

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 1\_MCQ

Attempt : 1  
Total Mark : 15  
Marks Obtained : 11

#### **Section 1 : MCQ**

1. Which is the correct operator for power(xy)?

**Answer**

x\*\*y

**Status :** Correct

**Marks :** 1/1

2. Evaluate the expression given below if A= 16 and B = 15

A % B // A

**Answer**

0.0

**Status :** Wrong

**Marks :** 0/1

3. What is the output of the following number conversion?

```
z = complex(1.25)
print(z)
```

**Answer**

(1.25+0j)

**Status : Correct**

**Marks : 1/1**

4. What is the output of the following program?

```
print((1, 2) + (3, 4))
```

**Answer**

(1, 2, 3, 4)

**Status : Correct**

**Marks : 1/1**

5. Which of the following represents the bitwise XOR operator?

**Answer**

`^`

**Status : Correct**

**Marks : 1/1**

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

```
x = int(34.56 - 2 * 2)
print(x)
```

**Answer**

30

**Status : Correct**

**Marks : 1/1**

7. Which of the following expressions results in an error?

**Answer**

int('10.8')&nbsp;

**Status : Correct**

**Marks : 1/1**

8. What will be the output for the below code?

```
x=15  
y=12  
print(x&y)
```

**Answer**

12

**Status : Correct**

**Marks : 1/1**

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

```
X = 2+9*((3*12)-8)/10  
print(X)
```

**Answer**

27.2&nbsp;

**Status : Correct**

**Marks : 1/1**

10. What will the following code output?

```
z = 3 + 4j  
print(abs(z))
```

**Answer**

3.0

**Status : Wrong**

**Marks : 0/1**

11. Which of the following is an example of the type casting?

**Answer**

int(2)

**Status : Wrong**

**Marks : 0/1**

12. Which of the following operators has its associativity from right to left?

**Answer**

\*\*

**Status : Correct**

**Marks : 1/1**

13. The value of the expressions  $4/(3*(2-1))$  and  $4/3*(2-1)$  is the same.  
True or False?

**Answer**

True

**Status : Correct**

**Marks : 1/1**

14. What is the value of the following expression?

float(22//3+3/3)

**Answer**

8.0

**Status : Correct**

**Marks : 1/1**

15. What is the value of x in the following program?

```
x = int(43.55+2/2)
print(x)
```

**Answer**

23



Scanned with OKEN Scanner

**Status : Wrong**

**Marks : 0/1**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 1\_COD

Attempt : 1

Total Mark : 5

Marks Obtained : 3

#### **Section 1 : Coding**

##### **1. 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 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**

```
def swap_salaries():  
  
    alice_salary=int(input())  
    bob_salary=float(input())
```

```
        print("Initial salaries:\n")  
        print(f"Alic's salary={alice_salary}\n")  
        print(f"Bob's salary={bob_salary:.2f}\n")
```

```
print("New salaries after swap\n")
temp_salary= alice_salary
alice_salary=bob_salary
bob_salary =temp_salary
print(f"Alic's salary={alice_salary:.2f}\n")
print(f"Bob's salary={bob_salary}\n")
```

**Status : Wrong**

**Marks : 0/1**

## 2. 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***

```
def generate_offer_code(day,stock):  
    return day^stock  
day=int(input())  
stock=int(input())  
offer_code = generate_offer_code(day,stock)  
print("offer code:",offer_code)
```

**Status : Correct**

**Marks : 1/1**

### **3. 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**

```
def calculate_allowance(tasks_older,tasks_younger):
    older_allowance = 50+(tasks_older*5)
    younger_allowance = 30+(tasks_younger*3)
    total_allowance = older_allowance+younger_allowance
    return older_allowance,younger_allowance,total_allowance

tasks_older=int(input())
tasks_younger=int(input())
older_allowance,younger_allowance,total_allowance=calculate_allowance(tasks_
older,tasks_younger)

print(f"older child allowance: Rs.{older_allowance}")
print(f"younger child allowance: Rs.{younger_allowance}")
print(f"Total allowance: Rs{total_allowance}")
```

Status : Wrong

Marks : 0/1

#### 4. 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***

```
def convert_to_integer(f):
    return int(f)
f = float(input())
result = convert_to_integer(f)
print(f"The integer value of {f} is: {result}")
```

Status : Correct

Marks : 1/1

## 5. 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:

$$\text{Average: } (625 + 1296 + 2401 + 4096)/4 = 2104.5$$

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

$$\text{Product} = 5 \times (5 + 3) = 5 \times 8 = 40$$

$$\text{Final result: } 2104.5 - 40 = 2064.5$$

### ***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**

```
def calculate_constant_value(a):
    a4=a**4
    a1_4=(a+1)**4
    a2_4=(a+2)**4
    a3_4=(a+3)**4

    average=(a4+a1_4+a2_4+a3_4)/4.0
    product= a*(a+3)
    result = average - product
    return result

a=int(input())
constant_value = calculate_constant_value(a)
print("Constant value:",constant_value)
```

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 1\_CY

Attempt : 2  
Total Mark : 40  
Marks Obtained : 40

#### **Section 1 : Coding**

##### **1. Problem Statement**

Liam and his friends are sharing the cost of a group purchase. The total cost of the purchase is subject to a 10% discount. One of the friends receives a 35% bonus, which means they will pay a larger portion of the discounted cost. The remaining cost is then divided equally among the other friends.

Write a program to:

Calculate the total cost after applying a 10% discount. Determine the amount paid by the friend who receives a 35% bonus. Calculate the amount each of the other friends will pay.

##### ***Input Format***

The first line of input consists of a float value  $f$ , representing the total cost.

The second line contains an integer value  $n$ , representing the total number of friends.

#### **Output Format**

The first line of output displays "Cost after a 10% discount: " followed by the discounted cost of the ticket package as a float value formatted to two decimal places.

The second line displays "Friend with a 35% bonus pays: " followed by the amount paid by the friend with the bonus as a float value formatted to two decimal places.

The third line displays "Each of the other friends pays: " followed by the individual share of the remaining cost as a float value formatted to two decimal places.

Refer to the sample output for the formatting specifications.

#### **Sample Test Case**

Input: 10000.0

5

Output: Cost after a 10% discount: 9000.00

Friend with a 35% bonus pays: 3150.00

Each of the other friends pays: 1462.50

#### **Answer**

```
# You are using Python
# Read input
f = float(input()) # Total cost
n = int(input()) # Total number of friends

# Apply 10% discount
discounted_cost = f * 0.90

# Friend with 35% bonus pays 35% of the discounted cost
bonus_friend_payment = discounted_cost * 0.35

# Remaining cost is divided among the other (n-1) friends
remaining_cost = discounted_cost - bonus_friend_payment
```

```
other_friends_payment = remaining_cost / (n - 1)

# Output with required formatting
print(f"Cost after a 10% discount: {discounted_cost:.2f}")
print(f"Friend with a 35% bonus pays: {bonus_friend_payment:.2f}")
print(f"Each of the other friends pays: {other_friends_payment:.2f}")
```

Status : Correct

Marks : 10/10

## 2. Problem Statement

John is developing a financial application to help users manage their investment portfolios. As part of the application, he needs to write a program that receives the portfolio's main value and the values of two specific investments as inputs. The program should then display these values in reverse order for clear visualization.

Help John achieve this functionality by writing the required program.

### ***Input Format***

The first line of input consists of a float, representing the first investment value.

The second line of input consists of a float, representing the second investment value.

The third line of input consists of an integer, representing the portfolio ID.

### ***Output Format***

The first line of output prints "The values in the reverse order:".

The second line prints the integer, representing the portfolio ID.

The third line prints the second float, representing the second investment value.

The fourth line prints the first float, representing the first investment value.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 35.29

9374.11

48

Output: The values in the reverse order:

48

9374.11

35.29

### **Answer**

```
# You are using Python
# Read input
first_investment = float(input())
second_investment = float(input())
portfolio_id = int(input())

# Output in reverse order
print("The values in the reverse order:")
print(portfolio_id)
print(second_investment)
print(first_investment)
```

**Status : Correct**

**Marks : 10/10**

### **3. Problem Statement**

Nina is working on a project involving multiple sensors. Each sensor provides a data point that needs to be processed to compute an aggregated value.

Given data points from three sensors, write a program to calculate the aggregated value using specific bitwise operations and arithmetic manipulations. The final result should be the aggregated value modulo 1000.

**Example:**

Input:

1 //sensor 1 data

2 //sensor 2 data

3 //sensor 3 data

Output

9

### Explanation

Calculate the bitwise AND of sensor 1 data and sensor 2 data: 0

Calculate the XOR of the result from step 1 and sensor 3 data: 3

Multiply the result from step 2 by 3: 9

Compute the final aggregated value by taking the result from step 3 modulo 1000: 9

So, the aggregated value is 9.

### ***Input Format***

The first line of input consists of an integer S1, representing sensor1 data.

The second line of input consists of an integer S2, representing sensor2 data.

The third line of input consists of an integer S3, representing sensor3 data.

### ***Output Format***

The output displays an integer representing the aggregated value.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 1

2

3

Output: 9

Answer

```
# You are using Python  
  
s1 = int(input())  
s2 = int(input())  
s3 = int(input())  
  
bitwise_and = s1 & s2  
xor_result = bitwise_and ^ s3  
multiplied = xor_result * 3  
aggregated_value = multiplied % 1000  
  
print(aggregated_value)
```

Status : Correct

Marks : 10/10

#### 4. Problem Statement

Emily is organizing a taco party and needs to determine the total number of tacos required and the total cost. Each attendee at the party will consume 2 tacos. To ensure there are enough tacos:

If there are 10 or more attendees, Emily will need to provide an additional 5 tacos. If there are fewer than 10 attendees, Emily must ensure a minimum of 20 tacos are provided.

The cost of each taco is \$25. Write a program that calculates both the total number of tacos required and the total cost based on the number of attendees.

##### ***Input Format***

The input consists of an integer n, representing the number of attendees.

##### ***Output Format***

The first line prints "Number of tacos needed: " followed by an integer representing the number of tacos needed for n attendees.

The second line prints "Total cost: " followed by an integer representing the total cost.

Refer to the sample output for the formatting specifications.

**Sample Test Case**

Input: 10

Output: Number of tacos needed: 25

Total cost: 625

**Answer**

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

# Each attendee consumes 2 tacos
tacos_needed = n * 2

# Apply extra taco rules
if n >= 10:
    tacos_needed += 5
elif tacos_needed < 20:
    tacos_needed = 20

# Cost per taco
total_cost = tacos_needed * 25

# Output results
print(f"Number of tacos needed: {tacos_needed} Total cost: {total_cost}")
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 1\_PAH**

Attempt : 2

Total Mark : 6

Marks Obtained : 6

#### **Section 1 : Coding**

##### **1. Problem Statement**

Mandy is debating with her friend Rachel about an interesting mathematical claim. Rachel asserts that for any positive integer  $n$ , the ratio of the sum of  $n$  and its triple to the integer itself is always 4. Mandy, intrigued by this statement, decides to validate it using logical operators and basic arithmetic.

She wants to confirm if the statement holds true for any positive integer  $n$ .

##### ***Input Format***

The input consists of a positive integer  $n$ , representing the integer to be tested.

##### ***Output Format***

The first line of output displays "Sum:" followed by an integer representing the

calculated sum.

The second line displays "Rachel's statement is: " followed by a Boolean value indicating whether Rachel's statement is correct.

Refer to the sample output for the formatting specifications.

#### **Sample Test Case**

Input: 12

Output: Sum: 48

Rachel's statement is: True

#### **Answer**

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

# Calculate the sum of n and its triple
total = n + 3 * n # which is 4 * n

# Check Rachel's statement: (n + 3n) / n == 4
is_correct = (total / n) == 4
```

```
# Output
print(f"Sum: {total} Rachel's statement is: {is_correct}")
```

**Status :** Correct

**Marks :** 1/1

## 2. Problem Statement

Ella, an avid TV show enthusiast, is planning a binge-watching marathon for a new series. She has a specific routine: after watching a set number of episodes, she takes a short break.

She is provided with the following information:

Each episode of the series has a fixed duration of 45 minutes. After a certain number of episodes, there is a break of 15 minutes.

Ella wants to know the total time she will need to watch the entire series, including the breaks. Your task is to help Ella by calculating the total viewing time.

#### ***Input Format***

The first line of input consists of an integer E, representing the total number of episodes in the series.

The second line consists of an integer B, representing the number of episodes watched before taking a break.

#### ***Output Format***

The output prints an integer representing the total viewing time required to watch the entire series, including the breaks.

Refer to the sample output for formatting specifications.

#### ***Sample Test Case***

Input: 5

2

Output: 255 minutes

#### ***Answer***

```
# You are using Python
# Read inputs
E = int(input()) # Total number of episodes
B = int(input()) # Episodes before a break
```

#### ***# Constants***

```
EPISODE_DURATION = 45
BREAK_DURATION = 15
```

```
# Total break count (no break after last episode)
breaks = (E - 1) // B
```

```
# Total viewing time
total_time = E * EPISODE_DURATION + breaks * BREAK_DURATION
```

```
# Output the result  
print(f"{total_time} minutes")
```

Status : Correct

Marks : 1/1

### 3. Problem Statement

Liam works at a car dealership and is responsible for recording the details of cars that arrive at the showroom. To make his job easier, he wants a program that can take the car's make, model, and price, and display the information in a formatted summary.

Assist him in the program.

#### ***Input Format***

The first line of input contains a string, representing the car make.

The second line contains a string, representing the car model.

The third line contains a float value, representing the car price.

#### ***Output Format***

The first line of output prints "Car Make: ", followed by the car make.

The second line prints "Car Model: ", followed by the car model.

The third line prints "Price: ", followed by the car price, formatted to two decimal places.

Refer to the sample output for formatting specifications.

#### ***Sample Test Case***

Input: Toyota

Camry

23450.75

Output: Car Make: Toyota

Car Model: Camry  
Price: Rs.23450.75

**Answer**

