*

19. How do you rename a file?

Answer

os.rename(existing_name, new_name)

Status : Correct *Marks : 1/1*

20. Which of the following is true about the finally block in Python?

Answer

The finally block is always executed, regardless of whether an exception occurs or not

Status : Correct *Marks : 1/1*

Rajalakshmi Engineering College

Name: Mythrayee V
Email: 241501125@rajalakshmi.edu.in
Roll no: 241501125
Phone: 9042130487
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_PAH

Attempt : 1
Total Mark : 30
Marks Obtained : 20

Section 1 : Coding

1. Problem Statement

John is a data analyst who often works with text files. He needs a program that can analyze the contents of a text file and count the number of times a specific character appears in the file.

John wants a simple program that allows him to specify a file and a character to count within that file.

Input Format

The first line of input consists of the file's name to be analyzed.

The second line of the input consists of the string they want to write within the file.

The third line of the input consists of a character to count within the file.

Output Format

If the character is found, the output displays "The character 'X' appears {Y} times in the file." where X is the character and Y i the count,

If the character does not appear in the file, the output displays "Character not found."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: test.txt
This is a test file to check the character count.
e

Output: The character 'e' appears 5 times in the file.

Answer

```
import os

file_name = input()

file_content = input()

char_to_count = input()

try:
    with open(file_name, 'w') as file:
        file.write(file_content)
except IOError:
    print(f"Error: Could not write to file {file_name}")
    exit()

try:
    with open(file_name, 'r') as file:
        content_from_file = file.read()
except FileNotFoundError:
```

```
    print(f"Error: File '{file_name}' not found.")
    exit()
except IOError:
    print(f"Error: Could not read from file {file_name}")
    exit()

content_for_count = content_from_file.lower()
char_to_count_lower = char_to_count.lower()

count = content_for_count.count(char_to_count_lower)


if count > 0:

    print(f"The character '{char_to_count}' appears {count} times in the file.")
else:

    print("Character not found in the file.")
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Reeta is playing with numbers. Reeta wants to have a file containing a list of numbers, and she needs to find the average of those numbers. Write a program to read the numbers from the file, calculate the average, and display it.

File Name: user_input.txt

Input Format

The input file will contain a single line of space-separated numbers (as a string).

These numbers may be integers or decimals.

Output Format

If all inputs are valid numbers, the output should print: "Average of the numbers is: X.XX" (where X.XX is the computed average rounded to two decimal places)

If the input contains invalid data, print: "Invalid data in the input."

Refer to the sample output for format specifications.

Sample Test Case

Input: 1 2 3 4 5

Output: Average of the numbers is: 3.00

Answer

```
# You are using Python
file_name = "user_input.txt"

# Read the numbers from the input
input_line = input()

# Write the input line to the file
try:
    with open(file_name, 'w') as file:
        file.write(input_line)
except IOError:
    # This error handling is basic. In a real application, you might log or alert.
    print("Error writing to file.")
    exit()

# Read the numbers from the file and calculate the average
try:
    with open(file_name, 'r') as file:
        numbers_str = file.read().strip().split()

    if not numbers_str: # Handle empty file or only whitespace
        print("Invalid data in the input.")
        exit()

    total_sum = 0
    count = 0
    for num_str in numbers_str:
        total_sum += float(num_str)
        count += 1
```

```
    if count > 0:
        average = total_sum / count
        print(f"Average of the numbers is: {average:.2f}")
    else:
        # This case should theoretically be caught by 'if not numbers_str:' but for
safety
        print("Invalid data in the input.")

except FileNotFoundError:
    print("Error: Input file not found.")
except ValueError:
    # This will catch errors if float(num_str) fails, meaning invalid numeric data
    print("Invalid data in the input.")
except IOError:
    print("Error reading from file.")
```

Status : Correct **Marks : 10/10**

3. Problem Statement

Peter manages a student database and needs a program to add students. For each student, Alex inputs their ID and name. The program checks for duplicate IDs and ensures the database isn't full.

If a duplicate or a full database is detected, an appropriate error message is displayed. Otherwise, the student is added, and a confirmation message is shown. The database has a maximum capacity of 30 students, and each student must have a unique ID.

Input Format

The first line contains an integer n, representing the number of students to be added to the school database.

The next n lines each contain two space-separated values, representing the student's ID (integer) and the student's name (string).

Output Format

The output will depend on the actions performed in the code.

If a student is added to the database, the output will display: "Student with ID [ID number] added to the database."

If there is an exception due to a duplicate student ID, the output will display: "Exception caught. Error: Student ID already exists."

If there is an exception due to the database being full, the output will display: "Exception caught. Error: Student database is full."

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 3
16 Sam
87 Sabari
43 Dani

Output: Student with ID 16 added to the database.
Student with ID 87 added to the database.
Student with ID 43 added to the database.

Answer

-

Status : Skipped ***Marks : 0/10***