# Rajalakshmi Engineering College

Name: ANGELIN SHREYA
Email: 241801022@rajalakshmi.edu.in
Roll no: 241801022
Phone: 8610007914
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 2_COD_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 30

## Section 1 : Coding

1. Problem Statement

Ethan, a curious mathematician, is fascinated by perfect numbers. A perfect number is a number that equals the sum of its proper divisors (excluding itself). Ethan wants to identify all perfect numbers within a given range.

Help him write a program to list these numbers.

***Input Format***

The first line of input consists of an integer start, representing the starting number of the range.

The second line consists of an integer end, representing the ending number of the range.

***Output Format***

The output prints all perfect numbers in the range, separated by a space.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1
100

Output: 6 28

***Answer***

```
# You are using Python
```

***Status :*** Wrong ***Marks : 0/10***

2. Problem Statement

You work as an instructor at a math enrichment program, and your goal is to develop a program that showcases the concept of using control statements to manipulate loops. Your task is to create a program that takes an integer 'n' as input and prints the squares of even numbers from 1 to 'n', while skipping odd numbers.

***Input Format***

The input consists of a single integer, which represents the upper limit of the range.

***Output Format***

The output displays the square of even numbers from 1 to 'n' separated by lines.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 10
Output: 4
16
36
64
100

***Answer***

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

for i in range(1 , n+1):
    if i % 2 == 0:
        print(i**2)
    else:
        pass
```

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

3. Problem Statement

John, a software developer, is analyzing a sequence of numbers within a given range to calculate their digit sum. However, to simplify his task, he excludes all numbers that are palindromes (numbers that read the same backward as forward).

Help John find the total sum of the digits of non-palindromic numbers in the range [start, end] (both inclusive).

Example:

Input:

10

20

Output:

55

Explanation:

Range [10, 20]: Non-palindromic numbers are 10, 12, 13, 14, 15, 16, 17, 18, 19 and 20.

Digit sums: 1+0 + 1+2 + 1+3 + 1+4 + 1+5 + 1+6 + 1+7 + 1+8 + 1+9 + 2+0 = 55.

Output: 55

***Input Format***

The first line of input consists of an integer, representing the starting number of the range.

The second line of input consists of an integer, representing the ending number of the range.

***Output Format***

The output prints a single integer, representing the total sum of the digits of all non-palindromic numbers in the range.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 10
20
Output: 55

***Answer***

```
# You are using Python
n1 = int(input())
n2 = int(input())

total_sum = 0
for i in range (n1 , n2+1):
    str_i = str(i)
    if str_i != str_i[::-1]:
        digit_sum = sum(int(digit) for digit in str_i)
        total_sum += digit_sum
```

```
print(total_sum)
```

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

4. Problem Statement

As a junior developer working on a text analysis project, your task is to create a program that displays the consonants in a sentence provided by the user, separated by spaces.

You need to implement a program that takes a sentence as input and prints the consonants while skipping vowels and non-alphabetic characters using only control statements.

***Input Format***

The input consists of a string representing the sentence.

***Output Format***

The output displays space-separated consonants present in the sentence.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: Hello World!
Output: H l l W r l d

***Answer***

```
# You are using Python
word = str(input())
vowel = 'aeiouAEIOU'
for i in word:
    if i.isalpha() and i not in vowel:
        print(i, end = " ")
    else:
            pass
```

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

5. Problem Statement

Emma, a mathematics enthusiast, is exploring a range of numbers and wants to count how many of them are not Fibonacci numbers.

Help Emma determine the count of non-Fibonacci numbers within the given range [start, end] using the continue statement.

***Input Format***

The first line of input consists of an integer, representing the starting number of the range.

The second line consists of an integer, representing the ending number of the range.

***Output Format***

The output prints a single integer, representing the count of numbers in the range that are not Fibonacci numbers.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1
10
Output: 5

***Answer***

