# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 2_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 13

## Section 1 : MCQ

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

```
i = 1
while False:
    if i%2 == 0:
        break
    print(i)
    i += 2
```

*Answer*

The code runs successfully but does not print anything

*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**

none of the mentioned

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


3.  How many times will the inner for loop be executed in the below code?

```
i=0
while(True):
  for j in range(4,0,-2):
    print(i*j)
    print('')
    i=i+1
  if(i%2==0):
    break
```

**Answer**

02

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


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

```
i = 5
while True:
    if i%0O11 == 0:
        break
    print(i)
    i += 1
```

**Answer**

5 6 7 8

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

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

*Answer*

012

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

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

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

*Answer*

1 3 5 7 9

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

7.   What is the output of the following code?

i = 5

```
while True:
    if i%0O9 == 0:
        break
    print(i)
    i += 1
```

**Answer**

Compile Time Error

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

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

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

**Answer**

1 3

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

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

**Answer**

break

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

10.  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*

11.   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**

5 6 7 8

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

12.   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*

13.   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*


14.  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 :* Wrong                                                                                      *Marks : 0/1*

15.  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*

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

## 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

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*

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

def is_perfect(n):
    if n<2:
        return False
    sum_divisors = 1
    for i in range(2,n):
        if n % i == 0:
            sum_divisors += i
    return sum_divisors == n

perfect_numbers = []
for num in range (start, end + 1):
    if is_perfect(num):
        perfect_numbers.append(str(num))
print(" ".join(perfect_numbers))
```

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

2.  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*

```python
# You are using Python
def print_consonants(sentence):
    vowels = "AEIOUaeiou"
    consonants = []
    for char in sentence:
        if char.isalpha() and char not in vowels:
            consonants.append(char)
        elif char == " ":
            consonants.append (" ")
    print(" ".join(consonants))
sentence = input()
print_consonants(sentence)
```

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

3. Problem Statement

John, a software developer, is analyzing a sequence of numbers within a given range to calculate their digit sum. However, to simplify his task, he

excludes all numbers that are palindromes (numbers that read the same backward as forward).

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

Example:

Input:

10

20

Output:

55

Explanation:

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

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

Output: 55

### Input Format

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

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

### Output Format

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

Refer to the sample output for formatting specifications.

*Sample Test Case*

Input: 10
20

Output: 55

*Answer*

```python
# You are using Python
def is_palindrome(n):
    return str(n) == str(n)[::-1]

def sum_of_digits(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 += sum_of_digits(num)
    return total_sum

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

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

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

4.  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**

```python
# You are using Python
def generate_fibonacci_up_to(limit):
    fib_nums = set()
    a, b = 0, 1
    while a <= limit:
        fib_nums.add(a)
        a, b = b, a + b
    return fib_nums

def count_non_fibonacci(start, end):
    fibonacci_numbers = generate_fibonacci_up_to(100)

    non_fibonacci_count = 0

    for num in range(start, end + 1):
        if num in fibonacci_numbers:
            continue
        non_fibonacci_count += 1

    return non_fibonacci_count

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

```
result = count_non_fibonacci(start, end)
print(result)
```

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


5.  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*

```python
# You are using Python
def print_even_squares(n):
    for i in range(1, n + 1):
        if i % 2 != 0:
            continue
```

```
    print(i * i)

n = int(input())

print_even_squares(n)
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

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*

```python
# You are using Python
def is_prime(x):
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True

def find_twin_primes(n):
    twin_primes = []
    num = 2

    while len(twin_primes) < n:
        if is_prime(num) and is_prime(num + 2):
            twin_primes.append((num, num + 2))
        num +=1

    return twin_primes

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

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

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

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

## 2. Problem Statement

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

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

*Input Format*

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

*Output Format*

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: 5
Output: 2 3 5 13 89

*Answer*

```python
# You are using Python
def is_prime(x):
    if x <= 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
```

```
        return False
    return True

def main():
    n = int(input())
    a, b = 0, 1
    prime_fibonacci_numbers = []

    while len(prime_fibonacci_numbers) < n:
        if is_prime(a):
            prime_fibonacci_numbers.append(a)
        a, b = b, a + b

    print(" ".join(map(str, prime_fibonacci_numbers)))

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

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

3.  Problem Statement

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

Help Alex to complete his task.

Example:

Input:

845

output:

12

Explanation:

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

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

Output: 12

*Input Format*

The input consists of a single integer X.

*Output Format*

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

Refer to the sample output for formatting specifications.

