Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 12

Section 1 : MCQ

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

Answer

^

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

2. Which of the following can convert the string to a float number?

Answer

float(str)

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

3. What is typecasting in Python?

Answer

Change data type property

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

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

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

Answer

27.2

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

5. What is the value of the following expression?

float(22//3+3/3)

Answer

8.0

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

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

Answer

int('10.8')

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

7. What is the return type of the function id?

Answer

bool

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

8. 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***

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

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

Answer

0b1101

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

10. What is used to concatenate two strings in Python?

Answer

+ operator

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

11. What is the value of the following expression?

8/4/2, 8/(4/2)

Answer

(1.0,4.0)

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

12. What is the output of the following program?

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

Answer

(1, 2, 3, 4)

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

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

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

Answer

Value Error: Missing an imaginary part of a complex number

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

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

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

Answer

30

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

15. Which of these is not a core data type?

Answer

Class

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 5

Section 1 : Coding

1. Problem Statement

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

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

Example:

Input:

5

Output:

Constant value: 2064.5

Explanation:

Find the Average:

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

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

Product = 5 × (5 + 3) = 5 × 8 = 40

Final result: 2104.5 - 40 = 2064.5

Input Format

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

Output: Constant value: 2064.5

Answer

```
a=int(input())
b=0
for i in range(0,4):
    c=(a+i)**4
    b+=c
d=b/4
e=a*(a+3)
f=d-e
print('Constant value:',f)
```

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

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

```
d=int(input())
s=int(input())
c=d^s
print('Offer code:',c)
```

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

3. Problem Statement

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

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

Input Format

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

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

Output Format

The first line of output displays "Initial salaries:"

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

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

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

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

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

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

Answer

```
n=int(input())
f=float(input())
print("Initial salaries:")
print("Alice's salary =",n)
print("Bob's salary =",f)
n,f=f,n
print('New salaries after swapping:')
print("Alice's salary =",n)
```

```
print("Bob's salary =",f)
```

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

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

```
f=float(input())
print("The integer value of",f,f" is:{int(f)}")
```

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

5. 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

```
n=int(input())
m=int(input())
o=n*5
```

```
p=m*3
q=50
r=30
s=o+q
t=p+r
print(f"Older child allowance: Rs.{s}")
print(f"Younger child allowance: Rs.{t}")
print(f"Total allowance: Rs.{s+t}")
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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
def validate_rachels_statement(n):
    # Calculate the sum of n and its triple
    calculated_sum = n + 3 * n

    # Check if the ratio of the sum to n is always 4
    statement_is_true = (calculated_sum / n) == 4

    # Output the results
    print(f"Sum: {calculated_sum}")
    print(f"Rachel's statement is: {statement_is_true}")

# Example usage
n = int(input())
validate_rachels_statement(n)
```

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
def calculate_total_time(episodes, break_after):
    episode_duration = 45  # duration of each episode in minutes
    break_duration = 15     # duration of each break in minutes

    total_time = episodes * episode_duration  # total time for all episodes
    # Calculate the number of breaks needed
    num_breaks = (episodes - 1) // break_after  # breaks only after complete sets
of episodes

    total_time += num_breaks * break_duration  # add total break time
```

```
    return total_time

# Input reading
E = int(input())
B = int(input())

# Calculate and output the total time
total_viewing_time = calculate_total_time(E, B)
print(f"{total_viewing_time} minutes")
```

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

3. 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 MB * 8 bits/byte = 6400 megabits) and divide it by the download speed (6400 Mbps / 40 Mbps = 160 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
def calculate_download_time(file_size_mb, internet_speed_mbps):
    # Convert MB to megabits
    file_size_mbps = file_size_mb * 8  # 1 MB = 8 megabits

    # Calculate the download time in seconds
    download_time_seconds = file_size_mbps / internet_speed_mbps

    # Convert seconds to hours, minutes, and seconds
    hours = int(download_time_seconds // 3600)
    minutes = int((download_time_seconds % 3600) // 60)
    seconds = int(download_time_seconds % 60)

    return hours, minutes, seconds

# Input format
N = int(input())
```

```
S = int(input())

# Calculate download time
hours, minutes, seconds = calculate_download_time(N, S)

# Output format
print(f"Download Time: {hours} hours, {minutes} minutes, and {seconds}
seconds")
```

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

4. 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
def calculate_ingredients(n):
    # Original recipe quantities for 15 cookies
    original_cookies = 15
    flour_per_15 = 2.5  # cups of flour
    sugar_per_15 = 1.0   # cups of sugar
    butter_per_15 = 0.5  # cups of butter

    # Calculate the quantities for n cookies
    flour_needed = (flour_per_15 / original_cookies) * n
    sugar_needed = (sugar_per_15 / original_cookies) * n
    butter_needed = (butter_per_15 / original_cookies) * n

    # Print the results rounded to two decimal places
    print(f"Flour: {flour_needed:.2f} cups")
    print(f"Sugar: {sugar_needed:.2f} cups")
    print(f"Butter: {butter_needed:.2f} cups")

# Input: number of cookies
n = int(input())
calculate_ingredients(n)
```

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

5. 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
# Function to record and display car details
def record_car_details():
    # Input for car make, model, and price
    car_make = input()
    car_model = input()
    car_price = float(input())

    # Output the formatted summary
    print(f"Car Make: {car_make}")
    print(f"Car Model: {car_model}")
    print(f"Price: Rs.{car_price:.2f}")
```

```
# Call the function to execute
record_car_details()
```

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

6. 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
# Function to display the climate details
def display_climate_details(room, temperature, humidity):
    print(f"Room: {room}")
    print(f"Temperature: {temperature:.2f}")
    print(f"Humidity: {humidity:.2f}%")

# Main program
if __name__ == "__main__":
    room_name = input()  # Input for room name
    temperature = float(input())  # Input for temperature
    humidity = float(input())  # Input for humidity

    # Display the climate details
    display_climate_details(room_name, temperature, humidity)
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Alex is an air traffic controller who needs to record and manage flight delays efficiently. Given a flight number, the delay in minutes (as a string), and the coordinates of the flight's current position (as a complex number),

Help Alex convert and store this information in a structured format.

Input Format

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

The second line consists of a string representing the delay in minutes.

The third line consists of two floats separated by a space, representing the real and imaginary parts of the complex number for the flight's position.

Output Format

The first line of output displays the complex number.

The second line displays a string with the flight number, delay, and the real and imaginary parts of the complex number, separated by commas.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 12345
30.5
12.3 45.6

Output: (12.3+45.6j)
12345, 30.5, 12.3, 45.6

Answer

```
n=int(input())
a=float(input())
b,c=input().split()
d=float(b)
f=float(c)
print(complex(d,f))
print(f"{n}, {a}, {d}, {f}")
```

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

2. Problem Statement

Shawn is planning for his younger sister's college education and wants to ensure she has enough funds when the time comes. He starts with an initial principal amount and plans to make regular monthly contributions to a savings account that offers a fixed annual interest rate.

Shawn needs to calculate the total amount that will accumulate by the time his sister is ready for college. Your task is to write a program that calculates the final amount in the savings account based on the initial

principal, monthly contributions, annual interest rate, and the number of months the money is invested.

Formula:

$A = P \times (1 + r/n)^{(n \times t)} + C \times [((1 + r/n)^{(n \times t)} - 1) / (r/n)]$

Where:

A = Final amount after the specified time

P = Initial principal amount

C = Monthly contribution

r = Annual interest rate (as a decimal, e.g., 5% = 0.05)

n = Number of compounding periods per year (12 for monthly compounding)

t = Total time in years (months / 12)

Input Format

The first line of input consists of a float P, representing the initial principal amount.

The second line of input consists of a float R, representing the annual interest rate (in percentage).

The third line of input consists of a float C, representing the monthly contribution.

The fourth line of input consists of an integer M, representing the number of months.