```
# You are using Python
n1= int(inout())
n2= int(input())
a,b=0,1
fib=set()
```

***Status :*** Wrong **Marks : 0/10**

# Rajalakshmi Engineering College

Name: ANGELIN SHREYA
Email: 241801022@rajalakshmi.edu.in
Roll no: 241801022
Phone: 8610007914
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

### REC_Python_Week 1_COD

Attempt : 1
Total Mark : 5
Marks Obtained : 3

### Section 1 : Coding

1. Problem Statement

Quentin, a mathematics enthusiast, is exploring the properties of numbers. He believes that for any set of four consecutive integers, calculating the average of their fourth powers and then subtracting the product of the first and last numbers yields a constant value.

To validate his hypothesis, check if the result is indeed constant and display.

Example:

Input:

5

Output:

Constant value: 2064.5

Explanation:

Find the Average:

Average: (625 + 1296 + 2401 + 4096)/4 = 2104.5

Now, we calculate the product of a and (a + 3):

Product = 5 × (5 + 3) = 5 × 8 = 40

Final result: 2104.5 - 40 = 2064.5

***Input Format***

The input consists of an integer a, representing the first of four consecutive integers.

***Output Format***

The output displays "Constant value: " followed by the computed result based on Quentin's formula.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 5
Output: Constant value: 2064.5

***Answer***

```
# You are using Python
a=int(input())
z1=a**4
y1=(a+1)**4
q1=(a+2)**4
w1=(a+3)**4
s=(z1+y1+q1+w1)/4
p=a*(a+3)
f=s-p
print("Constant value:",f)
```

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

2. Problem Statement

In a family, two children receive allowances based on the gardening tasks they complete. The older child receives an allowance rate of Rs.5 for each task, with a base allowance of Rs.50. The younger child receives an allowance rate of Rs.3 for each task, with a base allowance of Rs.30.

Your task is to calculate and display the allowances for the older and younger children based on the number of gardening tasks they complete, along with the total allowance for both children combined.

***Input Format***

The first line of input consists of an integer n, representing the number of chores completed by the older child.

The second line consists of an integer m, representing the number of chores completed by the youngest child.

***Output Format***

The first line of output displays "Older child allowance: Rs." followed by an integer representing the allowance calculated for the older sibling.

The second line displays "Younger child allowance: Rs." followed by an integer representing the allowance calculated for the youngest sibling.

The third line displays "Total allowance: Rs." followed by an integer representing the sum of both siblings' allowances.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 10
5
Output: Older child allowance: Rs.100

Younger child allowance: Rs.45
Total allowance: Rs.145

***Answer***

```
# You are using Python
n=int(input())
m=int(input())
x1=50+n*5
y1=30+m*3
z1=x1+y1
print("Older child allowance: Rs.",x1)
print("Younger child allowance: Rs.",y1)
print("Total allowance : Rs.",z1)
```

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

3. Problem Statement

A company has hired two employees, Alice and Bob. The company wants to swap the salaries of both employees. Alice's salary is an integer value and Bob's salary is a floating-point value.

Write a program to swap their salaries and print the new salary of each employee.

***Input Format***

The first line of input consists of an integer N, representing Alice's salary.

The second line consists of a float value F, representing Bob's salary.

***Output Format***

The first line of output displays "Initial salaries:"

The second line displays "Alice's salary = N", where N is Alice's salary.

The third line of output displays "Bob's salary = F", where F is Bob's salary.

After a new line space, the following line displays "New salaries after swapping:"

The next line displays "Alice's salary = X", where X is the swapped salary.

The last line displays "Bob's salary = Y", where Y is the swapped salary.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 10000
15400.55
Output: Initial salaries:
Alice's salary = 10000
Bob's salary = 15400.55

New salaries after swapping:
Alice's salary = 15400.55
Bob's salary = 10000

***Answer***