*Sample Test Case*

Input: 845
Output: 12

*Answer*

```python
# You are using Python
def sum_non_prime_digits(x):
    non_prime_digits = {'0', '1', '4', '6', '8', '9'}
    total_sum = 0

    for digit in str(x):
        if digit in non_prime_digits:
            total_sum += int(digit)

    return total_sum

def main():
    x=int(input())
    print(sum_non_prime_digits(x))

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

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

## 4. Problem Statement

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

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

In the event that an invalid mode is provided, the program should output an error message indicating the invalid selection.

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

### Input Format

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

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

### Output Format

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

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

Refer to the sample output for formatting specifications.

### Sample Test Case

Input: 1
10.0
Output: Intensity: 15.00

### Answer

```python
# You are using Python
def adjust_intensity(mode, intensity):
    if mode == 1:
        return intensity * 1.5

    elif mode == 2:
        return intensity * 2.0
    elif mode == 3:
        return intensity * 1.8
    else:
        return None

def main():
    n=int(input())
    m=float(input())
    result = adjust_intensity(n, m)
    if result is not None:
        print(f"Intensity: {result:.2f}")
    else:
        print("Invalid")
if __name__ == "__main__":
    main()
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

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***

```python
# You are using Python
import math

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

def factorial_workshop(n):
    for i in range(1, n + 1, 2):
        fact = math.factorial(i)
        digit_sum = sum_of_digits(fact)
        print(f"{i}! = {fact}, sum of digits = {digit_sum}")

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

***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*

```python
# You are using Python
def sum_excluding_multiples_of_3(start, end):
    total = 0
    for num in range(start, end + 1):
        if num % 3 == 0:
            continue
        total += num
    print(total)

start = int(input())
end = int(input())
sum_excluding_multiples_of_3(start, end)
```

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


3.  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

```python
# You are using Python
def sum_even_odd(n):
    even_sum = sum(num for num in range(1, n + 1) if num % 2 == 0)
    odd_sum = sum(num for num in range(1, n+ 1) if num %2!=0)

    print(f"Sum of even numbers from 1 to {n} is {even_sum}")
    print(f"Sum of odd numbers from 1 to {n} is {odd_sum}")

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

4. Problem Statement

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

*Input Format*

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

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

*Output Format*

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

Refer to the sample output for formatting specifications.

*Sample Test Case*

Input: 2000
2053
Output: 2000
2004
2008
2012
2016
2020
2024
2028
2032
2036
2040
2044
2048
2052

*Answer*

```python
# You are using Python
def print_leap_years(start_year, end_year):
    for year in range(start_year, end_year + 1):
        if(year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
            print(year)

start_year = int(input())
end_year = int(input())
print_leap_years(start_year, end_year)
```

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

## 5. 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*

```python
# You are using Python
def collatz_steps(n):
    steps = 0
    while n!= 1:
        if steps > 100:
            print("Exceeded 100 steps. Exiting...")
            break
        if n % 2 == 0:
            n //= 2
        else:
            n = 3 * n + 1

        steps += 1
    else:
        print(f"Steps taken to reach 1: {steps}")

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

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


6.  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*

```python
# You are using Python
def calculate_bill(units):
    if units <= 100:
        amount = 0
    elif units <= 200:
        amount = (units - 100) * 5
    else:
        amount = (100 * 5) + ((units - 200) * 10)

    print(f"Rs. {amount}")

units = int(input())
calculate_bill(units)
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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 Python code?

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

*Answer*

3

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

2.  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*

3.  What is the output of the following code?

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

**Answer**

9

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

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

**Answer**

The first three elements of the list

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

5.  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*

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

*Answer*

extend()

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

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

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

*Answer*

99

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

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

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

*Answer*

oje

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

9. 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*

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

*Answer*

print(list1[:-2])

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

11. 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*

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

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

*Answer*

nohtyP

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

13. 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*

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

*Answer*

list()

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


15.  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*


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

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

*Answer*

Python

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


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

*Answer*

Adds a new element to the end of the list

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


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

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

print(result)

*Answer*

**Python**

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

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

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

*Answer*

Hello World

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

20.  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*

21.  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*

22. 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*

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

**Answer**

[2, 33, 222, 14]

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

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

**Answer**

25

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

# Rajalakshmi Engineering College

Name: Karthik Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

### REC_Python_Week 3_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

## Section 1 : Coding

1. Problem Statement

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

Write a Python program for the same.

Note

(XXX) - Area code

XXX-XXXX - Phone number

*Input Format*

The input consists of a string, representing the phone number in the format

"(XXX) XXX-XXXX".

*Output Format*

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

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

*Answer*

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

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

2. Problem Statement

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

Ram needs your help to design a program.

*Input Format*

The input consists of two strings in separate lines.

*Output Format*

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: hello
word

Output: hellodrow

*Answer*

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

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


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*

```python
# You are using Python
numbers = eval(input())
negative_numbers = [num for num in numbers if num < 0]
non_negative_numbers = [num for num in numbers if num >= 0]
result = negative_numbers + non_negative_numbers
print("List =", result)
```

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

4.   Problem Statement

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

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

*Input Format*

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

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

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

*Output Format*

The output displays the following format:

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

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

Refer to the sample output for formatting specifications.