Output Format

The output displays "Final amount after X months: Rs." followed by the total accumulated amount, formatted to two decimal places, where X is the number of months.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 10000.0
5.0
2000.0
12

Output: Final amount after 12 months: Rs.35069.33

Answer

```
# You are using Python
p=float(input())
R=float(input())
c=float(input())
m=int(input())
n=12
t=m/12
r=R/100
A=p*(1+r/n)**(n*t)+c*((((1+r/n)**(n*t))-1)/(r/n))
print(f"Final amount after {m} months: Rs.{A:.2f}")
```

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

3. 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

```
n=int(input())

if(n>=10):
    print("Number of tacos needed:",n*2+5)
    print("Total cost:",((n*2)+5)*25)
elif(n<10):
    print("Number of tacos needed:",20)
    print("Total cost:",(20*25))
```

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

4. 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

```
s1=int(input())
s2=int(input())
s3=int(input())
a=s1&s2
b=a^s3
c=b*3%1000
print(c)
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 3
Total Mark : 15
Marks Obtained : 12

Section 1 : MCQ

1. What is the purpose of the pass statement in Python?

Answer

To do nothing and act as a placeholder.

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

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

```
i = 1
while True:
    if i%0O7 == 0:
        break
    print(i)
    i += 1
```

Answer

error

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

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

Answer

When loop condition becomes false

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

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

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

Answer

1 2

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

5. 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***

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

```
i = 5
```

```
while True:
   if i%0O11 == 0:
      break
   print(i, end = " ")
   i += 1
```

Answer

Compile Time Error

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

7. What is the output of the following?

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

Answer

error

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

8. Which keyword is used to immediately terminate a loop?

Answer

break

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

9. 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***

10. 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

Compile Time Error

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

11. What is the output of the following code?

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

Answer

0 1 2 3 4 Here

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

12. 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***

13. 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***

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

```
balloon_inflated = False
while not balloon_inflated:
    if not balloon_inflated:
        balloon_inflated = True
        print("inflate-", end="")
print("done")
```

Answer

inflate-done

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

15. 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***

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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

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
def is_palindrome(n):
    return str(n) == str(n)[::-1]

def digit_sum(n):
    return sum(int(digit) for digit in str(n))
```

```
def non_palindromic_digit_sum(start, end):
    total_sum = 0
    for num in range(start, end + 1):
        if not is_palindrome(num):
            total_sum += digit_sum(num)
    return total_sum

# Input
start = int(input())
end = int(input())

# Output
result = non_palindromic_digit_sum(start, end)
print(result)
```

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

2. Problem Statement

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

Help him write a program to list these numbers.

Input Format

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
100

Output: 6 28

Answer

```
# You are using Python
def is_perfect_number(num):
    # Calculate the sum of proper divisors
    proper_divisors_sum = sum(i for i in range(1, num) if num % i == 0)
    return proper_divisors_sum == num

def find_perfect_numbers(start, end):
    perfect_numbers = []
    for num in range(start, end + 1):
        if is_perfect_number(num):
            perfect_numbers.append(num)
    return perfect_numbers

# Input
start = int(input())
end = int(input())

# Finding and printing perfect numbers
perfect_numbers = find_perfect_numbers(start, end)
print(" ".join(map(str, perfect_numbers)))
```

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

3. Problem Statement

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

Input Format

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

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 10

Output: 4
16
36
64
100

Answer

```
# You are using Python
def print_even_squares(n):
    for i in range(2, n + 1, 2):  # Start from 2 and go up to n with a step of 2
        print(i * i)

# Input from the user
n = int(input())
if 1 <= n <= 30:
    print_even_squares(n)
else:
    print("Please enter a number between 1 and 30.")
```

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

4. Problem Statement

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

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

Input Format

The input consists of a string representing the sentence.

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

```
# You are using Python
def display_consonants(sentence):
    # Define vowels
    vowels = "aeiouAEIOU"
    consonants = []

    # Iterate through each character in the sentence
    for char in sentence:
        # Check if the character is an alphabetic character and not a vowel
        if char.isalpha() and char not in vowels:
            consonants.append(char)

    # Join the list of consonants with spaces and print the result
    print(" ".join(consonants))

# Take user input
user_input = input()
display_consonants(user_input)
```

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

5. Problem Statement

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

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

Input Format

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
10

Output: 5

Answer

```
# You are using Python
def is_fibonacci(n):
    # Function to check if a number is a Fibonacci number
    a, b = 0, 1
    while a < n:
        a, b = b, a + b
    return a == n

# Input from the user
start = int(input())
end = int(input())
```

```
non_fibonacci_count = 0

for num in range(start, end + 1):
    if is_fibonacci(num):
        continue  # If it's a Fibonacci number, skip to the next iteration
    non_fibonacci_count += 1  # Count non-Fibonacci numbers

print(non_fibonacci_count)
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 60

Section 1 : Coding

1. 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
def collatz_conjecture(n):
    steps = 0

    while n != 1:
        if steps > 100:
            print("Exceeded 100 steps. Exiting...")
            break

        if n % 2 == 0:
            n = n // 2
        else:
            n = 3 * n + 1

        steps += 1

    if steps <= 100:
        print(f"Steps taken to reach 1: {steps}")

# Input
try:
    n = int(input())
    if 1 <= n <= 100:
```

```
        collatz_conjecture(n)
    else:
        print("Please enter a number between 1 and 100.")
except ValueError:
    print("Invalid input. Please enter a valid integer.")
```

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

2. 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
# Get the starting and ending numbers from the user
start = int(input())
end = int(input())

# Initialize the sum variable
total_sum = 0

# Loop through the range from start to end (inclusive)
for number in range(start, end + 1):
    if number % 3 == 0:
        continue  # Skip the current iteration if the number is a multiple of 3
    total_sum += number  # Add number to total_sum if it's not a multiple of 3

# Print the resulting sum
print(total_sum)
```

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

3. 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
# Function to determine if a year is a leap year
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    return False

# Main program
def find_leap_years(start_year, end_year):
    leap_years = []
    for year in range(start_year, end_year + 1):
        if is_leap_year(year):
            leap_years.append(year)
    return leap_years

# Input
start_year = int(input())
end_year = int(input())

# Find and print leap years
leap_years = find_leap_years(start_year, end_year)
for year in leap_years:
    print(year)
```

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

4. 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

```
# You are using Python
def calculate_electricity_bill(units):
    if units <= 100:
        amount = 0
    elif units <= 200:
        amount = (units - 100) * 5
    else:
```

```
        amount = (100 * 5) + ((units - 200) * 10)

    return amount

# Input: Number of units consumed
units = int(input())
bill_amount = calculate_electricity_bill(units)

# Output: Total amount to be paid
print(f"Rs. {bill_amount}")
```

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
def sum_even_odd(N):
    sum_even = 0
    sum_odd = 0

    for num in range(1, N + 1):
        if num % 2 == 0:
            sum_even += num
        else:
            sum_odd += num

    print(f"Sum of even numbers from 1 to {N} is {sum_even}")
    print(f"Sum of odd numbers from 1 to {N} is {sum_odd}")

# Input
N = int(input())
sum_even_odd(N)
```

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

6. 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

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

```
# You are using Python
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

def sum_of_digits(num):
    return sum(int(digit) for digit in str(num))

def odd_factorial_sum(n):
    output = []
    for i in range(1, n + 1, 2):  # Iterate through odd numbers only
        fact = factorial(i)
        digit_sum = sum_of_digits(fact)
        output.append(f"{i}! = {fact}, sum of digits = {digit_sum}")

    return output

# Input
n = int(input())
results = odd_factorial_sum(n)

# Output
for result in results:
    print(result)
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. 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

```
# You are using Python
# Read the selected lighting mode and initial intensity level
n = int(input())
m = float(input())

# Define the intensity multipliers for each mode
if n == 1:
    final_intensity = m * 1.5
    print(f"Intensity: {final_intensity:.2f}")
elif n == 2:
    final_intensity = m * 2.0
    print(f"Intensity: {final_intensity:.2f}")
elif n == 3:
    final_intensity = m * 1.8
    print(f"Intensity: {final_intensity:.2f}")
else:
```

```
    print("Invalid")
```

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

2. Problem Statement

Taylor is tasked with a mathematical challenge that requires finding the smallest positive number divisible by all integers from 1 to n.

Help Taylor to determine the smallest positive number that is divisible by all integers from 1 to n. Make sure to employ the break statement to ensure efficiency in the program.

Input Format

The input consists of a single integer, n.

Output Format

The output displays the smallest positive number that is divisible by all integers from 1 to n.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 10
Output: 2520

Answer

```
# You are using Python
import math
from functools import reduce

def lcm(a, b):
    return abs(a * b) // math.gcd(a, b)

def smallest_multiple(n):
    return reduce(lcm, range(1, n + 1))
```

```
# Input
n = int(input())

# Output
print(smallest_multiple(n))
```

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

3. Problem Statement

Rohith is a data analyst who needs to categorize countries based on their population growth rates. Each country is assigned a unique code. Rohith will receive a code and corresponding data based on the code. If the data falls within specific thresholds, he needs to classify the country's priority level.