```
# You are using Python
# Input
make = input()
model = input()
price = float(input())

# Output
print(f"Car Make: {make} Car Model: {model} Price: Rs.{price:.2f}")
```

**Status : Correct**

**Marks : 1/1**

**4. Problem Statement**

A smart home system tracks the temperature and humidity of each room. Create a program that takes the room name (string), temperature (float), and humidity (float).

Display the room's climate details.

***Input Format***

The first line of input consists of a string, representing the room name.

The second line consists of a float value, representing the temperature.

The third line consists of a float value, representing the humidity.

***Output Format***

The first line of output prints "Room: " followed by the room name (string).

The second line prints "Temperature: " followed by the temperature (float) formatted to two decimal places.

The third line prints "Humidity: " followed by the humidity (float) formatted to two decimal places and a percentage sign (%).

Refer to the sample output for formatting specifications.

#### **Sample Test Case**

Input: Living Room

23.45

45.78

Output: Room: Living Room

Temperature: 23.45

Humidity: 45.78%

#### **Answer**

```
# You are using Python
# Input
room = input()
temperature = float(input())
humidity = float(input())
```

```
# Output
print(f"Room: {room} Temperature: {temperature:.2f} Humidity: {humidity:.2f}%")
```

**Status : Correct**

**Marks : 1/1**

## **5. Problem Statement**

Oliver is planning a movie night with his friends and wants to download a high-definition movie. He knows the file size of the movie in megabytes (MB) and his internet speed in megabits per second (Mbps). To ensure the movie is ready in time, Oliver needs to calculate the download time.

Your task is to write a program that calculates the download time and displays it in hours, minutes, and seconds.

#### **Example**

Input:

MB = 800

mbps = 40

**Output:**

Download Time: 0 hours, 2 minutes, and 40 seconds

**Explanation:**

Convert the file size to bits ( $800 \text{ MB} * 8 \text{ bits/byte} = 6400 \text{ megabits}$ ) and divide it by the download speed ( $6400 \text{ Mbps} / 40 \text{ Mbps} = 160 \text{ seconds}$ ). Now, convert the download time in seconds to hours, minutes, and seconds: 160 seconds is equal to 2 minutes and 40 seconds. So, the download time is 0 hours, 2 minutes and 40 seconds.

#### ***Input Format***

The first line of input consists of an integer N, representing the file size in megabytes (MB).

The second line consists of an integer S, representing the network speed in megabits per second(mbps).

#### ***Output Format***

The output prints "Download Time: X hours, Y minutes, and Z seconds", where X, Y, and Z are integers representing the hours, minutes, and seconds respectively.

Refer to the sample output for formatting specifications.

#### ***Sample Test Case***

Input: 180

3

Output: Download Time: 0 hours, 8 minutes, and 0 seconds

#### ***Answer***

```
# You are using Python
# Input
N = int(input()) # File size in MB
S = int(input()) # Speed in Mbps

# Convert MB to megabits (1 MB = 8 megabits)
total_megabits = N * 8
```

```
# Calculate total seconds required to download  
total_seconds = total_megabits // S  
  
# Convert seconds to hours, minutes, and seconds  
hours = total_seconds // 3600  
minutes = (total_seconds % 3600) // 60  
seconds = (total_seconds % 3600) % 60  
  
# Output  
print(f"Download Time: {hours} hours, {minutes} minutes, and {seconds}  
seconds")
```

**Status :** Correct

**Marks :** 1/1

## 6. Problem Statement

Shawn, a passionate baker, is planning to bake cookies for a large party. His original recipe makes 15 cookies, with the following ingredient quantities: 2.5 cups of flour, 1 cup of sugar, and 0.5 cups of butter.

Write a program to calculate the amounts of flour, sugar, and butter needed for a different number of cookies. Provide the ingredient quantities for a specified number of cookies, maintaining the original proportions of the recipe.

### ***Input Format***

The input consists of an integer n, representing the number of cookies.

### ***Output Format***

The first line prints "Flour: X cups" where X represents the amount of flour required for n cookies, as a double value rounded to two decimal places.

The second line prints "Sugar: Y cups" where Y represents the amount of Sugar required for n, as a double value rounded to two decimal places.

The third line prints "Butter: Z cups" where Z represents the amount of flour required for n, as a double value rounded to two decimal places.

Refer to the sample output for formatting specifications.

**Sample Test Case**

Input: 15

Output: Flour: 2.50 cups

Sugar: 1.00 cups

Butter: 0.50 cups

**Answer**

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

# Original recipe makes 15 cookies
original_cookies = 15
flour_per_cookie = 2.5 / original_cookies
sugar_per_cookie = 1.0 / original_cookies
butter_per_cookie = 0.5 / original_cookies

# Calculate required amounts for n cookies
flour = flour_per_cookie * n
sugar = sugar_per_cookie * n
butter = butter_per_cookie * n

# Output formatted to 2 decimal places
print(f"Flour: {flour:.2f} cups Sugar: {sugar:.2f} cups Butter: {butter:.2f} cups")
```

Status : Correct

Marks : 1/1

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 2\_MCQ**

Attempt : 1  
Total Mark : 15  
Marks Obtained : 11

#### **Section 1 : MCQ**

1. What is the output of the following code?

```
i = 5
while True:
    if i%009 == 0:
        break
    print(i)
    i += 1
```

**Answer**

5 6 7 8

**Status : Wrong**

**Marks : 0/1**

2. What is the output of the following program?



Scanned with OKEN Scanner

```
i=1  
while(i<3):  
    j=0  
    while(j<3):  
        print(i%3,end=" ")  
        j=j+1  
    i=i+1
```

**Answer**

0 0 0 1 1 1

**Status : Wrong**

**Marks : 0/1**

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

```
i = 1  
while True:  
    if i % 2 == 0:  
        i += 1  
        continue  
    if i > 10:  
        break  
    print(i, end = " ")  
    i += 2
```

**Answer**

1 3 5 7 9

**Status : Correct**

**Marks : 1/1**

4. What will be the output of the following code snippet?

```
i = 0  
while i < 5:  
    if i % 2 == 0:  
        i += 1  
        continue  
    print(i, end=" ")
```

i += 1

**Answer**

1 3

**Status : Correct**

**Marks : 1/1**

5. What is the output of the following?

```
i = 2
while True:
    if i%3 == 0:
        break
    print(i)
    i += 2
```

**Answer**

2 4

**Status : Correct**

**Marks : 1/1**

6. What is the output of the following?

```
True = False
while True:
    print(True)
    break
```

**Answer**

False

**Status : Wrong**

**Marks : 0/1**

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

```
i = 5
while True:
    if i%0011 == 0:
        break
```

```
print(i, end = " ")  
i += 1
```

**Answer**

5 6 7 8 9 10

**Status : Wrong**

**Marks : 0/1**

8. When does the else statement written after the loop execute?

**Answer**

When loop condition becomes false

**Status : Correct**

**Marks : 1/1**

9. What is the output of the following?

```
for i in range(10):  
    if i == 5:  
        break  
    else:  
        print(i, end=' ')  
else:  
    print("Here")
```

**Answer**

0 1 2 3 4

**Status : Correct**

**Marks : 1/1**

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

```
i = 1  
while True:  
    if i%3 == 0:  
        break  
    print(i)  
    i += 1
```

**Answer**

12

**Status : Correct**

**Marks : 1/1**

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

```
i = 0
while i < 5:
    print(i)
    i += 1
    if i == 3:
        break
    else:
        print(0)
```

**Answer**

012

**Status : Correct**

**Marks : 1/1**

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

```
i = 0
for i in range(10):
    break
print(i)
```

**Answer**

0

**Status : Correct**

**Marks : 1/1**

13. Which keyword used in loops can skip the remaining statements for a particular iteration and start the next iteration?

**Answer**

continue

Status : Correct

Marks : 1/1

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

```
i = 1
while True:
    if i % 2 == 0:
        i += 1
        continue
    if i > 10:
        break
    print(i)
    i += 2
```

Answer

1 3 5 7 9

Status : Correct

Marks : 1/1

15. What will the following code output?

```
x = 0
while x < 5:
    if x == 3:
        break
    x += 1
else:
    print("Completed")
print(x)
```

Answer

3

Status : Correct

Marks : 1/1

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 2\_COD\_Updated**

Attempt : 1  
Total Mark : 50  
Marks Obtained : 50

#### **Section 1 : Coding**

##### **1. 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**

```
n = int(input())
```

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

Status : Correct

Marks : 10/10

## 2. 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
start=int(input())
end=int(input())
total = 0
for num in range (start,end + 1):
    if str(num)== str(num)[::-1]:
        continue
    total += sum(int(d) for d in str(num))
```

```
print(total)
```

Status : Correct

Marks : 10/10

### 3. 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***

```
def is_fib(n: int):
    a,b = 0,1
    while a <= n:
        if a== n:
            return True
        a,b = b,a+b
```

```
a, b = b, a + b  
return False
```

```
def count_non_fib(s: int, e: int):  
    c = 0  
    for i in range(s, e+1):  
        if is_fib(i):  
            continue  
        c += 1  
    return c
```

```
s = int(input())  
e = int(input())  
print(count_non_fib(s,e))
```

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**

```
s = input()  
vowels=['A','E','I','O','U']  
con = []
```

```
for l in s:  
    if l.isalpha() and l.upper() not in vowels:  
        con.append(l)
```

```
for l in con:  
    print(l,end="")  
print()
```

**Status : Correct**

**Marks : 10/10**

## **5. 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**

```
start = int(input())
end = int(input())
```

```
for num in range(start,end+1):
    if sum(i for i in range(1,num) if num % i == 0)==num:
        print(num,end=" ")
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 2\_CY

Attempt : 2  
Total Mark : 40  
Marks Obtained : 30

#### **Section 1 : Coding**

##### **1. Problem Statement**

Gabriel is working on a wildlife research project where he needs to compute various metrics for different animals based on their characteristics. Each animal type requires a different calculation: a deer's distance traveled, a bear's weight based on footprint size, or a bird's altitude based on its flying pattern.

Conditions:

For Deer (Mode 'D' or 'd'): Distance = speed of sound \* time taken, where the speed of sound in air is 343 meters per second.  
For Bear (Mode 'B' or 'b'): Weight = footprint size \* average weight, where the average weight per square inch for a bear is 5.0 pounds.  
For Bird (Mode 'F' or 'f'): Altitude = flying pattern \* distance covered (in meters).

Write a program to help Gabriel analyze the characteristics of animals

based on the given inputs.

### ***Input Format***

The first line of input consists of a character, representing the type of animal 'D'/'d' for deer, 'B/b' for bear, and 'F/f' for bird.

If the choice is 'D' or 'd':

The second line of input consists of a floating-point value T, representing the time taken from the deer's location to the observer.

If the choice is 'B' or 'b':

The second line of input consists of a floating-point value S, representing the size of the bear's footprint in square inches.

If the choice is 'F' or 'f':

1. The second line of input consists of a floating-point value P, representing the bird's flying pattern.
2. The third line consists of a floating-point value D, representing the distance covered by the bird in meters.

### ***Output Format***

The output prints one of the following:

If the choice is 'D' or 'd':

The output prints "Distance: X m" where X is a floating point value rounded off to two decimal places, representing the calculated distance traveled by the sound wave in meters.

If the choice is 'B' or 'b':

The output prints "Weight: Y lb" where Y is a floating point value rounded off to two decimal places, representing the estimated weight of the bear in pounds.

If the choice is 'F' or 'f':

The output prints "Altitude: Z m" where Z is a floating point value rounded off to two decimal places, representing the calculated altitude of the bird's flight in meters.

If the given choice is invalid, print "Invalid".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: d

2.5

Output: Distance: 857.50 m

### **Answer**

```
choice = input()
```

```
if choice =='D' or choice =='d':  
    t=float(input())  
    print(f"Distance:{(t*343) : .2f}m")  
  
elif choice =='B' or choice =='b':  
    s=float(input())  
    print(f"Weight:{(s*5.0) : .2f}lb")  
  
elif choice =='F' or choice =='f':  
    p=float(input())  
    d=float(input())  
    print(f"Altitude:{(p*d) : .2f}m")  
  
else:  
    print("Invalid")
```

**Status :** Correct

**Marks :** 10/10

## **2. Problem Statement**

Alex is practicing programming and is curious about prime and non-prime digits. He wants to write a program that calculates the sum of the non-prime digits in a given integer using loops.

Help Alex to complete his task.

**Example:**

**Input:**

845

**output:**

12

**Explanation:**

Digits: 8 (non-prime), 4 (non-prime), 5 (prime)

The sum of Non-Prime Digits:  $8 + 4 = 12$

Output: 12

#### ***Input Format***

The input consists of a single integer X.

#### ***Output Format***

The output prints an integer representing the sum of non-prime digits in X.

Refer to the sample output for formatting specifications.

#### ***Sample Test Case***

**Input:** 845

**Output:** 12

#### ***Answer***

```
def is_prime(digit):
    if digit < 2:
        return False
    for i in range(2,int(digit**0.5)+1):
        if digit % i==0:
            return False
```

```
return True
def sum_of_non_prime_digits(n):
    sum_non_prime = 0
    num_str = str(n)
    for digit_char in num_str:
        digit = int(digit_char)
        if not is_prime(digit):
            sum_non_prime += digit
    return sum_non_prime
if __name__ == "__main__":
    x = int(input())
    result = sum_of_non_prime_digits(x)
    print(result)
```

Status : Correct

Marks : 10/10

### 3. Problem Statement

Max is fascinated by prime numbers and the Fibonacci sequence. He wants to combine these two interests by creating a program that outputs the first n prime numbers within the Fibonacci sequence.

Your task is to help Max by writing a program that prints the first n prime numbers in the Fibonacci sequence using a while loop along with the break statement to achieve the desired functionality.