```
# You are using Python
I=float(input())
F=int(I)
print(f"The integer value of {I} is : {F}")
```

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

4. Problem Statement

Bob, the owner of a popular bakery, wants to create a special offer code for his customers. To generate the code, he plans to combine the day of the month with the number of items left in stock.

Help Bob to encode these two values into a unique offer code.

Note: Use the bitwise operator to calculate the offer code.

Example

Input:

15
9

Output:

Offer code: 6

Explanation:

Given the day of the month 15th day (binary 1111) and there are 9 items left (binary 1001), the offer code is calculated as 0110 which is 6.

***Input Format***

The first line of input consists of an integer D, representing the day of the month.

The second line consists of an integer S, representing the number of items left in stock.

***Output Format***

The output displays "Offer code: " followed by an integer representing the encoded offer code.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 15
9

Output: Offer code: 6

***Answer***

```
# You are using Python
N=int(intput())
S=input().strip()
rl,im=map(float,input().split())
if im>=0:
    complex_str=f"({rl}+{im}j)"
else:
    complex_str=f"({rl}{im}j)"
```

```
print(complex_str)
print(f"{N},{s},{rl},{im}")
```

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

5. Problem Statement

A science experiment produces a decimal value as the result. However, the scientist needs to convert this value into an integer so that it can be used in further calculations.

Write a Python program that takes a floating-point number as input and converts it into an integer.

***Input Format***

The input consists of a floating point number, F.

***Output Format***

The output prints "The integer value of F is: {result}", followed by the integer number equivalent to the floating point number.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 10.36
Output: The integer value of 10.36 is: 10

***Answer***

```
# You are using Python
I=float(input())
F=int(I)
print(f"The integer value of {I} is : {F}")
```

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

# Rajalakshmi Engineering College

Name: ANGELIN SHREYA
Email: 241801022@rajalakshmi.edu.in
Roll no: 241801022
Phone: 8610007914
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 3_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 40

## Section 1 : Coding

1. Problem Statement

Alex is working on a Python program to manage a list of elements. He needs to append multiple elements to the list and then remove an element from the list at a specified index.

Your task is to create a program that helps Alex manage the list. The program should allow Alex to input a list of elements, append them to the existing list, and then remove an element at a specified index.

***Input Format***

The first line contains an integer n, representing the number of elements to be appended to the list.

The next n lines contain integers, representing the elements to be appended to the list.

The third line of input consists of an integer M, representing the index of the element to be popped from the list.

***Output Format***

The first line of output displays the original list.

The second line of output displays the list after popping the element of the index M.

The third line of output displays the popped element.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 5
64
98
-1
5
26
3
Output: List after appending elements: [64, 98, -1, 5, 26]
List after popping last element: [64, 98, -1, 26]
Popped element: 5

***Answer***

```
# You are using Python
```

***Status :*** Wrong ***Marks : 0/10***

2. Problem Statement

You have a string containing a phone number in the format "(XXX) XXX-XXXX". You need to extract the area code from the phone number and create a new string that contains only the area code.

Write a Python program for the same.

Note

(XXX) - Area code

XXX-XXXX - Phone number

***Input Format***

The input consists of a string, representing the phone number in the format "(XXX) XXX-XXXX".

***Output Format***

The output displays "Area code: " followed by a string representing the area code for the given phone number.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: (123) 456-7890
Output: Area code: 123

***Answer***

```
# You are using Python
n = input()
print(f"Area code: {n[1]+n[2]+n[3]}")
```

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

3. Problem Statement

Dhruv wants to write a program to slice a given string based on user-defined start and end positions.

The program should check whether the provided positions are valid and then return the sliced portion of the string if the positions are within the string's length.

***Input Format***

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

The second line consists of the start position (0-based index) as an integer.

The third line consists of the end position (0-based index) as an integer.

***Output Format***

The output displays the following format:

If the start and end positions are valid, print the sliced string.

If the start and end positions are invalid, print "Invalid start and end positions".

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: pythonprogramming
0
5
Output: python

***Answer***