Your task is to write a program that reads a country code and its associated data, and then determines if the priority is "High" or "Low."

Thresholds:France: Priority is "High" if the percentage < 50, else "Low".Japan: Priority is "High" if life expectancy > 80, else "Low".Brazil: Priority is "High" if the urban population > 80, else "Low".

Input Format

The first line of input consists of an integer, representing the country code (1 for France, 2 for Japan, 3 for Brazil).

If the country code is 1,

- The second line consists of a floating-point value N, representing the percentage of the English-speaking population.

If the country code is 2,

- The second line consists of a floating-point value A, representing the average life expectancy in years.

If the country code is 3,

- The second line consists of a floating-point value P, representing the percentage of the urban population.

Output Format

The first line of output displays "Priority: High" or "Priority: Low" based on the input data.

If the country code is invalid, print "Invalid".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
30.0

Output: Priority: High

Answer

```
# You are using Python
def categorize_priority():
    # Read the country code input
    country_code = int(input())

    if country_code == 1:  # France
        percentage = float(input())
        if percentage < 50:
            print("Priority: High")
        else:
            print("Priority: Low")

    elif country_code == 2:  # Japan
        life_expectancy = float(input())
        if life_expectancy > 80:
            print("Priority: High")
        else:
            print("Priority: Low")

    elif country_code == 3:  # Brazil
        urban_population = float(input())
        if urban_population > 80:
```

```
        print("Priority: High")
    else:
        print("Priority: Low")

else:
    print("Invalid")

# Call the function
categorize_priority()
```

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

4. Problem Statement

Nisha is a mathematics enthusiast, eager to explore the realm of twin prime numbers. The objective is to develop a program that enables the discovery and presentation of twin prime pairs.

The program should take an integer 'n' as input and generate 'n' pairs of twin primes, displaying the pairs with a difference of 2 between them.

Input Format

The input consists of a single integer, n.

Output Format

The output displays the 'n' pairs of twin primes, the pairs with a difference of 2 between them.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5

Output: 3 5
5 7
11 13
17 19

29 31

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 find_twin_primes(n):
    count = 0
    num = 3  # Start checking for twin primes from 3
    twin_primes = []

    while count < n:
        if is_prime(num) and is_prime(num + 2):
            twin_primes.append((num, num + 2))
            count += 1
        num += 2  # Check only odd numbers

    return twin_primes

def main():
    n = int(input())
    twin_prime_pairs = find_twin_primes(n)

    for pair in twin_prime_pairs:
        print(pair[0], pair[1])

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

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 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***

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

Answer

Adds a new element to the end of the list

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

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

Answer

append()

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

5. 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***

6. 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***

7. 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***

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

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

Answer

nohtyp

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

9. 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***

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

Answer

25

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

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

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

Answer

" Python"

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

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

Answer

[2, 33, 222, 14]

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

13. 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***

14. 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***

15. 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***

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

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

Answer

Python

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

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

Answer

[2, 3, 4]

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

18. 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***

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

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

Answer

oje

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

20. 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***

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

Answer

list()

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

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

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

Answer

3

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

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

Answer

Access elements in the list from the end

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

24. 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***

25. 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***

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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

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

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

Input Format

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

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

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

Output Format

The output displays the following format:

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: pythonprogramming
0
5
Output: python

Answer

```
# You are using Python
# Get input string
input_string = input()

# Get start and end positions
start = int(input())
end = int(input())

# Validate positions
if 0 <= start <= end < len(input_string):
    print(input_string[start:end+1])  # Slicing the string
else:
    print("Invalid start and end positions")
```

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

2. Problem Statement

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

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

Input Format

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

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

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

Output Format

The first line of output displays the original list.

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

The third line of output displays the popped element.

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

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

for _ in range(n):
    elements.append(int(input()))

M = int(input())

print("List after appending elements:", elements)
popped_element = elements.pop(M)
print("List after popping last element:", elements)
print("Popped element:", popped_element)
```

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

3. Problem Statement

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

Example

Input:

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

Output:

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

Explanation:

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

Input Format

The input consists of a single line containing a list of integers enclosed in square

brackets separated by commas.

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

```
# You are using Python
import ast

input_string = input()
arr = ast.literal_eval(input_string)

negative_nums = [num for num in arr if num < 0]
positive_nums = [num for num in arr if num >= 0]

result = negative_nums + positive_nums

print("List =", result)
```

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

4. Problem Statement

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

Write a Python program for the same.

Note

(XXX) - Area code

XXX-XXXX - Phone number

Input Format

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

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: (123) 456-7890

Output: Area code: 123

Answer

```
# You are using Python
phone_number = input()
area_code = phone_number[1:4]
print("Area code:", area_code)
```

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

5. 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

```
first_string = input()
second_string = input()

reversed_second = second_string[::-1]
result = first_string + reversed_second

print(result)
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

def analyze_string(s):
    uppercase = sum(1 for c in s if c.isupper())
    lowercase = sum(1 for c in s if c.islower())
    digits = sum(1 for c in s if c.isdigit())
    special_chars = len(s) - (uppercase + lowercase + digits)

    print(f"Uppercase letters: {uppercase}")
    print(f"Lowercase letters: {lowercase}")
    print(f"Digits: {digits}")
    print(f"Special characters: {special_chars}")

# Take user input
log_entry = input().strip()
analyze_string(log_entry)
```

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

2. 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

```
import re

def count_word_occurrences(text, word):
```

```
    words = re.findall(r'\b' + re.escape(word) + r'\b', text)  # Matches whole words
    return len(words)

# Taking input
text = input().strip()
word = input().strip()

# Printing the output as per format
print(count_word_occurrences(text, word))
```

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

3. 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 an 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

```
def find_peaks(numbers):
    peaks = [numbers[i] for i in range(len(numbers))
             if (i == 0 and numbers[i] > numbers[i+1]) or  # First element condition
                (i == len(numbers) - 1 and numbers[i] > numbers[i-1]) or  # Last element
condition
                (0 < i < len(numbers) - 1 and numbers[i] > numbers[i-1] and numbers[i] >
numbers[i+1])]  # Middle elements
    return peaks

# Taking input
numbers = list(map(int, input().split()))

# Finding and displaying peaks
print("Peaks:", find_peaks(numbers))
```

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

4. 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

```
def find_missing_positive(n, numbers):
    # Filter positive numbers and store them in a set
    positive_nums = set(num for num in numbers if num > 0)

    # Start checking from 1 onwards
    smallest_missing = 1
    while smallest_missing in positive_nums:
        smallest_missing += 1

    return smallest_missing

# Taking input
n = int(input().strip())  # Read number of elements
numbers = list(map(int, input().split()))  # Read space-separated integers

# Finding and printing the smallest missing positive number
```

```
print(find_missing_positive(n, numbers))
```

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

5. 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

```
# Taking input
A = input().strip()

# Finding the length of the string using len()
length_A = len(A)
```

```
# Copying the string using built-in functionality
B = A  # Simple assignment creates a copy of the string

# Printing the required outputs
print(f"Length of the string: {length_A}")
print(f"Copied string: {B}")
```

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

```
def transform_list(numbers):
    sorted_numbers = sorted(numbers)  # Sort the list in ascending order

    # Apply transformation rules based on index
    transformed_list = [0 if i == 0 else (sorted_numbers[i] ** 3 if i % 2 == 0 else
sorted_numbers[i] ** 2) for i in range(len(sorted_numbers))]

    return transformed_list

# Taking input
N = int(input().strip())  # Read the size of the list
numbers = [int(input().strip()) for _ in range(N)]  # Read N integers

# Displaying the original list (sorted)
sorted_numbers = sorted(numbers)
print("Original List:", sorted_numbers)

# Transforming and displaying the replaced list
print("Replaced List:", transform_list(sorted_numbers))
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 30

Section 1 : Coding

1. 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'}
    consonants = set("bcdfghjklmnpqrstvwxyz")

    # Condition 1: Check if the string starts with a consonant
    if s[0] not in consonants:
        return False

    used_consonants = set()
    prev_was_vowel = False

    for i in range(len(s)):
        char = s[i]

        if char in consonants:
            # Condition 3: Each consonant appears exactly once
            if char in used_consonants:
                return False
            used_consonants.add(char)