#### ***Input Format***

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

#### ***Output Format***

The output displays space-separated first n prime numbers found in the Fibonacci sequence.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 5  
Output: 2 3 5 13 89

### **Answer**

```
def is_prime(num):  
    if num<= 1:  
        return 0  
    for i in range(2,int(num**0.5)+1):  
        if num % i == 0:  
            return 0  
    return 1  
  
n = int(input())  
n1,n2 =1,1  
count=0  
  
while count<n:  
    n3=n1+n2  
    if is_prime(n3):  
        count +=1  
        print(n3 ,end="")  
    n1 , n2 =n2,n3
```

Status : Correct

Marks : 10/10

#### **4. Problem Statement**

John is tasked with configuring the lighting for a high-profile event, where different lighting modes affect the ambiance of the venue. He can choose from three distinct lighting modes, each requiring a specific adjustment to the initial light intensity:

Ambient Lighting (Mode 1): The intensity level is multiplied by 1.5.  
Stage Lighting (Mode 2): The intensity level is multiplied by 2.0.  
Spotlight (Mode 3): The intensity level is multiplied by 1.8.

In the event that an invalid mode is provided, the program should output an

error message indicating the invalid selection.

Your task is to write a program that reads the selected lighting mode and the initial intensity level, applies the appropriate adjustment, and prints the final intensity.

#### ***Input Format***

The first line of input is an integer n, representing the lighting mode.

The second line is a floating value m, representing the initial intensity level of the light.

#### ***Output Format***

The output displays "Intensity: " followed by a float representing the adjusted intensity level, formatted to two decimal places, if the mode is valid.

If the mode is invalid, the output should display "Invalid".

Refer to the sample output for formatting specifications.

#### ***Sample Test Case***

**Input:** 1

10.0

**Output:** Intensity: 15.00

#### ***Answer***

-

**Status :** Skipped

**Marks :** 0/10

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 2\_PAH\_Updated**

Attempt : 1  
Total Mark : 60  
Marks Obtained : 50

#### **Section 1 : Coding**

##### **1. Problem Statement**

Imagine being entrusted with the responsibility of creating a program that simulates a math workshop for students. Your task is to develop an interactive program that not only calculates but also showcases the charm of factorial values. Your program should efficiently compute and present the sum of digits for factorial values of only odd numbers within a designated range. This approach will ingeniously keep even factorials at bay, allowing students to delve into the intriguing world of mathematics with enthusiasm and clarity.

##### ***Input Format***

The input consists of a single integer, n.

##### ***Output Format***



Scanned with OKEN Scanner

The output displays the factorial and sum of digits of the factorial of odd numbers within the given range.

Refer to the sample output for the formatting specifications.

#### **Sample Test Case**

**Input:** 6

**Output:** 1! = 1, sum of digits = 1  
3! = 6, sum of digits = 6  
5! = 120, sum of digits = 3

#### **Answer**

```
import math
n= int(input())
for i in range(1,n+1,2):
    s=math .factorial(i)
    print(f"{i}! = {s}, sum of digits = {sum(int(d)for d in str(s))}")
```

**Status :** Correct

**Marks :** 10/10

## **2. Problem Statement**

Kamali recently received her electricity bill and wants to calculate the amount she needs to pay based on her usage. The electricity company charges different rates based on the number of units consumed.

For the first 100 units, there is no charge. For units consumed beyond 100 and up to 200, there is a charge of Rs. 5 per unit. For units consumed beyond 200, there is a charge of Rs. 10 per unit.

Write a program to help Kamali calculate the amount she needs to pay for her electricity bill based on the units consumed.

#### **Input Format**

The input consists of an integer, representing the number of units.

### **Output Format**

The output prints the total amount of the electricity bill, an integer indicating the amount Kamali needs to pay in the format "Rs. amount".

Refer to the sample output for the exact format.

### **Sample Test Case**

Input: 350

Output: Rs. 2000

### **Answer**

```
units = int(input())
if(units<100):
    print("Rs.0")
elif(100<units<290):
    print(f"Rs.{(units-100)*5}")
else:
    print(f"Rs.{(100*5)+((units-200)*10)}")
```

**Status : Correct**

**Marks : 10/10**

### **3. Problem Statement**

Sophia, a primary school teacher, wants to calculate the sum of numbers within a given range, excluding those that are multiples of 3.

Write a program to help Sophia compute the sum of all numbers between start and end (inclusive) that are not divisible by 3 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 of input consists of an integer, representing the ending number of the range.

#### **Output Format**

The output prints a single integer, representing the sum of numbers in the range that are not multiples of 3.

Refer to the sample output for formatting specifications.

#### **Sample Test Case**

Input: 1  
10  
Output: 37

#### **Answer**

```
# You are using Python
start=int(input())
end=int(input())
```

```
total=0
for num in range(start,end+1):
    if num % 3 == 0:
        continue
    total += num
print(total)
```

**Status : Correct**

**Marks : 10/10**

#### **4. Problem Statement**

As a software engineer, your goal is to develop a program that facilitates the identification of leap years in a specified range. Your task is to create a program that takes two integer inputs, representing the start and end years of the range and then prints all the leap years within that range.

### **Input Format**

The first line of the input consists of an integer, which represents the start year.

The second line consists of an integer, which represents the end year.

### **Output Format**

The output displays the leap years within the given range, separated by lines.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 2000

2053

Output: 2000

2004

2008

2012

2016

2020

2024

2028

2032

2036

2040

2044

2048

2052

### **Answer**

```
# You are using Python
start=int(input())
end=int(input())
for year in range(start,end+1):
    if(year % 4 == 0 and year %100 != 0)or(year % 400 == 0):
        print(year)
```

Status : Correct

Marks : 10/10

## 5. Problem Statement

Aarav is fascinated by the concept of summing numbers separately based on their properties. He plans to write a program that calculates the sum of even numbers and odd numbers separately from 1 to a given positive integer.

Aarav wants to input an integer value to represent the upper limit of the range. Help Aarav by developing a program that computes and displays the sum of even and odd numbers separately.

### ***Input Format***

The input consists of a single integer N, where N is the upper limit of the range.

### ***Output Format***

The output consists of two lines:

- The first line displays the sum of even numbers from 1 to N.
- The second line displays the sum of odd numbers from 1 to N.

Refer to the sample output for the exact format.

### ***Sample Test Case***

Input: 10

Output: Sum of even numbers from 1 to 10 is 30  
Sum of odd numbers from 1 to 10 is 25

### ***Answer***

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

```
total_even = 0
total_odd = 0
for num in range(1,n + 1):
    if num % 2 == 0:
        total_even += num
```

```
else:  
    total_old += num  
  
print(f"Sum of even numbers from 1 to {n} is",total_even)  
print(f"Sum of odd numbers from 1 to {n} is",total_old)
```

**Status :** Correct

**Marks :** 10/10

## 6. Problem Statement

Rajesh wants to design a program that simulates a real-time scenario based on a mathematical concept known as the Collatz Conjecture. This concept involves the repeated application of rules to a given starting number until the number becomes 1. The rules are as follows:

If the number is even, divide it by 2. If the number is odd, multiply it by 3 and add 1.

Your task is to write a program that takes a positive integer as input, applies the Collatz Conjecture rules to it, counts the number of steps taken to reach 1, and provides an output accordingly. If the process exceeds 100 steps, the program should print a message indicating so and use break to exit.

### ***Input Format***

The input consists of a single integer, n.

### ***Output Format***

The output displays the total number of steps taken to reach 1 if it's under 100.

If it's more than 100, it displays "Exceeded 100 steps. Exiting....".

Refer to sample output for the formatting specifications.

### ***Sample Test Case***

**Input:** 6

Output: Steps taken to reach 1: 8

**Answer**

# You are using Python

**Status : Wrong**

**Marks : 0/10**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 3\_MCQ**

Attempt : 1  
Total Mark : 25  
Marks Obtained : 20

#### **Section 1 : MCQ**

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

```
my_list = [1, 2, 2, 3]
print(my_list.count(2))
```

*Answer*

2

**Status :** Correct

**Marks :** 1/1

2. What is the output of the following Python code?

```
a = "Hello"
b = "World"
c = a + " " + b
```

print(c)

**Answer**

Hello World

**Status : Correct**

**Marks : 1/1**

3. Suppose list1 is [2, 33, 222, 14, 25], What is list1[-1]?

**Answer**

[25, 14, 222, 33, 2]

**Status : Wrong**

**Marks : 0/1**

4. Suppose list1 is [2, 33, 222, 14, 25], What is list1[-1]?

**Answer**

25

**Status : Correct**

**Marks : 1/1**

5. What is the output of the following Python code?

```
text = "Python"  
result = text.center(10, "*")  
print(result)
```

**Answer**

\*\*Python\*\*

**Status : Correct**

**Marks : 1/1**

6. If you have a list lst = [1, 2, 3, 4, 5, 6], what does the slicing operation lst[-3:] return?

**Answer**

The last three elements of the list

**Status : Correct**

**Marks : 1/1**

7. Which method is used to add multiple items to the end of a list?

**Answer**

`extend()`

**Status : Correct**

**Marks : 1/1**

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

```
numbers = [1, 2, 3, 4, 5]
numbers.remove(6)
print(numbers)
```

**Answer**

`ValueError: list.remove(x): x not in list`

**Status : Correct**

**Marks : 1/1**

9. What is the output of the following Python code?

```
name = "John"
age = 25
message = "My name is %s and I am %d years old." % (name, age)
print(message)
```

**Answer**

My name is John and I am 25 years old.

**Status : Correct**

**Marks : 1/1**

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

```
numbers = [1, 2, 3, 4, 5]
numbers.append(6, 7)
print(numbers)
```

**Answer**

[1, 2, 3, 4, 5, (6, 7)]

**Status : Wrong**

**Marks : 0/1**

11. Which of the following is a valid way to use the '%' operator to concatenate strings in Python?

**Answer**

"%s %s" % (string1, string2)

**Status : Correct**

**Marks : 1/1**

12. What is the output of the following code?

```
my_list = [3, 6, 1, 2, 5, 4]
print(sorted(my_list) == my_list.sort())
```

**Answer**

False

**Status : Correct**

**Marks : 1/1**

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

```
text = " Python "
answer = text.strip()
print(answer)
```

**Answer**

Python

**Status : Correct**

**Marks : 1/1**

14. What does negative indexing in Python lists allow you to do?

**Answer**



Scanned with OKEN Scanner

Access elements in the list from the end

Status : Correct

Marks : 1/1

15. Which method in Python is used to create an empty list?

Answer

list()

Status : Correct

Marks : 1/1

16. What is the output of the following Python code?

```
string1 = "Hello"  
string2 = "World"  
result = string1 + string2  
print(result)
```

Answer

HelloWorld

Status : Correct

Marks : 1/1

17. What is the output of the following Python code?

```
word = "Python"  
result = word[::-1]  
print(result)
```

Answer

nohtyp

Status : Wrong

Marks : 0/1

18. What is the output of the following Python code?

```
txt = "My Classroom"  
print(txt.find("o"))
```

`print(txt.index("o"))`

**Answer**

88

**Status : Wrong**

**Marks : 0/1**

19. What does the append() method do in Python?

**Answer**

Adds a new element to the end of the list

**Status : Correct**

**Marks : 1/1**

20. What is the output of the following Python code?

```
word = "programming"
answer = word.index("gram")
print(answer)
```

**Answer**

3

**Status : Correct**

**Marks : 1/1**

21. What is the output of the following code?

```
my_list = [1, 2, 3]
my_list *= 2
print(len(my_list))
```

**Answer**

6

**Status : Correct**

**Marks : 1/1**

22. What is the output of the following Python code?

```
b = "Projects!"  
print(b[2:5])
```

**Answer**

oje

**Status : Correct**

**Marks : 1/1**

23. What is the result of the slicing operation lst[-5:-2] on the list lst = [1, 2, 3, 4, 5, 6]?

**Answer**

[3, 4, 5]

**Status : Wrong**

**Marks : 0/1**

24. What does the following code output?

```
lst = [10, 20, 30, 40, 50]  
print(lst[-4:-1])
```

**Answer**

[20, 30, 40]

**Status : Correct**

**Marks : 1/1**

25. Suppose list1 is [4, 2, 2, 4, 5, 2, 1, 0], Which of the following is the correct syntax for slicing operation?

**Answer**

all of the mentioned options

**Status : Correct**

**Marks : 1/1**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 3\_COD

Attempt : 1  
Total Mark : 50  
Marks Obtained : 50

#### **Section 1 : Coding**

##### **1. 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***

```
phone_number =input()  
area_code = phone_number[1:4]  
print("Area code:",area_code)
```

**Status : Correct**

**Marks : 10/10**

## **2. 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***

```
input_list=eval(input())
negatives=[x for x in input_list if x<0]
non_negatives=[x for x in input_list if x>=0]
arranged_list = negatives + non_negatives
print("list =",arranged_list)
```

Status : Correct

Marks : 10/10

### **3. 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***

```
str1 =input()  
str2 =input()  
reversed_str2 = str2[::-1]  
result = str1+ reversed_str2  
print(result)
```

**Status : Correct**

**Marks : 10/10**

## **4. 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**

```
input_string=input()
start = int(input())
end = int(input())
if 0<= start <=end < len(input_string):
    print(input_string[start:end +1])
else:
    print("Invalid start and end positions")
```

**Status : Correct**

**Marks : 10/10**

## **5. 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***

```
n = int(input())
list =[int(input()) for _ in range(n)]
M= int(input())
print(f"List after appending elements: {list}")
popped_element = list.pop(M)
```

```
print(f"List after popping last element: {list}")  
print(f"Popped element: {popped_element}")
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 3\_PAH**

Attempt : 1  
Total Mark : 60  
Marks Obtained : 60

#### **Section 1 : Coding**

##### **1. Problem Statement**

Imagine you are developing a text analysis tool for a cybersecurity company. Your task is to analyze 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.

##### ***Input Format***

The input consists of the log entry provided as a single string.

##### ***Output Format***

The output consists of four lines:

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

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

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

The fourth line contains an integer representing the count of special characters in the format "Special characters: {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**