```
# You are using Python
s=input()
a=int(input())
b=int(input())
if (b<=len(s) and a<=len(s) and a<b):
   print(s[a:b+1])
else:
   print("Invalid start and end positions")
```

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

4. Problem Statement

Ram is working on a program to manipulate strings. He wants to create a program that takes two strings as input, reverses the second string, and then concatenates it with the first string.

Ram needs your help to design a program.

***Input Format***

The input consists of two strings in separate lines.

***Output Format***

The output displays a single line containing the concatenated string of the first string and the reversed second string.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: hello
word

Output: hellodrow

***Answer***

```
# You are using Python
a= input()
b= input()
print(a+b[::-1])
```

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

5. Problem Statement

Given a list of positive and negative numbers, arrange them such that all negative integers appear before all the positive integers in the array. The order of appearance should be maintained.

Example

Input:

[12, 11, -13, -5, 6, -7, 5, -3, -6]

Output:

List = [-13, -5, -7, -3, -6, 12, 11, 6, 5]

Explanation:

The output is the arranged list where all the negative integers appear before the positive integers while maintaining the original order of appearance.

***Input Format***

The input consists of a single line containing a list of integers enclosed in square brackets separated by commas.

***Output Format***

The output displays "List = " followed by an arranged list of integers as required, separated by commas and enclosed in square brackets.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: [12, 11, -13, -5, 6, -7, 5, -3, -6]
Output: List = [-13, -5, -7, -3, -6, 12, 11, 6, 5]

***Answer***

```
# You are using Python
l = eval(input())
t = []
for i in l:
    if i< 0:
        t.append(i)

for j in l:
    if j not in t:
        t.append(j)
```

```
print(f"List = {t}")
```

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

# Rajalakshmi Engineering College

Name: ANGELIN SHREYA
Email: 241801022@rajalakshmi.edu.in
Roll no: 241801022
Phone: 8610007914
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 4_COD_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 46

## Section 1 : Coding

1. Problem Statement

Implement a program that needs to identify Armstrong numbers. Armstrong numbers are special numbers that are equal to the sum of their digits, each raised to the power of the number of digits in the number.

Write a function is_armstrong_number(number) that checks if a given number is an Armstrong number or not.

Function Signature: armstrong_number(number)

***Input Format***

The first line of the input consists of a single integer, n, representing the number to be checked.

***Output Format***

The output should consist of a single line that displays a message indicating whether the input number is an Armstrong number or not.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 153

Output: 153 is an Armstrong number.

***Answer***

```
def isamst(n):
    s=str(n)
    l=len(s)
    summ=0
    for i in s:
        summ+=int(i)**3
    return summ
n=int(input())
result=isamst(n)
if result==n:
    print(n,"is an Armstrong number.")
else:
    print(n,"is not an Armstrong number.")
```

***Status :*** Partially correct ***Marks : 6/10***

2. Problem Statement

Sara is developing a text-processing tool that checks if a given string starts with a specific character or substring. She needs to implement a function that accepts a string and a character (or substring), and returns True if the string starts with the provided character/substring, or False otherwise.

Write a program that uses a lambda function to help Sara perform this check.

***Input Format***

The first line contains a string `str` representing the main string to be checked.

The second line contains a string `n`, which is the character or substring to check if the main string starts with it.

***Output Format***

The first line of output prints "True" if the string starts with the given character/ substring, otherwise prints "False".

Refer to the sample for the formatting specifications.

***Sample Test Case***

Input: Examly
e

Output: False

***Answer***

```
# You are using Python
s=input()
n=input()
res=lambda s,n: s.index(n)==0 if n in s else False
print(res(s,n))
```

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

3. Problem Statement

Imagine you are building a messaging application, and you want to know the length of the messages sent by the users. You need to create a program that calculates the length of a message using the built-in function len().

***Input Format***

The input consists of a string representing the message.

***Output Format***

The output prints an integer representing the length of the entered message.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: hello!!

Output: 7

***Answer***