            # Condition 2: Alternates between consonants and vowels
            if i > 0 and not prev_was_vowel:
                return False

            prev_was_vowel = False  # Mark consonant occurrence

        elif char in vowels:
            # Condition 4: Vowels can appear consecutively but should not be
```

```
followed immediately by a consonant
            prev_was_vowel = True

    return True

# Read input string
s = input().strip()

# Print output
print(is_perfect_string(s))
```

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

2. 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

```
def manage_shopping_list():
    # Read initial list and convert elements to integers
    shopping_list = list(map(int, input().strip().split()))

    print(f"List1: {shopping_list}")
```

```
    # Read item to remove and convert to integer
    item_to_remove = int(input().strip())

    if item_to_remove in shopping_list:
        shopping_list.remove(item_to_remove)
        print(f"List after removal: {shopping_list}")
    else:
        print("Element not found in the list")

    # Read new items and convert elements to integers
    new_items = list(map(int, input().strip().split()))
    shopping_list.extend(new_items)

    print(f"Final list: {shopping_list}")

# Run the function
manage_shopping_list()
```

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

3. Problem Statement

You have two strings str1 and str2, both of equal length.

Write a Python program to concatenate the two strings such that the first character of str1 is followed by the first character of str2, the second character of str1 is followed by the second character of str2, and so on.

For example, if str1 is "abc" and str2 is "def", the output should be "adbecf".

Input Format

The input consists of two strings in each line.

Output Format

The output displays the concatenated string in the mentioned format.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: abc
def

Output: adbecf

Answer

```
def interleave_strings(str1, str2):
    return ''.join(a + b for a, b in zip(str1, str2))

# Read input strings
str1 = input().strip()
str2 = input().strip()

# Ensure both strings are of equal length
if len(str1) == len(str2):
    print(interleave_strings(str1, str2))
else:
    print("Error: Strings must be of equal length.")
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 is the output of the code shown?

```
def f1():
 global x
 x+=1
 print(x)
x=12
print("x")
```

Answer

13

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

2. What keyword is used to define a lambda function in Python?

Answer

lambda

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

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 is the output of the following code snippet?

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

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

Answer

1

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

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

Answer

map()

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

6. 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***

7. 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***

8. 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***

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

```
num = -5
```

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

Answer

5

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

10. 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***

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

```
def fun(x, y=2, z=3):
    return x + y + z

result = fun(1, z=4)
print(result)
```

Answer

7

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

12. How is a lambda function different from a regular named function in Python?

Answer

A lambda function does not have a name, while a regular function does

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

13. What is the output of the code shown?

```
def f():
 global a
 print(a)
 a = "hello"
 print(a)
a = "world"
f()
print(a)
```

Answer

hellohelloworld

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

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

```
multiply = lambda x, y: x * y
print(multiply(2, 'Hello'))
```

Answer

HelloHello

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

15. What is the output of the code shown below?

```
def f1(x):
    x += 1
    print(x)

global_variable = 15
f1(global_variable)
print("hello")
```

Answer

16hello

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 40

Section 1 : Coding

1. Problem Statement

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

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

Help her build this program to automate her calculations.

Input Format

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

Output Format

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

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

The third line prints the sum of all these powers.

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

```
# You are using Python
def exponential_calculator(base, exponent):
    result = pow(base, exponent)
    powers = [pow(base, i) for i in range(1, exponent + 1)]
    total_sum = sum(powers)

    print(result)
    print(powers)
    print(total_sum)

# Read inputs
base = int(input())
exponent = int(input())

# Execute function
exponential_calculator(base, exponent)
```

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

2. Problem Statement

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

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

Input Format

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

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

Output Format

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

Refer to the sample for the formatting specifications.

Sample Test Case

Input: Examly
e
Output: False

Answer

```
# You are using Python
# Lambda function to check if a string starts with the given character or substring
starts_with = lambda s, n: s.startswith(n)

# Read inputs
s = input().strip()
n = input().strip()
```

```
# Print the result
print(starts_with(s, n))
```

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

3. 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

```
# You are using Python
# Function to check if a number is an Armstrong number
def is_armstrong_number(number):
    num_str = str(number)
    num_digits = len(num_str)
```

```
    armstrong_sum = sum(int(digit) ** num_digits for digit in num_str)

    if armstrong_sum == number:
        print(f"{number} is an Armstrong number.")
    else:
        print(f"{number} is not an Armstrong number.")

# Read input
n = int(input().strip())

# Execute function
is_armstrong_number(n)
```

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

4. 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):

uppercase_count = 0
    lowercase_count = 0
    digit_count = 0
    special_count = 0
    for char in input_string:
        if 'A' <= char <= 'Z':
            uppercase_count += 1
        elif 'a' <= char <= 'z':
            lowercase_count += 1
        elif '0' <= char <= '9':
            digit_count += 1
        else:
            special_count += 1

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

5. 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

```
def message_length(message):
    print(len(message))

# Read input
message = input()

# Execute function
message_length(message)
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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

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

```
# You are using Python
# Recursive function to compress a string
def compress_string(s, index=0, count=1, result=""):
    if index >= len(s) - 1:  # Base case: Reached the end of the string
        result += s[index] + (str(count) if count > 1 else "")
        return result

    if s[index] == s[index + 1]:  # If current and next characters are the same, increment count
        return compress_string(s, index + 1, count + 1, result)
    else:  # If they differ, append the current character and count (if >1), then reset count
        result += s[index] + (str(count) if count > 1 else "")
        return compress_string(s, index + 1, 1, result)

# Read input
input_string = input().strip()

# Execute function and print result
print(compress_string(input_string))
```

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

2. 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))  # Convert number to string and sum its digits


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

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

3. 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

```
# Read temperature values from the two sensors
temp1 = float(input())
temp2 = float(input())
def difftemp(temp1,temp2):
    tempdiff=abs(temp1-temp2)
    print(f"Temperature difference:{tempdiff:.2f} °C")
difftemp(temp1,temp2)
```

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

4. 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

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

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

div_by_3 = list(filter(lambda x: x % 3 == 0, numbers))
div_by_5 = list(filter(lambda x: x % 5 == 0, numbers))

print(len(div_by_3))  # Count of numbers divisible by 3
print(len(div_by_5))  # Count of numbers divisible by 5
print(sum(div_by_3))  # Sum of numbers divisible by 3
print(sum(div_by_5))  # Sum of numbers divisible by 5
```

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

5. 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 = 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())

# You are using Python
def calculate_bmi(weight,height):
    bmi=weight/(height*height)
    print(f"Your BMI is:{bmi:.2f}")


calculate_bmi(weight, height)
```

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

6. 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:  # Continue until num becomes a single-digit number
        num = sum(int(digit) for digit in str(num))  # Sum the digits
    return num

# Read input number


print(digital_root(num))
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 38.5

Section 1 : Coding

1. 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

```
# You are using Python
# Read input prices as a space-separated string and convert to a list of integers
prices = list(map(int, input().split()))

# Filter even prices
even_prices = [price for price in prices if price % 2 == 0]

# Determine and print the result
if even_prices:
    print(f"The maximum even price is: {max(even_prices)}")
else:
    print("No even prices were found")
```

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

2. Problem Statement

Develop a text analysis tool that needs to count the occurrences of a specific substring within a given text string.

Write a function count_substrings(text, substring) that takes two inputs: the text string and the substring to be counted. The function should count how many times the substring appears in the text string and return the count.

Function Signature: count_substrings(text, substring)

Input Format

The first line of the input consists of a string representing the text.

The second line consists of a string representing the substring.

Output Format

The output should display a single line of output containing the count of occurrences of the substring in the text string.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: programming is fun and programming is cool
programming
Output: The substring 'programming' appears 2 times in the text.

Answer

```
def count_substrings(text, substring):
    return text.count(substring)  # Count occurrences of substring in text

# Read input text
text = input().strip()

# Read input substring
substring = input().strip()

# Get the count of occurrences
count = count_substrings(text, substring)

# Print the formatted output
print(f"The substring '{substring}' appears {count} times in the text.")
```

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

3. Problem Statement

You are tasked with designing a shipping cost calculator program that calculates the shipping cost for packages based on their weight and destination. The program utilizes different shipping rates for domestic, international, and remote destinations. The rates for each destination type are provided as global constants.

Constant Values:

DOMESTIC_RATE = 5.0

INTERNATIONAL_RATE = 10.0

REMOTE_RATE = 15.0

Function Signature: calculate_shipping(weight, destination)

Formula: shipping cost = weight * destination rate

Input Format

The first line of the input consists of a float representing the weight of the package.

The second line consists of a string representing the destinations(Domestic or International or Remote).

Output Format

The program outputs any one of the following:

1. If the input is valid and the destination is recognized, the output should consist of a single line stating the calculated shipping cost for the given weight and destination in the format: "Shipping cost to [destination] for a [weight] kg package: $[calculated cost]" with two decimal places.
2. If the input weight is not a positive float, print "Invalid weight. Weight must be greater than 0."
3. If the input destination is not one of the valid options, print "Invalid destination."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5.5
Domestic
Output: Shipping cost to Domestic for a 5.5 kg package: $27.50

Answer

#

```
# Define global constants for shipping rates
DOMESTIC_RATE = 5.0
INTERNATIONAL_RATE = 10.0
REMOTE_RATE = 15.0

def calculate_shipping(weight, destination):
    # Validate weight
    if weight <= 0:
        return "Invalid weight. Weight must be greater than 0."