```
log_entry = input()
uppercase_count = 0
lowercase_count = 0
digits_count = 0
special_characters_count = 0

for char in log_entry:
    if 'A' <= char <='Z':
        uppercase_count += 1
    elif 'a' <= char <= 'z':
        lowercase_count += 1
    elif '0' <= char <= '9':
        digits_count +=1
    else:
        special_characters_count += 1

print(f"Uppercase letters:{uppercase_count}")
print(f"Lowercase letters:{lowercase_count}")
print(f"Digits:{digits_count}")
```

```
print(f"Special characters:{special_characters_count }")
```

Status : Correct

Marks : 10/10

## 2. Problem Statement

Kyara is analyzing a series of measurements taken over time. She needs to identify all the "peaks" in this list of integers.

A peak is defined as an element that is greater than its immediate neighbors. Boundary elements are considered peaks if they are greater than their single neighbor.

Your task is to find and list all such peaks using list comprehension.

Example

Input

1 3 2 4 1 5 7 6 10 2 8

Output

Peaks: [3, 4, 7, 10, 8]

Explanation

3 is a peak because it's greater than 1 and 2.

4 is a peak because it's greater than 2 and 1.

7 is a peak because it's greater than 5 and 6.

10 is a peak because it's greater than 6 and 2.

8 is a peak because it is a boundary element and it is greater than 2.

***Input Format***

The input consists of several integers separated by spaces, representing the measurements.

***Output Format***

The output displays "Peaks: " followed by a list of integers, representing the peak elements in the list.

Refer to the sample output for the formatting specifications.

#### **Sample Test Case**

Input: 1 3 2 4 1 5 7 6 10 2 8

Output: Peaks: [3, 4, 7, 10, 8]

#### **Answer**

```
nums =list(map(int,input().strip().split()))
Peaks = [nums[i] for i in range(len(nums))if
        (i > 0 and i < len (nums) -1 and nums[i]> nums[i-1] and nums[i] >nums[i+1])or
        (i == 0 and len(nums) > 1 and nums[i] > nums[i+1])or
        (i == len(nums)- 1 and len(nums)>1 and nums[i]> nums[i -1])]

print(f"Peaks:{Peaks}")
```

**Status : Correct**

**Marks : 10/10**

### **3. Problem Statement**

Accept an unsorted list of length n with both positive and negative integers, including 0. The task is to find the smallest positive number missing from the array. Assume the n value is always greater than zero.

#### **Input Format**

The first line consists of n, which means the number of elements in the array.

The second line consists of the values in the list as space-separated integers.

#### **Output Format**

The output displays the smallest positive number, which is missing from the array.

Refer to the sample output for the formatting specifications.

**Sample Test Case**

Input: 6  
-5 2 0 -1 -10 2

Output: 1

**Answer**

```
n = int(input().strip())
arr = list(map(int,input().strip().split()))
positive_set = set( X for X in arr if X > 0)
missing =1
while missing in positive_set:
    missing +=1
print(missing)
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Neha is learning string operations in Python and wants to practice using built-in functions. She is given a string A, and her task is to:

Find the length of the string using a built-in function. Copy the content of A into another string B using built-in functionality.

Help Neha implement a program that efficiently performs these operations.

***Input Format***

The input consists of a single line containing the string A (without spaces).

***Output Format***

The first line of output prints the length of the given string.

The second line prints the copied string without an extra newline at the end.

Refer to the sample output for the formatting specifications.

**Sample Test Case**

Input: technology-23

Output: Length of the string: 13

Copied string: technology-23

**Answer**

```
A = input().strip()  
length = len(A)  
B = A  
print(f"Length of the string: {length}")  
print(f"Copied string: {B}")
```

**Status : Correct**

**Marks : 10/10**

## 5. Problem Statement

Gowri was doing her homework. She needed to write a paragraph about modern history. During that time, she noticed that some words were repeated repeatedly. She started counting the number of times a particular word was repeated.

Your task is to help Gowri to write a program to get a string from the user. Count the number of times a word is repeated in the string.

**Note:** Case-sensitive

***Input Format***

The first line of input consists of a string, str1.

The second line consists of a single word that needs to be counted, str2.

***Output Format***

The output displays the number of times the given word is in the string.

If the second string str2 is not present in the first string str1, it prints 0.

Refer to the sample output for the formatting specifications.

#### **Sample Test Case**

Input: I felt happy because I saw the others were happy and because I knew I  
should feel happy  
happy

Output: 3

#### **Answer**

```
str1 =input().strip()  
str2 =input().strip()  
words =str1.strip()  
count =words.count(str2)  
print(count)
```

**Status :** Correct

**Marks :** 10/10

## **6. Problem Statement**

You are tasked with writing a program that takes n integers as input from the user and stores them in a list. After this, you need to transform the list according to the following rules:

The element at index 0 should be replaced with 0. For elements at even indices (excluding index 0), replace the element with its cube. For elements at odd indices, replace the element with its square.

Additionally, you should sort the list in ascending order before applying these transformations.

#### ***Input Format***

The first line of input represents the size of the list, N.

The elements of the list are represented by the next N lines.

#### ***Output Format***

The first line of output displays "Original List: " followed by the original list.

The second line displays "Replaced List: " followed by the replacement list as per the given condition.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 5

5  
1  
2  
3  
4

Output: Original List: [1, 2, 3, 4, 5]  
Replaced List: [0, 4, 27, 16, 125]

### **Answer**

```
n = int(input().strip())
lst =[int(input().strip())for _ in range(n)]
lst.sort()
print(f"Original List:{lst}")
replaced_list = [0 if i == 0 else (x**3 if i % 2 == 0 else x**2)for i,x in
enumerate(lst)]
print(f"Replaced List: {replaced_list}")
```

Status : Correct

Marks : 10/10

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 3\_CY

Attempt : 1  
Total Mark : 30  
Marks Obtained : 25

#### **Section 1 : Coding**

##### **1. Problem Statement**

Raja needs a program that helps him manage his shopping list efficiently. The program should allow him to perform the following operations:

**Add Items:** Raja should be able to add multiple items to his shopping list at once. He will input a space-separated list of items, each item being a string.

**Remove Item:** Raja should be able to remove a specific item from his shopping list. He will input the item he wants to remove, and if it exists in the list, it will be removed. If the item is not found, the program should notify him.

**Update List:** Raja might realize he forgot to add some items initially. After removing unnecessary items, he should be able to update his list by adding more items. Similar to the initial input, he will provide a space-separated

list of new items.

#### ***Input Format***

The first line consists of the initial list of integers should be entered as space-separated values.

The second line consists of the element to be removed should be entered as a single integer value.

The third line consists of the new elements to be appended should be entered as space-separated values.

#### ***Output Format***

The output displays the current state of Raja's shopping list after each operation. After adding items, removing items, and updating the list, the program prints the updated shopping list in the following format:

"List1: [element1, element2, ... ,element\_n]

List after removal: [element1, element2, ... ,element\_n]

Final list: [element1, element2, ... ,element\_n]".

If the item is not found in the removing item process, print the message "Element not found in the list".

Refer to the sample output for the formatting specifications.

#### ***Sample Test Case***

Input: 1 2 3 4 5

3  
6 7 8

Output: List1: [1, 2, 3, 4, 5]

List after removal: [1, 2, 4, 5]

Final list: [1, 2, 4, 5, 6, 7, 8]

### Answer

```
list1 = list(map(int,input().split()))
remove_items = list(map(int,input().split()))
add_items = list(map(int,input().split()))
print("List1:",list1)
for item in remove_items:
    if item in list1:
        list1.remove(item)
        print(f"List after removal:{list1}")
    else:
        print(f"Element not found in the list")
list1.extend(add_items)
print("Final list:",list1)
```

Status : Correct

Marks : 10/10

## 2. Problem Statement

A company is creating email accounts for its new employees. They want to use a naming convention for email addresses that consists of the first letter of the employee's first name, followed by their last name, followed by @company.com.

The company also has a separate email domain for administrative employees.

Write a program that prompts the user for their first name, last name, role, and company and then generates their email address using the appropriate naming convention based on their role. This is demonstrated in the below examples.

Note:

The generated email address should consist of the first letter of the first name, the last name in lowercase, and a suffix based on the role and company, all in lowercase.

### ***Input Format***

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

The second line consists of the last name of an employee as a string.

The third line consists of the role of the employee as a string.

The last line consists of the company name as a string.

### ***Output Format***

The output consists of a single line containing the generated email address for the employee, following the specified naming convention.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: John

Smith

admin

iamNeo

Output: jsmith@admin.iamneo.com

### ***Answer***

```
first_name = input().strip()
last_name = input().strip()
role = input().strip()
company = input().strip()
email_prefix = first_name[0].lower() + last_name.lower()
if role.lower() == "admin":
    domain = "admin." + company.lower() + ".com"
else:
    domain = company.lower() + ".com"

email = email_prefix +"@" + domain
print(email)
```

**Status : Correct**

**Marks : 10/10**

### 3. Problem Statement

Write a program to check if a given string is perfect.

A perfect string must satisfy the following conditions:

The string starts with a consonant. The string alternates between consonants and vowels. Each consonant appears exactly once. Vowels can occur consecutively multiple times but should not be followed immediately by a consonant.

If the string satisfies all these conditions, print "True"; otherwise, print "False".

#### ***Input Format***

The input consists of a string.

#### ***Output Format***

The output prints "True" if the string is perfect. Otherwise, print "False".

Refer to the sample output for formatting specifications.

#### ***Sample Test Case***

Input: capacitor

Output: True

#### ***Answer***

```
def is_perfect_string(s):
    vowels = {'a','e','i','o','u'}
    if s[0] in vowels: return False
    seen_consonants, prev_vowel = set(), False
    for c in s:
        if c in vowels: prev_vowel = True
        else:
            if not prev_vowel or c in seen_consonants: return False
            seen_consonants.add(c)
            prev_vowel = False
    return True
```

```
print(is_perfect_string(input().strip()))
```

**Status :** Partially correct

**Marks :** 5/10



Scanned with OKEN Scanner

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 4\_MCQ**

Attempt : 1  
Total Mark : 15  
Marks Obtained : 13

#### **Section 1 : MCQ**

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

```
def func(a, b=5, c=10):
    print('a is', a, 'and b is', b, 'and c is', c)
```

```
func(3, 7)
func(25, c = 24)
func(c = 50, a = 100)
```

#### **Answer**

a is 3 and b is 7 and c is 10a is 25 and b is 5 and c is 24a is 100 and b is 5 and c is 50

Status : **Correct**

**Marks : 1/1**

2. What is the output of the following code snippet?



Scanned with OKEN Scanner

```
def add(a, b=2):  
    return a - b
```

```
result = add(3)  
print(result)
```

**Answer**

6

**Status : Wrong**

**Marks : 0/1**

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

```
def cube(x):  
    return x * x * x  
x = cube(3)  
print(x)
```

**Answer**

27

**Status : Correct**

**Marks : 1/1**

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

```
def C2F(c):  
    return c * 9/5 + 32  
print(C2F(100))  
print(C2F(0))
```

**Answer**

212.032.0

**Status : Correct**

**Marks : 1/1**

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

```
num = -5
```

```
result = abs(num)
print(result)
```

**Answer**

5

**Status : Correct**

**Marks : 1/1**

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

```
value = 42
result = abs(value) + len(str(value))
print(result)
```

**Answer**

44

**Status : Correct**

**Marks : 1/1**

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

```
def is_even(number):
    if number % 2 == 0:
        return True

result = is_even(6)
print(result)
```

**Answer**

True

**Status : Correct**

**Marks : 1/1**

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

```
def absolute_value(x):
    if x < 0:
        return -x
    return x
```

```
result = absolute_value(-9)
print(result, absolute_value(5))
```

**Answer**

9 5

**Status : Correct**

**Marks : 1/1**

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

```
num1 = 10
num2 = -10
result = abs(num1) + abs(num2)
print(result)
```

**Answer**

20

**Status : Correct**

**Marks : 1/1**

10. Which of the following functions can take a lambda function as a parameter in Python?

**Answer**

All of the mentioned options

**Status : Wrong**

**Marks : 0/1**

11. What is the output of the following code snippet?

```
def square(x):
    return x ** 2
```

```
result = square(4)
print(result)
```

**Answer**

16

Status : Correct

Marks : 1/1

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

```
number = 7  
result = abs(number) + pow(number, 2)  
print(result)
```

Answer

56

Status : Correct

Marks : 1/1

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

```
def display(*args):  
    for arg in args:  
        print(arg)
```

```
display(10, 20, 30)
```

Answer

102030

Status : Correct

Marks : 1/1

14. What is the output of the following code snippet?

```
def my_function(x):  
    x += 5  
    return x
```

```
a = 10  
result = my_function(a)  
print(a, result)
```

Answer

10.15

Status : Correct

Marks : 1/1

15. What is the main advantage of using lambda functions in Python?

**Answer**

They allow you to write shorter code than regular functions

Status : Correct

Marks : 1/1

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 4\_COD\_Updated**

Attempt : 1  
Total Mark : 50  
Marks Obtained : 41.5

#### **Section 1 : Coding**

##### **1. 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):
    # You are using Python
    # Type your code here
    u,l,d,sp=0,0,0,0
    for i in range(len(input_string)):
        if input_string[i].isupper():
            if input_string[i].isupper():
                u += 1
            else:
                sp +=1
        else:
            if input_string[i].isdigit():
                d+=1
            else:
                l+=1
    return u,l,d,sp
```

```
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 :** Partially correct

**Marks :** 1.5/10

## 2. 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 `baseexponent` 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 `base1` to `baseexponent`.