```
# You are using Python
s=input()
def count(s):
    s=len(s)
    return s
print(count(s))
```

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

4. Problem Statement

Sneha is building a more advanced exponential calculator. She wants to implement a program that does the following:

Calculates the result of raising a given base to a specific exponent using Python's built-in pow() function.Displays all intermediate powers from base¹ to base^exponent as a list.Calculates and displays the sum of these intermediate powers.

Help her build this program to automate her calculations.

***Input Format***

The input consists of line-separated two integer values representing base and exponent.

***Output Format***

The first line of the output prints the calculated result of raising the base to the exponent.

The second line prints a list of all powers from base^1 to base^exponent.

The third line prints the sum of all these powers.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 2
3
Output: 8
[2, 4, 8]
14

***Answer***

```
# You are using Python
def calc(n,m):
    p=pow(n,m)
    expo=[]
    for i in range(1,m+1):
        expo1=pow(n,i)
        expo.append(expo1)
    summ=sum(expo)
    return p,expo,summ
n=int(input())
m=int(input())
for i in calc(n,m):
    print(i)
```

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

5. Problem Statement

Imagine you are developing a text analysis tool for a cybersecurity company. Your task is to create a function that analyzes input strings to

categorize and count the characters into four categories: uppercase letters, lowercase letters, digits, and special characters. The company needs this tool to process log files and identify potential security threats.

Function Signature: analyze_string(input_string)

***Input Format***

The input consists of a single string (without space), which may include uppercase letters, lowercase letters, digits, and special characters.

***Output Format***

The first line contains an integer representing the count of uppercase letters in the format "Uppercase letters: [count]".

The second line contains an integer representing the count of lowercase letters in the format "Lowercase letters: [count]".

The third line contains an integer representing the count of digits in the format "Digits: [count]".

The fourth line contains an integer representing the count of special characters in the format "Special characters: [count]".

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: Hello123

Output: Uppercase letters: 1
Lowercase letters: 4
Digits: 3
Special characters: 0

***Answer***

```
def analyze_string(input_string):
    u=0
    l=0
    d=0
```

```
    s=0
    sp=input_string
    for i in sp:
        if i.isupper():
            u+=1
        elif i.islower():
            l+=1
        elif i.isdigit():
            d+=1
        else:
            s+=1
    return u,l,d,s

input_string = input()
uppercase_count, lowercase_count, digit_count, special_count =
analyze_string(input_string)

print("Uppercase letters:", uppercase_count)
print("Lowercase letters:", lowercase_count)
print("Digits:", digit_count)
print("Special characters:", special_count)
```

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

# Rajalakshmi Engineering College

Name: ANGELIN SHREYA
Email: 241801022@rajalakshmi.edu.in
Roll no: 241801022
Phone: 8610007914
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

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

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
# Read input string from user
input_str = input()


with open("text_file.txt", "w") as f:
    f.write(input_str)

# Read the string back from the file
with open("text_file.txt", "r") as f:
    original_str = f.read()

# Print outputs as required
print(f"Original String: {original_str}")
print(f"Upper-Case String: {original_str.upper()}")
print(f"Lower-Case String: {original_str.lower()}")
```

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

2. 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:
    price = float(input())
```

```
    quantity = int(input())

    if price <= 0:
        raise ValueError("Invalid Price")
    if quantity <= 0:
        raise ValueError("Invalid Quantity")

    total_cost = price * quantity

    if total_cost > 1000:
        raise RuntimeError("Excessive Cost")

    print(round(total_cost, 1))

except ValueError as ve:
    print(ve)
except RuntimeError as re:
    print(re)
```

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

3. 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 NotEligibleException(Exception):
    pass

age = int(input())

try:
    if age < 18:
        raise NotEligibleException
    print("Eligible to vote")
except NotEligibleException:
    print("Not eligible to vote")
```

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

4. 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
sentence = input()

with open("sentence_file.txt", "w") as f:
    f.write(sentence)

with open("sentence_file.txt", "r") as f:
    content = f.read()

words = content.split()
print(len(words))
```

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

5. 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:
    n_str = input()
    try:
        n = int(n_str)
    except:
        print("Error: You must enter a numeric value.")
        exit()

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

    arr = []
    for _ in range(n):
        elem_str = input()
        try:
            elem = int(elem_str)
```

```
            arr.append(elem)
        except:
            print("Error: You must enter a numeric value.")
            exit()

    average = sum(arr) / n
    print(f"The average is: {average:.2f}")

except Exception as e:
    print(e)
```