    # Define destination rates
    rates = {
        "Domestic": DOMESTIC_RATE,
        "International": INTERNATIONAL_RATE,
        "Remote": REMOTE_RATE
    }

    # Validate destination
    if destination not in rates:
        return "Invalid destination."

    # Calculate shipping cost
    cost = weight * rates[destination]

    # Return cost as a numeric value
    return cost

# Read input
weight = float(input().strip())
destination = input().strip()
```

```
# Get shipping cost
shipping_cost = calculate_shipping(weight, destination)

if shipping_cost is not None:
    print(f"Shipping cost to {destination} for a {weight} kg package:
${shipping_cost:.2f}")
```

Status : Partially correct **Marks : 8.5/10**

4. 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 ≥ 6 and at least 2 different character types, the output prints "<password> is Moderate"

If Password length ≥ 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

```
# You are using Python
import string

def check_password_strength(password):
    # Check for different character types
    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)

    # Count character type diversity
    character_types = sum([has_lower, has_upper, has_digit, has_special])

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

    # Print formatted output
    print(f"{password} is {strength}")

# Read input password
password = input().strip()

# Check password strength
check_password_strength(password)
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 17

Section 1 : MCQ

1. 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***

2. 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***

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

Answer

( )

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

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

Answer

Keys must be integers

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

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

Answer

False

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

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

Answer

Values of a dictionary must be unique

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

7. What 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***

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

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

Answer

True

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

9. 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

None of the mentioned options

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

10. What will be the output?

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

Answer

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

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

11. Which of the following is a Python tuple?

Answer

(1, 2, 3)

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

12. 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***

13. 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***

14. Predict the output of the following Python program

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

Answer

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

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

15. 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***

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

Answer

t[3] = 45

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

17. 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***

18. 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***

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

Answer

Removes an arbitrary element

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

20. 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***

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 3
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

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

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

Note:

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

Input Format

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

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

```
# You are using Python
def filter_inventory(n, items, threshold):
    filtered_quantities = []

    for item in items:
        item_id, quantities = item
        # Use filter() to find quantities greater than the threshold
        filtered = list(filter(lambda x: x > threshold, quantities))
        filtered_quantities.extend(filtered)

    return filtered_quantities

# Read inputs
N = int(input())
items = []
```

```
for _ in range(N):
    item = eval(input().strip())  # Safely evaluate the tuple input
    items.append(item)

threshold = int(input().strip())

# Get filtered quantities
result = filter_inventory(N, items, threshold)

# Print result space-separated
print(" ".join(map(str, 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

```
# You are using Python
def analyze_product_ids():
    # Read the number of product IDs
    n = int(input())

    # Initialize a dictionary to track frequency of each product ID
    frequency = {}

    # Read each product ID and count its frequency
    for _ in range(n):
        product_id = int(input())
        if product_id in frequency:
            frequency[product_id] += 1
        else:
            frequency[product_id] = 1

    # Calculate total unique product IDs
```

```
    total_unique_ids = len(frequency)

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

    # Prepare the outputs
    print(frequency)
    print(f"Total Unique IDs: {total_unique_ids}")
    print(f"Average Frequency: {average_frequency:.2f}")

# Call the function to run the program
analyze_product_ids()
```

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

3. Problem Statement

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

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

Input Format

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

```
# You are using Python
def analyze_sales_data():
    # Read the number of customers
    n = int(input())

    # Initialize the dictionary to hold customer data
    customer_data = {}

    for _ in range(n):
        # Read each customer line and split into a list of integers
        data = list(map(int, input().split()))
        customer_id = data[0]
        amounts = data[1:]

        # Calculate total expenditure and maximum single expenditure
        total_expenditure = sum(amounts)
        max_expenditure = max(amounts)

        # Store results in the dictionary
        customer_data[customer_id] = [total_expenditure, max_expenditure]

    # Output the result
    print(customer_data)

# Call the function to execute the program
analyze_sales_data()
```

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

4. 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
# Read input sets
registered = set(map(int, input().split()))
attended = set(map(int, input().split()))
dropped_out = set(map(int, input().split()))

# Compute students who registered & attended
registered_attended = registered & attended

# Compute students who attended and did not drop out
final_students = registered_attended - dropped_out

# Print outputs
print(registered_attended)
print(final_students)
```

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

5. Problem Statement

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

Write a program to help Gowshik with this task.

Example:

Given list:

[1, 2, 3, 4]

[3, 5, 2, 1]

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

Input Format

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

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

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

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

```
# You are using Python
# Read the length of the lists
n = int(input().strip())

# Read the two lists of integers
list1 = list(map(int, input().split(",")))
list2 = list(map(int, input().split(",")))

# Compute the element-wise sum and format it as a tuple
result_tuple = tuple(x + y for x, y in zip(list1, list2))

# Print the output
print(result_tuple)
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 5_PAH

Attempt : 1
Total Mark : 60
Marks Obtained : 60

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

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

Answer

```
# Function to generate the dictionary
def generate_square_dict(n):
    return {i: i**2 for i in range(1, n+1)}

# Read input
n = int(input())

# Generate and print the dictionary
print(generate_square_dict(n))
```

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

2. 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

```
# You are using Python
def group_consecutive(ids):
    ids.sort()  # Ensure IDs are sorted
    grouped = []
    temp_group = [ids[0]]

    for i in range(1, len(ids)):
        if ids[i] == ids[i - 1] + 1:
            temp_group.append(ids[i])
        else:
            grouped.append(tuple(temp_group))
            temp_group = [ids[i]]

    grouped.append(tuple(temp_group))  # Append the last group

    return grouped

# Read input
n = int(input())
event_ids = [int(input()) for _ in range(n)]

# Get consecutive groups
result = group_consecutive(event_ids)
```

```
# Print the output, ensuring single-element tuples display without a trailing
comma
for group in result:
    if len(group) == 1:
        print(f"({group[0]})")
    else:
        print(group)
```

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

3. Problem Statement

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

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

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

Input Format

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

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

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

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

Output Format

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

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

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

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

For invalid choices: Print "Invalid choice".

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

```
# You are using Python
def manipulate_set():
    # Read the initial set of integers and convert them into a list of integers
    initial_set = list(map(int, input().strip().split()))

    # Create a set from the initial list to avoid duplicates
    number_set = set(initial_set)

    # Display the original set in descending order
    sorted_set = sorted(number_set, reverse=True)
    print(f'{{{", ".join(map(str, sorted_set))}}}')

    # Read the choice of operation
    choice = int(input().strip())

    if choice == 1:  # Find maximum value
        print(max(number_set))

    elif choice == 2:  # Find minimum value
        print(min(number_set))

    elif choice == 3:  # Remove a specific number
```

```
        number_to_remove = int(input().strip())
        if number_to_remove in number_set:
            number_set.remove(number_to_remove)
        # Display the updated set in descending order
        updated_sorted_set = sorted(number_set, reverse=True)
        print(f'{{{", ".join(map(str, updated_sorted_set))}}}')

    else:  # Invalid choice
        print("Invalid choice")

# Run the function
manipulate_set()
```

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

4. 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 process_integers(input_string):
    # Split the input string into a list of integers
    integers = input_string.split()

    # Use an ordered set approach to remove duplicates while preserving order
    unique_integers = []
    seen = set()

    for number in integers:
        if number not in seen:
            seen.add(number)
            unique_integers.append(number)

    # Concatenate the unique integers into a single string
    result = ''.join(unique_integers)

    # Print the result
    print(result)

# Example usage
if __name__ == "__main__":
    # Input can be taken from standard input or hardcoded for testing
    input_string = input()
    process_integers(input_string)
```