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
import math
a=int(input())
b=int(input())
arr=[]
rs=pow(a,b)
for i in range(1,b+1):
    t=pow(a,i)
    arr.append(t)
s=sum(arr)
print(f"{rs}\n{arr}\n{s}")
```

**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**

```
m=input()  
print(len(m))
```

**Status : Correct**

**Marks : 10/10**

#### 4. 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**

```
n=int(input())
a=0
sums =0
for i in str(n):
    a+=1
```

```
temp = n
while temp> 0:
    digit =temp % 10
    sums += digit ** a
    temp//= 10
if(n == sums):
    print(f"\{n} is an Armstrong number.")
else:
    print(f"\{n} is not an Armstrong number.")
```

**Status : Correct**

**Marks : 10/10**

## **5. 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".

241501146

241501146

241501146

241501146

Refer to the sample for the formatting specifications.

**Sample Test Case**

Input: Examly  
e

Output: False

**Answer**

```
# You are using Python
s=input()
c=input()
t= lambda x: x[0]
if t(s)==t(c):
    print("True")
else:
    print("False")
```

**Status :** Correct

**Marks :** 10/10

241501146

241501146

241501146

241501146

241501146

241501146

241501146

241501146

# Rajalakshmi Engineering College

Name: praneeth sai

Email: 241501146@rajalakshmi.edu.in

Roll no: 241501146

Phone: 6380562561

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 4\_CY**

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### **Section 1 : Coding**

#### **1. Problem Statement**

Create a program for a mathematics competition where participants need to find the smallest positive divisor of a given integer n. Your program should efficiently determine this divisor using the min() function and display the result.

#### ***Input Format***

The input consists of a single positive integer n, representing the number for which the smallest positive divisor needs to be found.

#### ***Output Format***

The output prints the smallest positive divisor of the input integer in the format:  
"The smallest positive divisor of [n] is: [smallest divisor]".

Refer to the sample output for the exact format.

#### **Sample Test Case**

Input: 24

Output: The smallest positive divisor of 24 is: 2

#### **Answer**

```
# You are using Python
def smallest_divisor(n):
    return min([i for i in range(2, n + 1) if n % i == 0])

# Input
n = int(input())

# Output
print(f"The smallest positive divisor of {n} is: {smallest_divisor(n)}")
```

**Status : Correct**

**Marks : 10/10**

## **2. Problem Statement**

Meena is analyzing a list of integers and needs to count how many numbers in the list are even and how many are odd. She decides to use lambda functions to filter the even and odd numbers from the list.

Write a program that takes a list of integers, counts the number of even and odd numbers using lambda functions, and prints the results.

#### **Input Format**

The first line contains an integer  $n$ , representing the number of integers in the list.

The second line contains  $n$  space-separated integers.

#### **Output Format**

The first line of output prints an integer representing the count of even numbers.

The second line of output prints an integer representing the count of odd numbers.

Refer to the sample output for the formatting specifications.

**Sample Test Case**

Input: 7  
12 34 56 78 98 65 23

Output: 5  
2

**Answer**

```
n = int(input())
```

```
numbers = list(map(int, input().split()))
```

```
even_count = len(list(filter(lambda x: x % 2 == 0, numbers)))
```

```
odd_count = len(list(filter(lambda x: x % 2 != 0, numbers)))
```

```
print(even_count)  
print(odd_count)
```

**Status : Correct**

**Marks : 10/10**

### 3. Problem Statement

Amrita is developing a password strength checker for her website. She wants the checker to consider the length and the diversity of characters used in the password. A strong password should be long and include a mix of character types: uppercase, lowercase, digits, and special symbols.

She also wants the feedback to be user-friendly, so she wants to include the actual password in the output. Help Amrita finish this password checker using Python's built-in string methods.

**Character Types Considered:**

Lowercase letters (a-z) Uppercase letters (A-Z) Digits (0-9) Special characters (from `string.punctuation`, e.g. @, !, #, \$)

***Input Format***

The input consists of a single string representing the user's password.

***Output Format***

The program prints the strength of the password in this format:

If the password length < 6 characters or fewer than 2 of the 4 character types, the output prints "<password> is Weak"

If password length  $\geq 6$  and at least 2 different character types, the output prints "<password> is Moderate"

If Password length  $\geq 10$  and all 4 character types present, the output prints "<password> is Strong"

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: password123

Output: password123 is Moderate

***Answer***

```
import string
```

```
def check_password_strength(password):
    has_lower = any(c.islower() for c in password)
    has_upper = any(c.isupper() for c in password)
    has_digit = any(c.isdigit() for c in password)
```

```
has_special = any(c in string.punctuation for c in password)

character_types = sum([has_lower, has_upper, has_digit, has_special])

if len(password) < 6 or character_types < 2:
    strength = "Weak"
elif len(password) >= 10 and character_types == 4:
    strength = "Strong"
else:
    strength = "Moderate"

print(f"{password} is {strength}")

if __name__ == "__main__":
    password_input = input()
    check_password_strength(password_input)
```

Status : Correct

Marks : 10/10

#### 4. Problem Statement

Implement a program for a retail store that needs to find the highest even price in a list of product prices. Your goal is to efficiently determine the maximum even price from a series of product prices. Utilize the max() inbuilt function in the program.

For example, if the prices are 10 15 24 8 37 16, the even prices are 10 24 8 16. So, the maximum even price is 24.

##### ***Input Format***

The input consists of a series of product prices separated by a space.

The prices should be entered as a space-separated string of numbers.

##### ***Output Format***

If there are even prices in the input, the output prints "The maximum even price is: " followed by the maximum even price.

If there are no even prices in the input, the output prints "No even prices were found".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 10 15 24 8 37 16

Output: The maximum even price is: 24

### **Answer**

```
# Input: A space-separated string of product prices
prices = input().split()

# Convert the input strings to integers
prices = [int(price) for price in prices]

# Filter the even prices using a list comprehension
even_prices = [price for price in prices if price % 2 == 0]

# Check if there are any even prices
if even_prices:
    max_even_price = max(even_prices)
    print(f"The maximum even price is: {max_even_price}")
else:
    print("No even prices were found")
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 4\_PAH\_Updated

Attempt : 1  
Total Mark : 60  
Marks Obtained : 60

#### **Section 1 : Coding**

##### **1. Problem Statement**

Create a Python program to monitor temperatures in a greenhouse using two sensors. Calculate and display the absolute temperature difference between the two sensor readings to ensure proper temperature control.

Note: Use the `abs()` built-in function.

##### ***Input Format***

The first line consists of a floating-point number, representing the temperature reading from Sensor 1.

The second line consists of a floating-point number, representing the temperature reading from Sensor 2.

##### ***Output Format***

The output displays the absolute temperature difference between Sensor 1 and Sensor 2, rounded to two decimal places.

Refer to the sample output for the exact format.

#### **Sample Test Case**

Input: 33.2

26.7

Output: Temperature difference: 6.50 °C

#### **Answer**

```
sensor1=float(input())
sensor2=float(input())
td=abs(sensor1-sensor2)
print(f"Temperature difference:{td:.2f} °C")
```

**Status : Correct**

**Marks : 10/10**

## **2. Problem Statement**

Ella is designing a messaging application that needs to handle long text messages efficiently. To optimize storage and transmission, she plans to implement a text compression feature that replaces consecutive repeated characters with the character followed by its count, while leaving non-repeated characters unchanged.

Help Ella create a recursive function to achieve this compression without altering the original message's meaning.

Function Specification: `def compress_string(*args)`

#### ***Input Format***

The input consists of a single line containing the string to be compressed.

#### ***Output Format***

The output consists of a single line containing the compressed string.

Refer to the sample output for the formatting specifications.

**Sample Test Case**

Input: aaaBBBccc

Output: a3B3c3

**Answer**

```
def compress_string(s, index=0, count=1, result=""):
    if index >= len(s): # Base case: If we reach the end of the string
        return result

    if index < len(s) - 1 and s[index] == s[index + 1]: # If the current character is the
        same as the next one
        return compress_string(s, index + 1, count + 1, result)

    result += s[index] + (str(count) if count > 1 else "") # Append the character and
    count if greater than 1

    return compress_string(s, index + 1, 1, result) # Reset count for the next
    character
```

```
s=input().strip()
print(compress_string(s))
```

**Status : Correct**

**Marks : 10/10**

**3. Problem Statement**

Alice works at a digital marketing company, where she analyzes large

datasets. One day, she's tasked with processing customer ID numbers, which are long numeric sequences.

To simplify her task, Alice needs to calculate the digital root of each ID. The digital root is obtained by repeatedly summing the digits of a number until a single digit remains.

Help Alice write a program that reads a customer ID number, calculates its digital root, and prints the result using a loop-based approach.

For example, the sum of the digits of 98675 is  $9 + 8 + 6 + 7 + 5 = 35$ , then  $3 + 5 = 8$ , which is the digital root.

Function prototype: def digital\_root(num)

#### ***Input Format***

The input consists of an integer num.

#### ***Output Format***

The output prints an integer representing the sum of digits for a given number until a single digit is obtained.

Refer to the sample output for the formatting specifications.

#### ***Sample Test Case***

Input: 451110

Output: 3

#### ***Answer***

```
num = int(input())
def digital_root(num):
    while num >= 10:
        num = sum(int(digit) for digit in str(num))
    return num
```

```
input_1 = 451110
```

```
input_2 = 1234
```

```
input_3 = 98675
```

```
print(digital_root(num))
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Hussain wants to create a program to calculate a person's BMI (Body Mass Index) based on their weight in kilograms and height in meters. The BMI is a measure of a person's body fat relative to their height.

Your program should take user input for weight and height, calculate the BMI, and display the result.

**Function Signature:** calculate\_bmi(weight, height)

**Formula:**  $BMI = \frac{Weight}{(Height)^2}$

#### ***Input Format***

The first line of input consists of a positive floating-point number, the person's weight in kilograms.

The second line of input consists of a positive floating-point number, the person's height in meters.

#### ***Output Format***

The output displays "Your BMI is: [BM] followed by a float value representing the calculated BMI, rounded off two decimal points.

Refer to the sample output for the formatting specifications.

**Sample Test Case**

Input: 70.0

1.75

Output: Your BMI is: 22.86

**Answer**

```
weight = float(input())
height = float(input())
```

```
def calculate_bmi(weight, height):
```

```
    bmi = weight / (height ** 2) # Apply the BMI formula
```

```
    print(f"Your BMI is: {bmi:.2f}") # Print the BMI rounded to two decimal places
```

```
calculate_bmi(weight, height)
```

**Status : Correct**

**Marks : 10/10**

## 5. Problem Statement

Ravi is working on analyzing a set of integers to determine how many of them are divisible by 3 and how many are divisible by 5. He decides to use lambda functions to filter and count the numbers based on their divisibility.

Write a program that takes a list of integers, calculates how many numbers are divisible by 3, and how many are divisible by 5, and then prints the results.

Additionally, the program should calculate the total sum of all numbers

divisible by 3 and divisible by 5 separately.

#### ***Input Format***

The first line contains an integer  $n$ , representing the number of integers in the list.

The second line contains  $n$  space-separated integers.

#### ***Output Format***

The first line should print the count of numbers divisible by 3.

The second line should print the count of numbers divisible by 5.

The third line should print the sum of numbers divisible by 3.

The fourth line should print the sum of numbers divisible by 5.

Refer to the sample output for the formatting specifications.

#### ***Sample Test Case***

Input: 6  
3 5 6 10 15 20

Output: 3  
4  
24  
50

#### ***Answer***

```
def count_and_sum_divisibles(n, numbers):
    divisible_by_3 = list(filter(lambda x: x % 3 == 0, numbers))
    divisible_by_5 = list(filter(lambda x: x % 5 == 0, numbers))

    print(len(divisible_by_3)) # Count of numbers divisible by 3
    print(len(divisible_by_5)) # Count of numbers divisible by 5
    print(sum(divisible_by_3)) # Sum of numbers divisible by 3
    print(sum(divisible_by_5)) # Sum of numbers divisible by 5
```

# Example usage:

```
n = int(input()) # Read the number of integers  
numbers = list(map(int, input().split())) # Read the list of integers  
  
count_and_sum_divisibles(n, numbers)
```

**Status :** Correct

**Marks :** 10/10

## 6. Problem Statement

Sophia is developing a feature for her online banking application that calculates the total sum of digits in customers' account numbers. This sum is used to generate unique verification codes for secure transactions. She needs a program that takes an account number as input and outputs the sum of its digits.

Help Sophia to complete her task.

Function Specification: `def sum_digits(num)`

### ***Input Format***

The input consists of an integer, representing the customer's account number.

### ***Output Format***

The output prints an integer representing the sum of the digits of the account number.

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 123245

Output: 17

### ***Answer***

```
num = int(input())  
  
def sum_digits(num):
```

```
return sum(int(digit) for digit in str(num))
```

```
sum = sum_digits(num)  
print(sum)
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 : 16

#### **Section 1 : MCQ**

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

**Answer**

()

**Status :** Correct

**Marks :** 1/1

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

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

**Answer**

False



Scanned with OKEN Scanner

Status : Correct

Marks : 1/1

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

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

Answer

Too many arguments for sum() method

Status : Wrong

Marks : 0/1

4. 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}(1, 2, 7, 8)

Status : Wrong

Marks : 0/1

5. 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

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

**Answer**

Values of a dictionary must be unique

**Status : Correct**

**Marks : 1/1**

7. 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**

8. What is the output of the following code?

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

**Answer**

False

**Status : Correct**

**Marks : 1/1**

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

**Answer**

Keys must be integers

**Status : Correct**

**Marks : 1/1**

10. 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: 'B', 3: 'C'}

**Status :** Correct

**Marks :** 1/1

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

**Answer**

t[3] = 45

**Status :** Correct

**Marks :** 1/1

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

**Answer**

**Status :** Skipped

**Marks :** 0/1

13. 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}

**Status :** Correct

**Marks :** 1/1

14. 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**

15. Which of the following is a Python tuple?

**Answer**

(1, 2, 3)

**Status : Correct**

**Marks : 1/1**

16. 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**

17. 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**



Scanned with OKEN Scanner

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

Status : Correct

Marks : 1/1

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

Answer

Removes all the key-value pairs

Status : Wrong

Marks : 0/1

19. What will be the output?

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

Answer

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

Status : Correct

Marks : 1/1

20. 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) t=t+(3,4) 2) max(t)

Status : Correct

Marks : 1/1

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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**