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

# Rajalakshmi Engineering College

Name: ANGELIN SHREYA
Email: 241801022@rajalakshmi.edu.in
Roll no: 241801022
Phone: 8610007914
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 7_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 42.5

## Section 1 : Coding

1. Problem Statement

A company tracks the monthly sales data of various products. You are given a table where each row represents a product and each column represents its monthly sales in sequential months.

Your task is to compute the cumulative monthly sales for each product using numpy, where the cumulative sales for a month is the total sales from month 1 up to that month.

***Input Format***

The first line of input consists of two integer values, products and months, separated by a space.

Each of the next products lines consists of months integer values representing the monthly sales data of a product.

***Output Format***

The first line of output prints: "Cumulative Monthly Sales:"

The second line of output prints: the 2D numpy array cumulative_array that contains the cumulative sales data for each product.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 2 4
10 20 30 40
5 15 25 35

Output: Cumulative Monthly Sales:
[[ 10  30  60 100]
 [  5  20  45  80]]

***Answer***

```
# You are using Python
import numpy as np

# Read number of products and months
products, months = map(int, input().split())

# Read sales data for each product
sales_data = []
for _ in range(products):
    row = list(map(int, input().split()))
    sales_data.append(row)

# Convert to NumPy array
sales_array = np.array(sales_data)

# Compute cumulative sum along axis=1 (i.e., row-wise)
cumulative_array = np.cumsum(sales_array, axis=1)

# Output
print("Cumulative Monthly Sales:")
print(cumulative_array)
```

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

2. Problem Statement

Rekha works in hospital data management and receives patient records with missing or incomplete data. She needs to clean the records by performing the following tasks:

Calculate the mean of the available Age values.Replace any missing (NaN) values in the Age column with this mean age.Remove any rows where the Diagnosis value is missing (NaN).Reset the DataFrame index after removing these rows.

Implement this data cleaning task using the pandas package.

***Input Format***

The first line of input contains an integer n representing the number of patient records.

The second line contains the CSV header — comma-separated column names (e.g., "Name,Age,Diagnosis,Gender").

The next n lines each contain one patient record in comma-separated format.

***Output Format***

The first line of output is the text:

Cleaned Hospital Records:

The next lines print the cleaned pandas DataFrame (as produced by print(cleaned_df)).

This will include the updated values of the Age column (with missing ages filled by the mean age), and any rows with missing Diagnosis removed.

The DataFrame will be displayed using the default pandas print() representation.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 5
PatientID,Name,Age,Diagnosis
1,John Doe,45,Flu
2,Jane Smith,,Cold
3,Bob Lee,50,
4,Alice Green,38,Fever
5,Tom Brown,,Infection

Output: Cleaned Hospital Records:

| | PatientID | Name | Age | Diagnosis |
|---|---|---|---|---|
| 0 | 1 | John Doe | 45.000000 | Flu |
| 1 | 2 | Jane Smith | 44.333333 | Cold |
| 2 | 4 | Alice Green | 38.000000 | Fever |
| 3 | 5 | Tom Brown | 44.333333 | Infection |

***Answer***

```
# You are using Python
import pandas as pd
import numpy as np
import sys

# Read input
n = int(input())
columns = input().strip().split(',')

# Read the next n lines into a list of records
data = [input().strip().split(',') for _ in range(n)]

# Create DataFrame
df = pd.DataFrame(data, columns=columns)

# Convert 'Age' column to numeric (handle missing/empty strings as NaN)
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

# Calculate mean of available ages
mean_age = df['Age'].mean()

# Fill missing Age values with mean
df['Age'] = df['Age'].fillna(mean_age)
```

```
# Remove rows with missing Diagnosis
df = df[df['Diagnosis'].notna() & (df['Diagnosis'] != '')]

# Reset index
df = df.reset_index(drop=True)

# Output
print("Cleaned Hospital Records:")
print(df)
```

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

3. Problem Statement

Sita is analyzing her company's daily sales data to find all sales values that are multiples of 5 and exceed 100. She wants to filter these specific sales values from the list.