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

def generate_primes(n):
    primes = []
    current_num = 2  # The first prime number
    while len(primes) < n:
        if is_prime(current_num):
            primes.append(current_num)
        current_num += 1

    prime_dict = {i + 1: primes[i] for i in range(n)}  # Creating the dictionary
    return prime_dict

# Input from the user
n = int(input())
output_dict = generate_primes(n)
print(output_dict)
```

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

6. Problem Statement

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

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

Input Format

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
5 10 15
Output: ((5, 10), (10, 15), (15, None))

Answer

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

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

# Initialize an empty list to store the pairs
pairs = []

# Forming the pairs of consecutive integers
for i in range(n - 1):
    pairs.append((elements[i], elements[i + 1]))

# Add the last pair with None
pairs.append((elements[-1], None))

# Convert list of pairs to tuple format and print
output = tuple(pairs)
print(output)
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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

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

```
def find_matching_bases(seq1, seq2):
    # Compare elements at the same positions
    matches = [seq1[i] for i in range(min(len(seq1), len(seq2))) if seq1[i] == seq2[i]]
    return matches

# Read input
n = int(input())  # Size of first tuple
seq1 = tuple(map(int, input().split()))  # First DNA sequence

m = int(input())  # Size of second tuple
seq2 = tuple(map(int, input().split()))  # Second DNA sequence

# Find matching bases
matching_bases = find_matching_bases(seq1, seq2)

# Print output as space-separated integers
print(" ".join(map(str, matching_bases)))
```

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

2. Problem Statement

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

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

Input Format

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

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

Output Format

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

Refer to the sample output for format specifications.

Sample Test Case

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

Answer

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

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

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

    # Print the result
    print(differences)

# Call the function to execute the program
calculate_pixel_differences()
```

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

3. Problem Statement

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

Help Riya to implement the above task using a dictionary.

Input Format

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

The next n1 lines contain two integers

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

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

The next n2 lines contain two integers

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

Output Format

The output should display a dictionary that includes:

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
4
4
1
8
7

Output: {4: 4, 8: 7}

Answer

```
# You are using Python
def main():
    # Input the first dictionary
    n1 = int(input())
    dict1 = {}
    for _ in range(n1):
        key = int(input())
        value = int(input())
        dict1[key] = value

    # Input the second dictionary
    n2 = int(input())
    dict2 = {}
    for _ in range(n2):
        key = int(input())
        value = int(input())
```

```
        dict2[key] = value

    # Output dictionary to hold the results
    output_dict = {}

    # Compare both dictionaries
    for key in dict1:
        if key in dict2:
            # Calculate absolute difference
            output_dict[key] = abs(dict1[key] - dict2[key])
        else:
            # Key is unique to dict1
            output_dict[key] = dict1[key]

    for key in dict2:
        if key not in dict1:
            # Key is unique to dict2
            output_dict[key] = dict2[key]

    # Print the output dictionary
    print(output_dict)

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

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

4. Problem Statement

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

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

Input Format

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

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

Output Format

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

```
def calculate_symmetric_difference(club_lists):
    # Start with the first club's set
    symmetric_diff = set(club_lists[0])

    # Iterate through the remaining club lists, updating the symmetric difference
    for club in club_lists[1:]:
        symmetric_diff ^= set(club)  # Symmetric difference update

    return symmetric_diff, sum(symmetric_diff)

# Read input
k = int(input())  # Number of clubs
club_lists = [list(map(int, input().split())) for _ in range(k)]

# Compute symmetric difference and sum
symmetric_difference, total_sum = calculate_symmetric_difference(club_lists)

# Print outputs
```

```
print(symmetric_difference)
print(total_sum)
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 19

Section 1 : MCQ

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

Answer

finally

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

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

Answer

error

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

3. How do you rename a file?

Answer

os.rename(existing_name, new_name)

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

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

Answer

To handle exceptions during code execution

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

5. What is the output of the following code?

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

Answer

Caught division by zero errorExecuted

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

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

file_object.seek(offset [,reference_point])

Answer

0

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

7. 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***

8. 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***

9. Which of the following is true about

fp.seek(10,1)

Answer

Move file pointer ten characters ahead from the current position

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

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

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

```
    "This is 4th line",
    "This is 5th line"
]

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

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

Answer

Displays Output

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

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

```
import _____ (1)
rec=[]
while True:
    rn=int(input("Enter"))
    nm=input("Enter")
    temp=[rn, nm]
    rec.append(temp)
    ch=input("Enter choice (y/N)")
    if ch.upper=="N":
        break
f.open("stud.dat","_____")(2)
_____.dump(rec,f)(3)
_____.close()(4)
```

Answer

(pickle,wb,pickle,f)

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

12. 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***

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

Answer

raise Exception()

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

14. 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***

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

Output:

Name of the file: ex.txt

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

Answer

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

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

16. 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***

17. 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***

18. 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***

19. 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***

20. 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***

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 2
Total Mark : 50
Marks Obtained : 48.5

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
def calculate_total_cost(price, quantity):
    try:
        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")

        return round(total_cost, 1)

    except ValueError as ve:
        return str(ve)
    except RuntimeError as re:
        return str(re)
```

```
# Input reading
try:
    price = float(input())
    quantity = int(input())

    result = calculate_total_cost(price, quantity)
    print(result)

except ValueError:
    print("Invalid input format.")
```

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

2. Problem Statement

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

Input Format

The input contains a positive integer representing age.

Output Format

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

Otherwise, the output displays "Eligible to vote".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 18
Output: Eligible to vote

Answer

```
# You are using Python
class NotEligibleToVoteException(Exception):
```

```
    """Custom exception for voting eligibility."""
    pass

def check_voting_eligibility(age):
    if age < 18:
        raise NotEligibleToVoteException("Not eligible to vote")
    else:
        return "Eligible to vote"

def main():
    try:
        age = int(input())

        # Ensure age is within the given constraints
        if age < 1 or age > 100:
            print("Age must be between 1 and 100.")
            return

        result = check_voting_eligibility(age)
        print(result)

    except NotEligibleToVoteException as e:
        print(e)
    except ValueError:
        print("Please enter a valid integer.")

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

Status : Partially correct ***Marks : 8.5/10***

3. 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
def main():
    # Get the input string from the user
    user_input = input()

    # Save the input string to a file
    file_name = "text_file.txt"
    with open(file_name, "w") as file:
        file.write(user_input)

    # Read the string back from the file
    with open(file_name, "r") as file:
        original_string = file.read().strip()
```

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

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

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

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

4. Problem Statement

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

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

File Name: sentence_file.txt

Input Format

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

Output Format

The output displays the count of words in the sentence.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: Four Words In This Sentence
Output: 5

Answer

```
# Define the file name
filename = "sentence_file.txt"

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

# Save the input to the file
with open(filename, "w") as file:
    file.write(sentence)

# Read the sentence back from the file
with open(filename, "r") as file:
    content = file.read().strip()  # Remove leading/trailing spaces

# Count words by splitting the sentence on spaces
word_count = len(content.split()) if content else 0

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

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

5. Problem Statement

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

Input Format

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

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

Output Format

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

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

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

```
def get_integer_input(prompt):
    """Helper function to safely get integer input."""
    try:
        value = int(input(prompt))
        return value
    except ValueError:
        print("Error: You must enter a numeric value.")
        exit()

# Get the length of the list
n = get_integer_input("")

# Validate n
if n <= 0 or n > 20:
    print("Error: The length of the list must be a non-negative integer.")
    exit()
```

```
# Get list elements
numbers = []
for _ in range(n):
    try:
        num = int(input())
        numbers.append(num)
    except ValueError:
        print("Error: You must enter a numeric value.")
        exit()

# Calculate average and display output
average = sum(numbers) / n
print(f"The average is: {average:.2f}")
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 28.5

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

```
MAX_CAPACITY = 30  # Maximum allowed students

# Read the number of students to be added
try:
    n = int(input().strip())

    if n <= 0:
        print("Exception caught. Error: Invalid number of students.")
        exit()
```

```
except ValueError:
    print("Exception caught. Error: You must enter a numeric value.")
    exit()

database = {}  # Dictionary to store student records
student_count = 0  # Tracks number of successfully added students

# Process student entries
for _ in range(n):
    try:
        student_data = input().strip().split()
        student_id = int(student_data[0])
        student_name = " ".join(student_data[1:])

        if student_id in database:
            print("Exception caught. Error: Student ID already exists.")
            continue  # Skip duplicate entry

        if student_count >= MAX_CAPACITY:
            print("Exception caught. Error: Student database is full.")
            break  # Stop adding more students

        database[student_id] = student_name
        student_count += 1
        print(f"Student with ID {student_id} added to the database.")

    except ValueError:
        print("Exception caught. Error: You must enter a numeric value.")
        exit()
```

Status : Partially correct ***Marks : 8.5/10***

2. Problem Statement

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

File Name: user_input.txt

Input Format

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

These numbers may be integers or decimals.