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***

```
# Input: Length of the lists  
n = int(input())
```

```
# Input: First list of integers  
list1 = list(map(int, input().split(',')))
```

```
# Input: Second list of integers  
list2 = list(map(int, input().split(',')))
```

```
# Compute the element-wise sum using zip and lambda  
result = tuple(map(lambda x, y: x + y, list1, list2))
```

```
# Output: The result as a tuple  
print(result)
```

Status : Correct

Marks : 10/10

## 2. 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

```
from collections import Counter
```

```
# Input: Number of product IDs
n = int(input())
```

```
# Input: List of product IDs
product_ids = [int(input()) for _ in range(n)]
```

```
# Calculate frequency of each product ID
frequency_dict = dict(Counter(product_ids))
```

```
# Calculate total number of unique product IDs
total_unique_ids = len(frequency_dict)
```

```
# Calculate average frequency
average_frequency = sum(frequency_dict.values()) / total_unique_ids
```

```
# Output the results
```

```
print(frequency_dict)
```

```
print(f"Total Unique IDs: {total_unique_ids}")
```

```
print(f"Average Frequency: {average_frequency:.2f}")
```

Status : Correct

Marks : 10/10

3. 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
# Input: Student IDs who registered for the workshops
registered = set(map(int, input().split()))

# Input: Student IDs who attended the workshops
attended = set(map(int, input().split()))

# Input: Student IDs who dropped out of the workshops
dropped_out = set(map(int, input().split()))

# Determine students who registered and attended
registered_and_attended = registered & attended

# Determine students who registered, attended, and did not drop out
final_participants = registered_and_attended - dropped_out

# Output the results
print(registered_and_attended)
print(final_participants)
```

**Status :** Correct

**Marks :** 10/10

#### 4. 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***

```
# Read number of tuples
N = int(input())

# Read N tuples
items = []
for _ in range(N):
    line = input().strip()
    # Evaluate the input tuple (e.g., (1, [6, 7]))
    item = eval(line)
    items.append(item)

# Read the threshold
threshold = int(input())
```

```
# Collect filtered quantities
filtered_quantities = []
for item_id, quantities in items:
    # Filter using filter() function
    above_threshold = list(filter(lambda x: x > threshold, quantities))
    filtered_quantities.extend(above_threshold)

# Print results space-separated
print(' '.join(map(str, filtered_quantities)))
```

**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**

```
n = int(input()) # Number of customers

customer_data = {}

for _ in range(n):
    data = list(map(int, input().split())) # Reading customer ID and amounts spent
    customer_id = data[0] # Extracting customer ID
    expenditures = data[1:] # List of expenditures

    total_expenditure = sum(expenditures) # Calculating total amount spent
    max_expenditure = max(expenditures) # Finding highest single expenditure

    customer_data[customer_id] = [total_expenditure, max_expenditure] # Storing data

print(customer_data)
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 : 56

#### **Section 1 : Coding**

##### **1. 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**

```
numbers = set(map(int, input().split()))
```

```
ch = int(input())
```

```
print("{", ", ".join(map(str, sorted(numbers, reverse=True))), "}")
```

```
if ch == 1:  
    print(max(numbers))  
elif ch == 2:
```

```
print(min(numbers))
elif ch == 3:
    n1 = int(input())
    if n1 in numbers:
        numbers.remove(n1)
    print("{", ", ".join(map(str, sorted(numbers, reverse=True))), "}")
else:
    print("Invalid choice")
```

Status : Correct

Marks : 10/10

## 2. 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
```

```
def main():
    # Step 1: Read the input list of integers
    numbers = input().split()

    # Step 2: Initialize a set to keep track of seen numbers and an empty list to
    # store unique numbers
    seen = set()
    unique_numbers = []

    # Step 3: Iterate through the input numbers
    for num in numbers:
        # If the number is not in the set, it's unique, so add it to the result
        if num not in seen:
            seen.add(num)
            unique_numbers.append(num)

    # Step 4: Concatenate the unique numbers into a single string
    result = ''.join(unique_numbers)

    # Step 5: Output the result
    print(result)

if __name__ == "__main__":
    main()
```

Status : Correct

Marks : 10/10

### 3. 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***

```
def group_consecutive_ids(ids):
    ids.sort()
    result = []
    temp = [ids[0]]
    for i in range(1, len(ids)):
        if ids[i] == ids[i - 1] + 1:
            temp.append(ids[i])
        else:
            result.append(tuple(temp))
            temp = [ids[i]]
    result.append(tuple(temp))

    return result

n = int(input())
event_ids = [int(input()) for _ in range(n)]
groups = group_consecutive_ids(event_ids)
```

```
for group in groups:  
    print(group)
```

Status : Partially correct

Marks : 6/10

#### 4. 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
```

```
def main():
    n = int(input()) # Read the number of elements
    elements = list(map(int, input().split())) # Read the space-separated integers

    result = []
    for i in range(n - 1):
        result.append((elements[i], elements[i + 1])) # Pair consecutive elements
    result.append((elements[-1], None)) # Pair the last element with None

    print(tuple(result)) # Output the result as a tuple of pairs

if __name__ == "__main__":
    main()
```

Status : Correct

Marks : 10/10

## 5. 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
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def generate_prime_dict(n):
    prime_dict = {}
    num = 2
    count = 1
    while count <= n:
        if is_prime(num):
            prime_dict[count] = num
            count += 1
        num += 1
    return prime_dict

# Read input
n = int(input())
prime_dictionary = generate_prime_dict(n)
print(prime_dictionary)
```

**Status : Correct**

**Marks : 10/10**

## **6. 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
# Read input
n = int(input())

# Generate the dictionary
square_dict = {i: i**2 for i in range(1, n+1)}

# Print the dictionary
print(square_dict)
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: praneeth sai

Email: 241501146@rajalakshmi.edu.in

Roll no: 241501146

Phone: 6380562561

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 : 40

#### **Section 1 : Coding**

##### **1. Problem Statement**

Samantha is working on a text analysis tool that compares two words to find common and unique letters. She wants a program that reads two words, w1, and w2, and performs the following operations:

Print the letters common to both words, in alphabetical order. Print the letters that are unique to each word, in alphabetical order. Determine if the set of letters in the first word is a superset of the letters in the second word. Check if there are no common letters between the two words and print the result as a Boolean value.

Ensure the program ignores case differences and leading/trailing spaces in the input words.

Your task is to help Samantha in implementing the same.

### ***Input Format***

The first line of input consists of a string representing the first word, w1.

The second line consists of a string representing the second word, w2.

### ***Output Format***

The first line of output should display the sorted letters common to both words, printed as a list.

The second line should display the sorted letters that are unique to each word, printed as a list.

The third line should display a Boolean value indicating if the set of letters in w1 is a superset of the set of letters in w2.

The fourth line should display a Boolean value indicating if there are no common letters between w1 and w2.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: program

Peace

Output: [a, p]

[c, e, g, m, o, r]

False

False

### ***Answer***

```
# You are using Python
# Read input words and clean them
w1 = input().strip().lower()
w2 = input().strip().lower()
```

```
# Convert to sets of letters
set1 = set(w1)
set2 = set(w2)
```

```
# Common letters
common = sorted(set1 & set2)

# Unique letters to each word (symmetric difference)
unique = sorted(set1 ^ set2)

# Superset check
is_superset = set1.issuperset(set2)

# Disjoint check (no common letters)
is_disjoint = set1.isdisjoint(set2)

# Output results
print(common)
print(unique)
print(is_superset)
print(is_disjoint)
```

Status : Correct

Marks : 10/10

## 2. Problem Statement

Emily is a librarian who keeps track of books borrowed and returned by her patrons. She maintains four sets of book IDs: the first set represents books borrowed, the second set represents books returned, the third set represents books added to the collection, and the fourth set represents books that are now missing. Emily wants to determine which books are still borrowed but not returned, as well as those that were added but are now missing. Finally, she needs to find all unique book IDs from both results.

Help Emily by writing a program that performs the following operations on four sets of integers:

Compute the difference between the borrowed books (first set) and the returned books (second set). Compute the difference between the added books (third set) and the missing books (fourth set). Find the union of the results from the previous two steps, and sort the final result in descending order.

**Input Format**

The first line of input consists of a list of integers representing borrowed books.

The second line of input consists of a list of integers representing returned books.

The third line of input consists of a list of integers representing added books.

The fourth line of input consists of a list of integers representing missing books.

#### ***Output Format***

The first line of output displays the difference between sets P and Q, sorted in descending order.

The second line of output displays the difference between sets R and S, sorted in descending order.

The third line of output displays the union of the differences from the previous two steps, sorted in descending order.

Refer to the sample output for the formatting specifications.

#### ***Sample Test Case***

Input: 1 2 3

2 3 4

5 6 7

6 7 8

Output: [1]

[5]

[5, 1]

#### ***Answer***

```
# You are using Python
# Read four lines of input and convert them into sets of integers
P = set(map(int, input().split())) # Borrowed books
Q = set(map(int, input().split())) # Returned books
R = set(map(int, input().split())) # Added books
S = set(map(int, input().split())) # Missing books

# Compute differences
```

```
diff1 = sorted(P - Q, reverse=True) # Borrowed but not returned  
diff2 = sorted(R - S, reverse=True) # Added but now missing  
  
# Compute union of the two results and sort descending  
union_diff = sorted(set(diff1) | set(diff2), reverse=True)  
  
# Print results  
print(diff1)  
print(diff2)  
print(union_diff)
```

**Status :** Correct

**Marks :** 10/10

### 3. 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**

```
# Read number of clubs
```

```
k = int(input())
```

```
# Initialize symmetric difference set
```

```
sym_diff = set()
```

```
# Read each club's member list and update symmetric difference
```

```
for _ in range(k):
```

```
    members = set(map(int, input().split()))
```

```
    sym_diff ^= members # Symmetric difference update
```

```
# Output the result
```

```
print(sym_diff)
```

```
print(sum(sym_diff))
```

**Status : Correct**

**Marks : 10/10**

### **4. 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**

```
# Read size and sequence for first DNA tuple
```

```
n = int(input())
```

```
seq1 = list(map(int, input().split()))
```

```
# Read size and sequence for second DNA tuple
```

```
m = int(input())
```

```
seq2 = list(map(int, input().split()))
```

```
# Compare elements at the same positions and collect matches
```

```
matches = [str(seq1[i]) for i in range(min(n, m)) if seq1[i] == seq2[i]]
```

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

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

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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**

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

## 2. 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
s = input()
with open("sentence_file.txt", "w") as f:
    f.write(s)
```

```
with open("sentence_file.txt", "r") as f:
    content = f.read().strip()
words = content.split() if content else []
```

```
print(len(words))
```

Status : Correct

Marks : 10/10

### 3. 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_input = input()  
    n = int(n_input)  
    if n <= 0:  
        print("Error: The length of the list must be a non-negative integer.")  
    else:  
        numbers = []  
        for _ in range(n):  
            try:  
                num_input = input()  
                num = int(num_input)  
                numbers.append(num)  
            except ValueError:  
                print("Error: You must enter a numeric value.")  
                exit()  
        average = sum(numbers) / len(numbers)  
        print(f"The average is: {average:.2f}")  
    except ValueError:  
        print("Error: You must enter a numeric value.")
```

**Status :** Correct

**Marks :** 10/10

## **4. 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**

## **5. 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
```

```
s = input()
```

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

```
    f.write(s)
```

```
with open("text_file.txt", "r") as f:
```

```
    original = f.read()
```

```
print(f"Original String: {original} Upper-Case String: {original.upper()} Lower-Case  
String: {original.lower()}")
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 : 15

#### **Section 1 : MCQ**

1. How do you rename a file?

**Answer**

`os.rename(existing_name, new_name)`

**Status :** Correct

**Marks :** 1/1

2. 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**

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

**Answer**

To display the final output of the program

**Status : Wrong**

**Marks : 0/1**

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

**Answer**

raise Exception()

**Status : Correct**

**Marks : 1/1**

5. 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**

6. What is the default value of reference\_point in the following code?



Scanned with OKEN Scanner

file\_object.seek(offset [,reference\_point])

**Answer**

0

**Status : Correct**

**Marks : 1/1**

7. 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**

(f,rw,pickle,pickle)

**Status : Wrong**

**Marks : 0/1**

8. 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**

9. 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**

10. Which of the following is true about

`fp.seek(10,1)`

**Answer**

Move file pointer ten characters ahead from the beginning of a file

**Status : Wrong**

**Marks : 0/1**

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

**Answer**

error

**Status : Wrong**

**Marks : 0/1**

12. 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**

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



Scanned with OKEN Scanner

```
# 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**

Displays Output

**Status : Correct**

**Marks : 1/1**

14. 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**

15. 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**

16. Which clause is used to clean up resources, such as closing files in



Scanned with OKEN Scanner

Python?

**Answer**

finally

**Status : Correct**

**Marks : 1/1**

17. 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**

18. 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.txt2) fo.name

Status : Wrong

Marks : 0/1

19. 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 error Executed

Status : Correct

Marks : 1/1

20. 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

# Rajalakshmi Engineering College

Name: praneeth sai

Email: 241501146@rajalakshmi.edu.in

Roll no: 241501146

Phone: 6380562561

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 : 40

#### **Section 1 : Coding**

##### **1. Problem Statement**

Write a program to obtain the start time and end time for the stage event show. If the user enters a different format other than specified, an exception occurs and the program is interrupted. To avoid that, handle the exception and prompt the user to enter the right format as specified.

Start time and end time should be in the format 'YYYY-MM-DD HH:MM:SS' If the input is in the above format, print the start time and end time. If the input does not follow the above format, print "Event time is not in the format "

##### ***Input Format***

The first line of input consists of the start time of the event.

The second line of the input consists of the end time of the event.

### **Output Format**

If the input is in the given format, print the start time and end time.