Help her to implement the task using the numpy package.

Formula:

To filter sales values:

Select all values s from sales such that (s % 5 == 0) and (s > 100)

***Input Format***

The first line of input consists of an integer value, n, representing the number of sales entries.

The second line of input consists of n floating-point values, sales, separated by spaces, representing daily sales figures.

***Output Format***

The output prints: filtered_sales

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 5
50.0 100.0 105.0 150.0 99.0
Output: [105. 150.]

***Answer***

```
# You are using Python
import numpy as np

# Read number of sales entries
n = int(input())

# Read the sales values and convert to NumPy array
sales = np.array(list(map(float, input().split())))

# Filter values: multiples of 5 and greater than 100
filtered_sales = sales[(sales % 5 == 0) & (sales > 100)]

# Output
print(filtered_sales)
```

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

4. Problem Statement

Alex is a data scientist analyzing the relationship between two financial indicators over time. He has collected two time series datasets representing daily values of these indicators over several months. Alex wants to understand how these two indicators correlate at different time lags to identify possible leading or lagging behaviors.

Your task is to help Alex compute the cross-correlation of these two time series using numpy, so he can analyze the similarity between the two signals at various time shifts.

***Input Format***

The first line of input consists of space-separated float values representing the first time series, array1.

The second line of input consists of space-separated float values representing the second time series, array2.

***Output Format***

The first line of output prints: "Cross-correlation of the two time series:"

The second line of output prints: the 1D numpy array cross_corr representing the cross-correlation of array1 and array2 across different lags.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 1.0 2.0 3.0
4.0 5.0 6.0
Output: Cross-correlation of the two time series:
[ 6. 17. 32. 23. 12.]

***Answer***

```
# You are using Python
import numpy as np

array1 = np.array(list(map(float, input().split())))
array2 = np.array(list(map(float, input().split())))

cross_corr = np.correlate(array1, array2, mode='full')
mid = len(cross_corr) // 2
result = cross_corr[mid:]

print("Cross-correlation of the two time series:")
print(result)
```

***Status :*** Partially correct ***Marks : 2.5/10***

5. Problem Statement

Sita works as a sales analyst and needs to analyze monthly sales data for different cities. She receives lists of cities, months, and corresponding

sales values and wants to create a pandas DataFrame using a MultiIndex of cities and months.

Help her to implement this task and calculate total sales for each city.

***Input Format***

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

The second line of input consists of n space-separated city names.

The third line of input consists of n space-separated month names.

The fourth line of input consists of n space-separated float values representing sales for each city-month combination.

***Output Format***

The first line of output prints: "Monthly Sales Data with MultiIndex:"

The next lines print the DataFrame with MultiIndex (City, Month) and their corresponding sales values.

The following line prints: "\nTotal Sales Per City:"

The final lines print the total sales per city, computed by grouping the sales data on city names.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 4
NYC NYC LA LA
Jan Feb Jan Feb
100 200 300 400
Output: Monthly Sales Data with MultiIndex:
Sales
City Month
NYC Jan 100.0

```
    Feb    200.0
LA  Jan    300.0
    Feb    400.0

Total Sales Per City:
     Sales
City
LA    700.0
NYC   300.0
```

***Answer***

```
# You are using Python
import pandas as pd

n = int(input())
cities = input().split()
months = input().split()
sales = list(map(float, input().split()))

index = pd.MultiIndex.from_tuples(zip(cities, months), names=["City", "Month"])
df = pd.DataFrame({"Sales": sales}, index=index)

print("Monthly Sales Data with MultiIndex:")
print(df)

total_sales = df.groupby(level="City").sum()

print("\nTotal Sales Per City:")
print(total_sales)
```

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