Output Format

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

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

Refer to the sample output for format specifications.

Sample Test Case

Input: 1 2 3 4 5
Output: Average of the numbers is: 3.00

Answer

```
filename = "user_input.txt"

# Read user input and save it to the file
user_input = input().strip()

with open(filename, "w") as file:
    file.write(user_input)

# Read the numbers from the file and process them
try:
    with open(filename, "r") as file:
        data = file.read().strip()

    numbers = data.split()

    # Attempt to convert each value into a float
    try:
        num_list = [float(num) for num in numbers]
    except ValueError:
        print("Invalid data in the input.")
```

```
        exit()

    # Compute and print the average
    average = sum(num_list) / len(num_list)
    print(f"Average of the numbers is: {average:.2f}")

except FileNotFoundError:
    print("Error: File not found.")
```

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

3. Problem Statement

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

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

Input Format

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

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

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

Output Format

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: test.txt
This is a test file to check the character count.
e
Output: The character 'e' appears 5 times in the file.

Answer

```
# Read file name, string to write, and character to count
file_name = input().strip()
content = input().strip()
char_to_count = input()

# Ensure the character is a single character (including spaces)
if len(char_to_count) != 1:
    print("Error: Please enter a single character.")
    exit()

# Write content to the specified file
with open(file_name, "w") as file:
    file.write(content)

# Read content from the file
with open(file_name, "r") as file:
    file_content = file.read()

# Count occurrences of the specified character **case-insensitively**
count = file_content.lower().count(char_to_count.lower())

# Print the 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***

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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

Bob, a data analyst, requires a program to automate the process of analyzing character frequency in a given text. This program should allow the user to input a string, calculate the frequency of each character within the text, save these character frequencies to a file named "char_frequency.txt," and display the results.

Input Format

The input consists of the string.

Output Format

The first line prints "Character Frequencies:".

The following lines print the character frequency in the format: "X: Y" where X is the character and Y is the count.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: aaabbbccc

Output: Character Frequencies:
a: 3
b: 3
c: 3

Answer

```
from collections import Counter

# Read input string
input_string = input().strip()

# Compute character frequencies
char_freq = Counter(input_string)

# Define output filename
filename = "char_frequency.txt"

# Write frequencies to file
with open(filename, "w") as file:
    file.write("Character Frequencies:\n")
    for char, count in char_freq.items():
        file.write(f"{char}: {count}\n")

# Display output
print("Character Frequencies:")
for char, count in char_freq.items():
    print(f"{char}: {count}")
```

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

2. 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

```
def record_sales():
    import os

    # Step 1: Input data
    N = int(input())
    if N > 30:
        print("Exceeding limit!")
        return

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

    # Step 2: Calculate total earnings for each day
    total_earnings = [items * M for items in items_sold]

    # Step 3: Save earnings to file
    with open('sales.txt', 'w') as file:
        for earnings in total_earnings:
            file.write(f"{earnings}\n")

    # Step 4: Read from file and display earnings
    print()
    with open('sales.txt', 'r') as file:
        for line in file:
            print(line.strip())

# Run the function
record_sales()
```

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

3. Problem Statement

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

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

Input Format

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

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

Output Format

The output should display any one of the following messages:

If both numbers are valid, print "Valid".

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 19ABC1001
9949596920
Output: Valid

Answer

```
# You are using Python
import re

class RegisterNumberException(Exception):
    pass

class MobileNumberException(Exception):
```

```
    pass

def validate_register_number(register_number):
    if len(register_number) != 9:
        raise RegisterNumberException("Register Number should have exactly 9
characters.")

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

def validate_mobile_number(mobile_number):
    if len(mobile_number) != 10:
        raise MobileNumberException("Mobile Number should have exactly 10
characters.")

    if not mobile_number.isdigit():
        raise MobileNumberException("Mobile Number should only contain digits.")

def main():
    try:
        register_number = input()
        mobile_number = input()

        # Validate the inputs
        validate_register_number(register_number)
        validate_mobile_number(mobile_number)

        print("Valid")

    except (RegisterNumberException, MobileNumberException) as e:
        print(f"Invalid with exception message: {e}")

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

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

4. Problem Statement

Implement a program that checks whether a set of three input values can

form the sides of a valid triangle. The program defines a function is_valid_triangle that takes three side lengths as arguments and raises a ValueError if any side length is not a positive value. It then checks whether the sum of any two sides is greater than the third side to determine the validity of the triangle.

Input Format

The first line of input consists of an integer A, representing side1.

The second line of input consists of an integer B, representing side2.

The third line of input consists of an integer C, representing side3.

Output Format

The output prints either "It's a valid triangle" if the input side lengths form a valid triangle,

or "It's not a valid triangle" if they do not.

If there is a ValueError, it should print "ValueError: <error_message>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
4
5
Output: It's a valid triangle

Answer

```
# You are using Python
def is_valid_triangle(side1, side2, side3):
    # Check if all side lengths are positive
    if side1 <= 0 or side2 <= 0 or side3 <= 0:
        raise ValueError("Side lengths must be positive")

    # Check if the triangle inequality theorem holds
    if (side1 + side2 > side3) and (side1 + side3 > side2) and (side2 + side3 > 
side1):
```

```
        return True
    else:
        return False

# Input reading
try:
    A = int(input())
    B = int(input())
    C = int(input())

    # Check if it is a valid triangle
    if is_valid_triangle(A, B, C):
        print("It's a valid triangle")
    else:
        print("It's not a valid triangle")

except ValueError as e:
    print(f"ValueError: {e}")
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 19

Section 1 : MCQ

1. What does the np.arange(10) function in NumPy do?

Answer

Creates an array with values from 1 to 9

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

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

Answer

pd.DataFrame()

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

3. What is the primary purpose of Pandas DataFrame?

Answer

None of the mentioned options

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

4. 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***

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

Answer

arr[0]

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

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

Answer

numpy.std()

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

7. 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

[3 4]

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

8. The important data structure of pandas is/are ____.

Answer

Both Series and Data Frame

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

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

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

Answer

Value Error

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

10. 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***

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

Answer

import numpy as np

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

12. 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***

13. Which NumPy function is used to create an identity matrix?

Answer

numpy.identity()

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

14. 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***

15. What does NumPy stand for?

Answer

Numerical Python

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

16. 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***

17. 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***

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

Answer

1

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

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

Answer

Array

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

20. 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***

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 47

Section 1 : Coding

1. 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

```
import pandas as pd
import numpy as np
```

```
import sys

# Read input
n = int(input().strip())
header = input().strip().split(',')
data = [input().strip().split(',') for _ in range(n)]

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

# Convert Age to numeric (float), coercing errors to NaN
if 'Age' in df.columns:
    df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

    # Calculate the mean of the available Age values (excluding NaN)
    mean_age = df['Age'].mean()

    # Replace NaN values in Age with the mean age
    df['Age'].fillna(mean_age, inplace=True)

# Remove rows where Diagnosis is missing or empty
if 'Diagnosis' in df.columns:
    df = df[df['Diagnosis'].notna() & (df['Diagnosis'].str.strip() != '')]

# Reset the DataFrame index
df.reset_index(drop=True, inplace=True)

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

Status : Partially correct ***Marks : 8.5/10***

2. 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

```
import numpy as np

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

# Initialize an array to hold the sales data
sales_data = np.zeros((products, months), dtype=int)

# Read the sales data for each product
for i in range(products):
    sales_data[i] = np.array(list(map(int, input().split())))
```

```
# Calculate cumulative sales
cumulative_array = np.cumsum(sales_data, axis=1)

# Print the results
print("Cumulative Monthly Sales:")
print(cumulative_array)
```

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

3. 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

# Input the first time series
array1 = np.array(list(map(float, input().strip().split())))

# Input the second time series
array2 = np.array(list(map(float, input().strip().split())))

# Compute the cross-correlation
cross_corr = np.correlate(array1, array2, mode='full')

# Print the results
print("Cross-correlation of the two time series:")
print(cross_corr)
```

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

4. 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

# Input: number of sales entries
n = int(input())

# Input: sales values
sales = np.array(list(map(float, input().strip().split())))

# Filter sales values that are multiples of 5 and exceed 100
filtered_sales = sales[(sales % 5 == 0) & (sales > 100)]

# Output the filtered sales
print(filtered_sales)
```

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

5. Problem Statement

Sita works as a sales analyst and needs to analyze monthly sales data for

different cities. She receives lists of cities, months, and corresponding sales values and wants to create a pandas DataFrame using a MultiIndex of cities and months.