If the input does not follow the given format, print "Event time is not in the format".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 2022-01-12 06:10:00

2022-02-12 10:10:12

Output: 2022-01-12 06:10:00

2022-02-12 10:10:12

### **Answer**

```
# You are using Python
from datetime import datetime
```

```
try:
```

```
    start_time = input()
    end_time = input()
```

```
    datetime.strptime(start_time, '%Y-%m-%d %H:%M:%S')
    datetime.strptime(end_time, '%Y-%m-%d %H:%M:%S')
```

```
    print(start_time, end_time)
```

```
except:
```

```
    print("Event time is not in the format")
```

**Status : Correct**

**Marks : 10/10**

## **2. 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

# Custom Exceptions
class IllegalArgumentException(Exception):
    pass
```

```
class NumberFormatException(Exception):
    pass

class NoSuchElementException(Exception):
    pass

reg_no = input()
mobile_no = input()

try:

    if len(reg_no) != 9:
        raise IllegalArgumentException("Register Number should have exactly 9
characters.")

    if not re.match(r'^\d{2}[A-Za-z]{3}\d{4}$', reg_no):
        raise IllegalArgumentException("Register Number should have the format: 2
numbers, 3 characters, and 4 numbers.")

    if not reg_no.isalnum():
        raise NoSuchElementException("Register Number should only contain digits
and alphabets.")

    if len(mobile_no) != 10:
        raise IllegalArgumentException("Mobile Number should have exactly 10
characters.")

    if not mobile_no.isdigit():
        raise NumberFormatException("Mobile Number should only contain digits.")

    print("Valid")

except (IllegalArgumentException, NumberFormatException,
NoSuchElementException) as e:
    print(f"Invalid with exception message: {e}")
```

### 3. Problem Statement

In the enchanted realm of Academia, you, the Academic Alchemist, are bestowed with a magical quill and a parchment to weave the grades of aspiring students into a tapestry of academic brilliance.

The mission is to craft a Python program that empowers faculty members to enter student grades for any two subjects, stores these magical grades in a mystical file, and then, with a wave of your virtual wand, calculates the GPA to unveil the true essence of academic achievement.

#### ***Input Format***

The input format is a string representing the student's name, any two subjects, and corresponding grades.

After entering grades, they can type 'done' when prompted for the student's name.

#### ***Output Format***

The output should display the (average of grades) calculated GPA with a precision of two decimal places.

The magical grades will be saved in a mystical file named "magical\_grades.txt".

Refer to the sample output for format specifications.

#### ***Sample Test Case***

Input: Alice

Math

95

English

88

done

Output: 91.50

### **Answer**

```
with open("magical_grades.txt", "w") as file:  
    while True:  
        student_name = input()  
        if student_name == 'done':  
            break  
        subject1 = input()  
        grade1 = int(input())  
        subject2 = input()  
        grade2 = int(input())  
  
        file.write(f"{student_name} {subject1} {grade1} {subject2} {grade2}\n")
```

```
with open("magical_grades.txt", "r") as file:  
    for line in file:  
        parts = line.strip().split()  
        grade1 = int(parts[2])  
        grade2 = int(parts[4])  
        gpa = (grade1 + grade2) / 2  
        print(f"{gpa:.2f}")
```

**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***

```
N = int(input())
items_sold = list(map(int, input().split()))
M = int(input())
```

```
if N > 30:
    print("Exceeding limit!")
```

```
else:  
    earnings = [x * M for x in items_sold]  
  
    with open("sales.txt", "w") as file:  
        for amount in earnings:  
            file.write(str(amount) + "\n")
```

```
with open("sales.txt", "r") as file:  
    for line in file:  
        print(line.strip(), end=' ')
```

Status : Correct

Marks : 10/10

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 7\_MCQ**

Attempt : 1  
Total Mark : 20  
Marks Obtained : 18

#### **Section 1 : MCQ**

1. Which of the following is a valid way to import NumPy in Python?

**Answer**

import numpy as np

**Status : Correct**

**Marks : 1/1**

2. What is the output of the following NumPy code?

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
r = arr[2:4]
print(r)
```

**Answer**



Scanned with OKEN Scanner

[3 4]

Status : Correct

Marks : 1/1

3. Which NumPy function is used to calculate the standard deviation of an array?

Answer

numpy.std()

Status : Correct

Marks : 1/1

4. The important data structure of pandas is/are \_\_\_\_.

Answer

Both Series and Data Frame

Status : Correct

Marks : 1/1

5. Which function is used to create a Pandas DataFrame?

Answer

pd.DataFrame()

Status : Correct

Marks : 1/1

6. Minimum number of argument we require to pass in pandas series ?

Answer

0

Status : Wrong

Marks : 0/1

7. What is the purpose of the following NumPy code snippet?

```
import numpy as np  
arr = np.zeros((3, 4))
```

print(arr)

**Answer**

Displays a 3x4 matrix filled with zeros

**Status : Correct**

**Marks : 1/1**

8. Which NumPy function is used to find the indices of the maximum and minimum values in an array?

**Answer**

argmax() and argmin()

**Status : Correct**

**Marks : 1/1**

9. What is the primary purpose of Pandas DataFrame?

**Answer**

To store data in tabular form for analysis and manipulation

**Status : Correct**

**Marks : 1/1**

10. What does NumPy stand for?

**Answer**

Numerical Python

**Status : Correct**

**Marks : 1/1**

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

```
import pandas as pd  
pd.Series([1,2], index= ['a','b','c'])
```

**Answer**

Value Error

**Status : Correct**

**Marks : 1/1**

12. What is the primary data structure used in NumPy for numerical computations?

**Answer**

Array

**Status : Correct**

**Marks : 1/1**

13. What is the result of the following NumPy operation?

```
import numpy as np  
arr = np.array([1, 2, 3])  
r = arr + 5  
print(r)
```

**Answer**

[6 7 8]

**Status : Correct**

**Marks : 1/1**

14. In NumPy, how do you access the first element of a one-dimensional array arr?

**Answer**

arr[0]

**Status : Correct**

**Marks : 1/1**

15. What is the output of the following code?

```
import numpy as np  
a = np.arange(10)  
print(a[2:5])
```

**Answer**

[2, 3, 4]

**Status : Correct**

**Marks : 1/1**

16. In the DataFrame created in the code, what is the index for the row containing the data for 'Jack'?

```
import pandas as pd

data = {'Name': ['Tom', 'Jack', 'nick', 'juli'],
        'marks': [99, 98, 95, 90]}

df = pd.DataFrame(data, index=['rank1',
                               'rank2',
                               'rank3',
                               'rank4'])
print(df)
```

**Answer**

rank2

**Status : Correct**

**Marks : 1/1**

17. What will be the output of the following code snippet?

```
import numpy as np
arr = np.array([1, 2, 3])
result = np.concatenate((arr, arr))
print(result)
```

**Answer**

[1 2 3 1 2 3]

**Status : Correct**

**Marks : 1/1**

18. What does the np.arange(10) function in NumPy do?

**Answer**

Creates an array with values from 0 to 10

**Status : Wrong**

**Marks : 0/1**

19. What is the output of the following NumPy code snippet?

```
import numpy as np  
arr = np.array([1, 2, 3, 4, 5])  
r = arr[arr > 2]  
print(r)
```

**Answer**

[3 4 5]

**Status : Correct**

**Marks : 1/1**

20. Which NumPy function is used to create an identity matrix?

**Answer**

`numpy.identity()`

**Status : Correct**

**Marks : 1/1**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 7\_PAH**

Attempt : 1  
Total Mark : 50  
Marks Obtained : 46

#### **Section 1 : Coding**

##### **1. Problem Statement**

You're analyzing the daily returns of a set of financial assets over a period of time. Each day is represented as a row in a 2D array, where each column represents the return of a specific asset on that day.

Your task is to identify which days had all positive returns across every asset using numpy, and output a boolean array indicating these days.

##### ***Input Format***

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

Each of the next rows lines consists of cols float values representing the returns of the assets for that day.

### **Output Format**

The first line of output prints: "Days where all asset returns were positive:"

The second line of output prints: the boolean array `positive_days`, indicating True for days where all asset returns were positive and False otherwise.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 3 4  
0.01 0.02 0.03 0.04  
0.05 0.06 0.07 0.08  
-0.01 0.02 0.03 0.04

Output: Days where all asset returns were positive:  
[ True True False]

### **Answer**

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

# Read input dimensions
rows, cols = map(int, input().split())

# Read the returns data into a numpy array
data = []
for _ in range(rows):
    row = list(map(float, input().split()))
    data.append(row)

arr = np.array(data)

# Check if all returns in a row are positive (>0)
positive_days = np.all(arr > 0, axis=1)

# Print output
print("Days where all asset returns were positive:", positive_days)
```

**Status : Correct**

**Marks : 10/10**

## 2. Problem Statement

Arjun is a data scientist working on an image processing task. He needs to normalize the pixel values of a grayscale image matrix to scale between 0 and 1. The input image data is provided as a matrix of integers.

Help him to implement the task using the numpy package.

Formula:

To normalize each pixel value in the image matrix:

$$\text{normalized\_pixel} = (\text{pixel} - \text{min\_pixel}) / (\text{max\_pixel} - \text{min\_pixel})$$

where min\_pixel and max\_pixel are the minimum and maximum pixel values in the image matrix, respectively. If all pixel values are the same, the normalized image matrix should be filled with zeros.

### ***Input Format***

The first line of input consists of an integer value, rows, representing the number of rows in the image matrix.

The second line of input consists of an integer value, cols, representing the number of columns in the image matrix.

The next rows lines each consist of cols integer values separated by a space, representing the pixel values of the image matrix.

### ***Output Format***

The output prints: normalized\_image

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 2

3

1 2 3

4 5 6

Output: [[0. 0.2 0.4]  
[0.6 0.8 1.]]

**Answer**

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

# Read inputs
rows = int(input())
cols = int(input())

# Read the image matrix
data = [list(map(int, input().split())) for _ in range(rows)]
img = np.array(data, dtype=float)

min_pixel = img.min()
max_pixel = img.max()

if max_pixel == min_pixel:
    normalized_img = np.zeros_like(img)
else:
    normalized_img = (img - min_pixel) / (max_pixel - min_pixel)

print(normalized_img)
```

**Status :** Correct

**Marks :** 10/10

**3. Problem Statement**

A software development company wants to classify its employees based on their years of service at the company. They want to categorize employees into three experience levels: Junior (less than 3 years), Mid (3 to 6 years, inclusive), and Senior (more than 6 years).

**Experience Level Classification:**

Junior: Years at Company < 3

Mid:  $3 \leq$  Years at Company < 6

Senior: Years at Company > 5

You need to create a Python program using the pandas library that reads employee data, processes it into a DataFrame, and adds a new column "Experience Level" to display the appropriate classification for each employee.

#### ***Input Format***

First line: an integer n representing the number of employees.

Next n lines: each line has a string Name and a floating-point number Years at Company (space-separated).

#### ***Output Format***

First line: "Employee Data with Experience Level:"

The employee data table printed with no index column, and with columns: Name, Years at Company, Experience Level.

Refer to the sample output for the formatting specifications.

#### ***Sample Test Case***

Input: 5

Alice 2

Bob 4

Charlie 7

Diana 3

Evan 6

Output: Employee Data with Experience Level:

Name	Years at Company	Experience Level
Alice	2.0	Junior
Bob	4.0	Mid
Charlie	7.0	Senior
Diana	3.0	Mid
Evan	6.0	Senior

#### ***Answer***

```
import pandas as pd
```

```
n = int(input().strip())

employee_data = []

for _ in range(n):
    name, years = input().strip().split()
    years = float(years)
    employee_data.append([name, years])

df = pd.DataFrame(employee_data, columns=["Name", "Years at Company"])

def classify_experience(years):
    if years < 3:
        return "Junior"
    elif 3 <= years < 6:
        return "Mid"
    else:
        return "Senior"

df["Experience Level"] = df["Years at Company"].apply(classify_experience)

print("Employee Data with Experience Level:")
print(df.to_string(index=False))
```

**Status :** Partially correct

**Marks :** 6/10

#### 4. Problem Statement

A company conducted a customer satisfaction survey where each respondent provides their RespondentID and an optional textual Feedback. Sometimes, respondents submit their ID without any feedback or with empty feedback.

Your task is to process the survey responses using pandas to replace any missing or empty feedback with the phrase "No Response". Finally, print the cleaned survey responses exactly as shown in the sample output.

### ***Input Format***

The first line contains an integer  $n$ , the number of survey responses.

Each of the next  $n$  lines contains:

A RespondentID (a single alphanumeric string without spaces),

Followed optionally by a Feedback string, which may be empty or missing.

If no feedback is provided after the RespondentID, treat it as missing.

### ***Output Format***

Print the line:

Survey Responses with Missing Feedback Filled:

Then print the cleaned survey data as a table with two columns: RespondentID and Feedback.

The table should have the headers exactly as:

RespondentID Feedback

Print each respondent's data on a new line, aligned to match the output produced by `pandas.DataFrame.to_string(index=False)`.

For any missing or empty feedback, print "No Response" in the Feedback column.

Maintain the spacing and alignment exactly as shown in the sample outputs.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 4  
101 Great service  
102  
103 Loved it  
104

Output: Survey Responses with Missing Feedback Filled:

RespondentID	Feedback
101	Great service
102	No Response
103	Loved it
104	No Response

### **Answer**

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

# Read number of responses
n = int(input())

# Prepare list to collect the data
data = []

# Read each line of input
for _ in range(n):
    line = sys.stdin.readline().rstrip()
    if not line.strip():
        continue # Skip empty lines
    parts = line.split(maxsplit=1)
    respondent_id = parts[0]
    feedback = parts[1].strip() if len(parts) > 1 and parts[1].strip() else "No
Response"
    data.append([respondent_id, feedback])

# Create DataFrame
df = pd.DataFrame(data, columns=["RespondentID", "Feedback"])
```

```
# Print output
print("Survey Responses with Missing Feedback Filled:")
print(df.to_string(index=False))
```

Status : **Correct**

Marks : **10/10**

## 5. Problem Statement

Arjun manages a busy customer service center and wants to analyze the distribution of customer wait times to improve service efficiency. He decides to group the wait times into intervals of 5 minutes each and count how many customers fall into each interval bucket.

Help him implement this bucketing and counting task using NumPy.

**Bucketing Logic:**

Divide the wait times into intervals (buckets) of size 5 minutes, e.g.:

[0–5), [5–10), [10–15), ...

Use NumPy's digitize function to determine which bucket each wait time falls into.

Count the number of wait times in each bucket and generate bucket labels.

### ***Input Format***

The first line contains an integer n, the number of customer wait times recorded.

The second line contains n space-separated floating-point numbers representing the wait times (in minutes).

### ***Output Format***

The first line of output is the text:

Wait Time Buckets and Counts:

Each subsequent line prints the bucket range and the number of wait times in that bucket, formatted as:

<bucket\_range>: <count>

where <bucket\_range> is the lower and upper bound of the bucket (inclusive lower bound, exclusive upper bound), for example:

0-5: 3

5-10: 2

10-15: 1

The output uses the default string formatting of Python's print() function (no extra spaces, no special formatting beyond the specified lines).

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 10  
2.0 3.0 7.0 8.0 12.0 14.0 18.0 19.0 21.0 25.0

Output: Wait Time Buckets and Counts:

0-5: 2  
5-10: 2  
10-15: 2  
15-20: 2  
20-25: 1

### **Answer**

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

# Read input
n = int(input())
wait_times = list(map(float, input().split()))

# Determine max wait time to set up bucket edges
```

```
max_wait = max(wait_times)
bucket_max = math.ceil(max_wait / 5) * 5 # Round up to nearest multiple of 5
bins = np.arange(0, bucket_max + 5, 5) # e.g., [0, 5, 10, 15, ..., bucket_max]

# Bucket the wait times using numpy.digitize
wait_times_array = np.array(wait_times)
bucket_indices = np.digitize(wait_times_array, bins, right=False)

# Count occurrences per bucket
counts = [np.sum(bucket_indices == i) for i in range(1, len(bins))]

# Print results
print("Wait Time Buckets and Counts:")
for i in range(len(counts)):
    print(f'{bins[i]}-{bins[i+1]}: {counts[i]}')
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: praneeth sai

Email: 241501146@rajalakshmi.edu.in

Roll no: 241501146

Phone: 6380562561

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 7\_COD

Attempt : 1

Total Mark : 50

Marks Obtained : 50

#### **Section 1 : Coding**

##### **1. 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

# Read input time series
array1 = np.array(list(map(float, input().split())))
array2 = np.array(list(map(float, input().split())))

# Compute cross-correlation using numpy's correlate function
cross_corr = np.correlate(array1, array2, mode='full')

# Print the output
print("Cross-correlation of the two time series:")
print(cross_corr)
```

**Status : Correct**

**Marks : 10/10**

## **2. 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

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

# Create MultiIndex from city and month
index = pd.MultiIndex.from_tuples(list(zip(cities, months)), names=["City",
"Month"])

# Create DataFrame
df = pd.DataFrame({'Sales': sales}, index=index)

# Group by city to get total sales
total_sales = df.groupby(level='City').sum()

# Output
print("Monthly Sales Data with MultiIndex:")
print(df)
print("\nTotal Sales Per City:")
print(total_sales)
```

**Status : Correct**

**Marks : 10/10**

3. 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 the number of products and months
products, months = map(int, input().split())
```

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

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

# Compute cumulative sales along months (axis=1)
cumulative_array = np.cumsum(sales_array, axis=1)

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

**Status : Correct**

**Marks : 10/10**

#### 4. 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
```

```
# Input
n = int(input())
columns = input().split(',')
```

```
data = [input().split(',') for _ in range(n)]  
  
# Create DataFrame  
df = pd.DataFrame(data, columns=columns)  
  
# Convert Age to numeric (float), coercing errors to NaN  
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')  
  
# Calculate mean age (excluding NaNs)  
mean_age = df['Age'].mean()  
  
# Fill NaN ages with mean age  
df['Age'] = df['Age'].fillna(mean_age)  
  
# Drop rows where Diagnosis is NaN or empty string after stripping  
df['Diagnosis'] = df['Diagnosis'].replace("", np.nan)  
df = df.dropna(subset=['Diagnosis'])  
  
# Reset index  
cleaned_df = df.reset_index(drop=True)  
  
# Output  
print("Cleaned Hospital Records:", cleaned_df)
```

**Status :** Correct

**Marks :** 10/10

## 5. 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 input
n = int(input())
sales = np.array(list(map(float, input().split())))

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

# Output
print(filtered_sales)
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 7\_CY

Attempt : 1  
Total Mark : 50  
Marks Obtained : 47.5

#### **Section 1 : Coding**

##### **1. Problem Statement**

Rekha is a meteorologist analyzing rainfall data collected over 5 years, with monthly rainfall recorded for each year. She wants to find the total rainfall each year and also identify the month with the maximum rainfall for every year.

Help her to implement the task using the numpy package.

Formula:

Yearly total rainfall = sum of all 12 months' rainfall for each year

Month with max rainfall = index of the maximum rainfall value within the 12 months for each year (0-based index)

***Input Format***

The input consists of 5 lines.

Each line contains 12 floating-point values separated by spaces, representing the rainfall data (in mm) for each month of that year.

#### ***Output Format***

The first line of output prints: yearly\_totals

The second line of output prints: max\_rainfall\_months

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 7.0 8.0 9.0 10.0 11.0 12.0  
2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0  
3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0  
4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0  
5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0  
Output: [ 78. 90. 102. 114. 126.]  
[11 11 11 11 11]
```

#### ***Answer***

```
# You are using Python  
import numpy as np  
  
# Read 5 lines of input, each with 12 float values  
data = []  
for _ in range(5):  
    row = list(map(float, input().split()))  
    data.append(row)  
  
# Convert to numpy array  
rainfall_data = np.array(data)  
  
# Calculate yearly totals  
yearly_totals = np.sum(rainfall_data, axis=1)  
  
# Find the index of the max rainfall month per year  
max_rainfall_months = np.argmax(rainfall_data, axis=1)
```

```
# Print outputs in the required format  
print(early_totals)  
print(max_rainfall_months)
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

You are working as a data analyst for a small retail store that wants to track the stock levels of its products. Each product has a unique Name (such as "Toothpaste", "Shampoo", "Soap") and an associated Quantity in stock. Management wants to identify which products have zero stock so they can be restocked.

Write a Python program using the pandas library to help with this task. The program should:

Read the number of products, n. Read n lines, each containing the Name of the product and its Quantity, separated by a space. Convert this data into a pandas DataFrame. Identify and display the Name and Quantity of products with zero stock. If no products have zero stock, display: No products with zero stock.

### ***Input Format***

The first line contains an integer n, the number of products.

The next n lines each contain:

<Product\_ID> <Quantity>

where <Product\_ID> is a single word (e.g., "Shampoo") and <Quantity> is a non-negative integer (e.g., 5).

### ***Output Format***

The first line of output prints:

Products with Zero Stock:

If there are any products with zero stock, the following lines print the pandas DataFrame showing those products with two columns: Product\_ID and Quantity.

The column headers Product\_ID and Quantity are printed in the second line.

Each subsequent line shows the product's name and quantity, aligned under the respective headers, with no index column.

The output formatting (spacing and alignment) follows the default pandas `to_string(index=False)` style.

If no products have zero stock, print:

No products with zero stock.

Refer to the sample output for the formatting specifications.

#### **Sample Test Case**

Input: 3

P101 10

P102 0

P103 5

Output: Products with Zero Stock:

Product\_ID Quantity

P102 0

#### **Answer**

# You are using Python

```
import pandas as pd

n = int(input())

# Read product data
products = []
for _ in range(n):
    entry = input().split()
    products.append([entry[0], int(entry[1])])

# Create DataFrame
df = pd.DataFrame(products, columns=["Product_ID", "Quantity"])

# Filter products with zero stock
zero_stock_df = df[df["Quantity"] == 0]

# Output
print("Products with Zero Stock:")
if zero_stock_df.empty:
    print("No products with zero stock.")
else:
    print(zero_stock_df.to_string(index=False))
```

**Status :** Partially correct

**Marks :** 7.5/10

### 3. Problem Statement

Rekha works as an e-commerce data analyst. She receives transaction data containing purchase dates and needs to extract the month and day from these dates using the pandas package.

Help her implement this task by performing the following steps:

Convert the Purchase Date column to datetime format, treating invalid date entries as NaT (missing).

Create two new columns:

Purchase Month, containing the month (as an integer) extracted from the Purchase Date.

Purchase Day, containing the day (as an integer) extracted from the Purchase Date. Keep the rest of the data as is.

### ***Input Format***

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

The second line contains the CSV header – comma-separated column names.

The next n lines each contain a transaction record in comma-separated format.

### ***Output Format***

The first line of output is the text:

Transformed E-commerce Transaction Data:

The next lines print the pandas DataFrame with:

The original columns (including Purchase Date, which is now in datetime format or NaT if invalid).

Two additional columns: Purchase Month and Purchase Day.

The output uses the default pandas DataFrame string representation as produced by print(transformed\_df).

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 3

Customer,Purchase Date

Alice,2023-05-15

Bob,2023-06-20

Charlie,2023-07-01

Output: Transformed E-commerce Transaction Data:

	Customer	Purchase Date	Purchase Month	Purchase Day
0	Alice	2023-05-15	5	15
1	Bob	2023-06-20	6	20
2	Charlie	2023-07-01	7	1

### **Answer**

```
# You are using Python
import pandas as pd
n=int(input())
header=input().split(',')
data=[input().split(',') for _ in range(n)]
df=pd.DataFrame(data,columns=header)
df['Purchase Date']=pd.to_datetime(df['Purchase Date'],errors='coerce')
df['Purchase Month']=df['Purchase Date'].dt.month
df['Purchase Day']=df['Purchase Date'].dt.day
print("Transformed E-commerce Transaction Data:")
print(df)
```

**Status : Correct**

**Marks : 10/10**

### **4. Problem Statement**

Arjun is monitoring hourly temperature data recorded continuously for multiple days. He needs to calculate the average temperature for each day based on 24 hourly readings.

Help him to implement the task using the numpy package.

**Formula:**

Reshape the temperature readings into rows where each row has 24 readings (one day).

Average temperature per day = mean of 24 hourly readings in each row.

#### ***Input Format***

The first line of input consists of an integer value, n, representing the total number of temperature readings.

The second line of input consists of n floating-point values separated by spaces, representing hourly temperature readings.

#### ***Output Format***

The output prints: avg\_per\_day

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 30

30.0 30.0

Output: [30.]

### **Answer**

```
# You are using Python
import numpy as np
n=int(input())
data=list(map(float,input().split()))
arr=np.array(data).reshape(-1,24)
avg_per_day=np.mean(arr, axis=1)

print(avg_per_day)
```

Status : **Correct**

Marks : **10/10**

## **5. Problem Statement**

Arjun is developing a system to monitor environmental sensors installed in different rooms of a smart building. Each sensor records multiple temperature readings throughout the day. To compare sensor data fairly despite differing scales, Arjun needs to normalize each sensor's readings so that they have a mean of zero and standard deviation of one.

Help him implement this normalization using numpy.

Normalization Formula:

### **Input Format**

The first line of input consists of two integers: sensors (number of sensors) and samples (number of readings per sensor).

The next sensors lines each contain samples space-separated floats representing the sensor readings.

### ***Output Format***

The first line of output prints: "Normalized Sensor Data."

The next lines print the normalized readings as a numpy array, where each row corresponds to a sensor's normalized values.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 3  
1.0 2.0 3.0  
4.0 5.0 6.0  
7.0 8.0 9.0

Output: Normalized Sensor Data:

```
[[-1.22474487 0. 1.22474487]  
 [-1.22474487 0. 1.22474487]  
 [-1.22474487 0. 1.22474487]]
```

### ***Answer***

```
# You are using Python  
import numpy as np  
  
sensors, samples = map(int, input().split())  
data = [list(map(float, input().split())) for _ in range(sensors)]  
arr = np.array(data)  
mean = arr.mean(axis=1, keepdims=True)  
std = arr.std(axis=1, keepdims=True)  
normalized = (arr - mean) / std  
print("Normalized Sensor Data:")  
print(normalized)
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: praneeth sai  
Email: 241501146@rajalakshmi.edu.in  
Roll no: 241501146  
Phone: 6380562561  
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 : 30

#### **Section 1 : Coding**

##### **1. 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**

```
# You are using Python
MAX_CAPACITY = 30

try:
    n = int(input())
    students = {}
    for _ in range(n):
        if len(students) >= MAX_CAPACITY:
            raise Exception("Student database is full.")
```

```
sid_name = input().strip().split()
sid = int(sid_name[0])
name = sid_name[1]
if sid in students:
    raise ValueError("Student ID already exists.")
students[sid] = name
print(f"Student with ID {sid} added to the database.")
except ValueError as ve:
    print(f"Exception caught. Error: {ve}")
except Exception as e:
    print(f"Exception caught. Error: {e}")
```

**Status :** Correct

**Marks :** 10/10

## 2. 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 is 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**

```
# Read inputs
filename = input().strip()
content = input()
char_to_count = input()

# Write the content to the specified file
with open(filename, 'w') as file:
    file.write(content)

# Read the content back from the file
with open(filename, 'r') as file:
    file_content = file.read()

# Count occurrences of the character (case-insensitive)
count = file_content.lower().count(char_to_count.lower())

# Print output based on count
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**

#### **3. 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***

```
import sys

try:
    line = input().strip()
    numbers_str = line.split()

    numbers = []
    for num_str in numbers_str:
        try:
            num = float(num_str)
            numbers.append(num)
        except ValueError:
```

```
print("Invalid data in the input.")  
    sys.exit()  
  
if len(numbers) == 0:  
    print("Invalid data in the input.")  
    sys.exit()  
  
avg = sum(numbers) / len(numbers)  
print(f"Average of the numbers is: {avg:.2f}")  
  
except Exception:  
    print("Invalid data in the input.")
```

Status : Correct

Marks : 10/10