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

Input Format

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

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

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

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

Output Format

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

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

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

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

```
NYC  Jan    100.0
     Feb    200.0
LA   Jan    300.0
     Feb    400.0

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

Answer

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

# Input reading
n = int(input())
cities = input().strip().split()
months = input().strip().split()
sales = list(map(float, input().strip().split()))

# Creating the MultiIndex
index = pd.MultiIndex.from_tuples(zip(cities, months), names=["City", "Month"])
sales_data = pd.DataFrame(sales, index=index, columns=["Sales"])

# Display the DataFrame
print("Monthly Sales Data with MultiIndex:")
print(sales_data)

# Calculating total sales per city
total_sales_per_city = sales_data.groupby(level="City").sum()

# Display total sales per city
print("\nTotal Sales Per City:")
print(total_sales_per_city)
```

Status : Partially correct ***Marks : 8.5/10***

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 50

Section 1 : Coding

1. 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

# Read number of responses
n = int(input())

# Prepare lists to hold RespondentID and Feedback
respondent_ids = []
feedbacks = []

# Read each response
for _ in range(n):
    response = input().strip().split(maxsplit=1)
    respondent_id = response[0]

    # Check if feedback is provided
    if len(response) == 1 or response[1].strip() == "":
        feedback = "No Response"
    else:
        feedback = response[1]

    # Append to lists
    respondent_ids.append(respondent_id)
    feedbacks.append(feedback)

# Create a DataFrame
survey_data = pd.DataFrame({
    'RespondentID': respondent_ids,
    'Feedback': feedbacks
})

# Print the output
```

```
print("Survey Responses with Missing Feedback Filled:")
print(survey_data.to_string(index=False))
```

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

2. 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 ≤ 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

# Read the number of employees
n = int(input().strip())

# Read employee data
data = [input().split() for _ in range(n)]

# Convert data to DataFrame
df = pd.DataFrame(data, columns=["Name", "Years at Company"])

# Convert "Years at Company" to float
df["Years at Company"] = df["Years at Company"].astype(float)

# Define the experience level classification
def classify_experience(years):
    if years < 3:
        return "Junior"
    elif 3 <= years < 6:  # Fixed condition to ensure 6 is classified as Senior
        return "Mid"
    else:
        return "Senior"

# Apply classification function
df["Experience Level"] = df["Years at Company"].apply(classify_experience)
```

```
# Print formatted output
print("Employee Data with Experience Level:")
print(df.to_string(index=False))
```

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

3. 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

```
import numpy as np

# Read input
n = int(input().strip())
wait_times = list(map(float, input().split()))

# Determine bucket boundaries (up to the highest wait time rounded to nearest 5)
max_time = int(np.ceil(max(wait_times) / 5) * 5)
```

```
bins = np.arange(0, max_time + 5, 5)

# Use NumPy digitize to classify wait times into buckets
bucket_indices = np.digitize(wait_times, bins, right=False)

# Count occurrences in each bucket
bucket_counts = {f"{bins[i]}-{bins[i+1]}": 0 for i in range(len(bins) - 1)}
for idx in bucket_indices:
    if idx <= len(bins) - 1:  # Ensure index is within range
        bucket_counts[f"{bins[idx-1]}-{bins[idx]}"] += 1

# Print formatted output
print("Wait Time Buckets and Counts:")
for bucket, count in bucket_counts.items():
    print(f"{bucket}: {count}")
```

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

4. 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
rows, cols = map(int, input().strip().split())
data = []

for _ in range(rows):
    row = list(map(float, input().strip().split()))
    data.append(row)

# Convert the list to a NumPy array
returns = np.array(data)

# Identify days with all positive returns
positive_days = np.all(returns > 0, axis=1)

# Print the output
print("Days where all asset returns were positive:")
print(positive_days)
```

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

5. 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:

normalized_pixel = (pixel - min_pixel) / (max_pixel - 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

def normalize_image(rows, cols, pixel_values):
    # Convert the input pixel values into a numpy array
    image_matrix = np.array(pixel_values)

    # Find the minimum and maximum pixel values
    min_pixel = np.min(image_matrix)
    max_pixel = np.max(image_matrix)

    # Check for the case where all pixel values are the same
    if min_pixel == max_pixel:
        normalized_image = np.zeros_like(image_matrix, dtype=float)
    else:
        # Normalize the pixel values
        normalized_image = (image_matrix - min_pixel) / (max_pixel - min_pixel)

    return normalized_image

# Example usage:
if __name__ == "__main__":
    # Input reading
    rows = int(input())
    cols = int(input())
    pixel_values = [list(map(int, input().split())) for _ in range(rows)]

    # Normalize the image
    normalized_image = normalize_image(rows, cols, pixel_values)

    # Output the result with specific formatting
    print(normalized_image)
```

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

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
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 : 50

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

# Input: Read 5 lines of rainfall data
data = []
for _ in range(5):
    line = input().strip()
    monthly_rainfall = list(map(float, line.split()))
    data.append(monthly_rainfall)

# Convert data to a numpy array
rainfall_array = np.array(data)

# Calculate yearly total rainfall (sum of each row)
yearly_totals = np.sum(rainfall_array, axis=1)

# Find the month with max rainfall for each year (index of max in each row)
```

```
max_rainfall_months = np.argmax(rainfall_array, axis=1)

# Output results
print(yearly_totals)
print(max_rainfall_months)
```

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

2. 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 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

def normalize_sensor_data(sensors, samples, readings):
    sensor_data = np.array(readings)

    # Normalize each sensor's readings
    normalized_data = (sensor_data - sensor_data.mean(axis=1,
keepdims=True)) / sensor_data.std(axis=1, keepdims=True)

    return normalized_data

# Input reading
sensors, samples = map(int, input().split())
readings = [list(map(float, input().split())) for _ in range(sensors)]

# Normalize the sensor data
normalized_data = normalize_sensor_data(sensors, samples, readings)

# Output the result
print("Normalized Sensor Data:")
print(normalized_data)
```

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

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
import sys

# Read input
n = int(input().strip())
header = input().strip().split(',')
data = [input().strip().split(',') for _ in range(n)]

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

# Convert 'Purchase Date' to datetime format
df['Purchase Date'] = pd.to_datetime(df['Purchase Date'], errors='coerce')

# Create new columns for Month and Day
df['Purchase Month'] = df['Purchase Date'].dt.month
df['Purchase Day'] = df['Purchase Date'].dt.day

# Output
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 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0

Output: [30.]

Answer

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

# Read the number of temperature readings
n = int(input().strip())

# Read the hourly temperature readings
temperature_readings = list(map(float, input().strip().split()))
```

```
# Reshape the temperature readings into rows of 24 readings each
temperature_array = np.array(temperature_readings).reshape(-1, 24)

# Calculate the average temperature for each day
avg_per_day = np.mean(temperature_array, axis=1)

# Print the result
print(avg_per_day)
```

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

5. 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

# Read the number of products
n = int(input().strip())

# Initialize an empty list to store product data
products = []

# Read the product data
for _ in range(n):
    line = input().strip()
    product_id, quantity = line.split()
    quantity = int(quantity)
    products.append((product_id, quantity))

# Create a DataFrame
df = pd.DataFrame(products, columns=['Product_ID', 'Quantity'])

# Identify products with zero stock
zero_stock_products = df[df['Quantity'] == 0]

# Output header
print("Products with Zero Stock:")

# Check if there are any products with zero stock
if zero_stock_products.empty:
    print("No products with zero stock.")
else:
    # Display the products with zero stock
    print(zero_stock_products.to_string(index=False))
```

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