*Sample Test Case*

Input: pythonprogramming
0
5
Output: python

*Answer*

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

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

5. Problem Statement

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

Your task is to create a program that helps Alex manage the list. The

program should allow Alex to input a list of elements, append them to the existing list, and then remove an element at a specified index.

### Input Format

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

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

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

### Output Format

The first line of output displays the original list.

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

The third line of output displays the popped element.

Refer to the sample output for the formatting specifications.

### Sample Test Case

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

### Answer

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

```python
my_list = []
for _ in range(n):
    my_list.append(int(input()))
M =int(input())
print(f"List after appending elements: {my_list}")
popped_element = my_list.pop(M)
print(f"List after popping last element: {my_list}")
print(f"Popped element: {popped_element}")
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

### REC_Python_Week 3_PAH

Attempt : 1
Total Mark : 60
Marks Obtained : 57.5

## Section 1 : Coding

1.   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 string
str1 = input()
str2 = input()
translator = str.maketrans('', '', string.punctuation)
words = str1.translate(translator).split()
count = sum(1 for word in words if word == str2)
print(count)
```

*Status :* Partially correct                                    *Marks : 7.5/10*

2.  Problem Statement

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

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

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

Example

Input

1 3 2 4 1 5 7 6 10 2 8

Output

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

Explanation

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

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

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

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

8 is a peak because it is 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

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

```
    (i == len(nums) - 1 and nums[i] > nums[i - 1]) or
    (0 < i < len(nums) - 1 and nums[i] > nums[i - 1] and nums[i] > nums[i + 1])]
print("Peaks:", peaks)
```

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

## 3.   Problem Statement

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

### Input Format

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

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

### Output Format

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

Refer to the sample output for the formatting specifications.

### Sample Test Case

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

### Answer

```
n = int(input())
arr = list(map(int, input().split()))
arr_set = set(arr)
missing = 1
while missing in arr_set:
    missing += 1
print(missing)
```

## 4.   Problem Statement

Imagine you are developing a text analysis tool for a cybersecurity company. Your task is to analyze input strings to categorize and count the characters into four categories: uppercase letters, lowercase letters, digits, and special characters. The company needs this tool to process log files and identify potential security threats.

*Input Format*

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

*Output Format*

The output consists of four lines:

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

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

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

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: Hello123

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

*Answer*

```
log_entry = input()
uppercase_count = sum(1 for ch in log_entry if ch.isupper())
lowercase_count = sum(1 for ch in log_entry if ch.islower())
digit_count = sum(1 for ch in log_entry if ch.isdigit())
special_count = sum(1 for ch in log_entry if not ch.isalnum())
print(f"Uppercase letters: {uppercase_count}")
print(f"Lowercase letters: {lowercase_count}")
print(f"Digits: {digit_count}")
print(f"Special characters: {special_count}")
```

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

5.   Problem Statement

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

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

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

*Input Format*

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

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

*Output Format*

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

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: 5
5
1
2
3
4

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

*Answer*

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

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


6.   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*

```python
# You are using Python
def process_string():
    A = input().strip()
    length_of_A = len(A)
    B = A
    print(f"Length of the string: {length_of_A}")
    print(f"Copied string: {B}")
process_string()
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

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*

```
str1 = input()
str2 = input()
if len(str1) != len(str2):
    print("Error: Strings must be of equal length.")
else:
    result = ""
    for ch1, ch2 in zip(str1, str2):
        result += ch1 + ch2
    print(result)
```

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


2. Problem Statement

Gina is working on a data analysis task where she needs to extract sublists from a given list of integers and find the median of each sublist. For each median found, she also needs to determine its negative index in the original list.

Help Gina by writing a program that performs these tasks.

Note: The median is the middle value in the sorted list of numbers, or the first value of the two middle values if the list has an even number of elements.

Example

Input

10

1 2 3 4 5 7 8 9 10 11

3

1 5

2 6

3 10

Output

3 : -8

4 : -7

7 : -5

Explanation

For the first range (1 to 5), the sublist is [1, 2, 3, 4, 5]. The median is 3, and its negative index in the original list is -8.

For the second range (2 to 6), the sublist is [2, 3, 4, 5, 7]. The median is 4, and its negative index in the original list is -7.

For the third range (3 to 10), the sublist is [3, 4, 5, 7, 8, 9, 10, 11]. The median is 7, and its negative index in the original list is -5.

*Input Format*

The first line of input consists of an integer N, representing the number of elements in the list.

The second line consists of N space-separated integers representing the elements of the list.

The third line consists of an integer R, representing the number of ranges.

The next R lines each consist of two integers separated by space representing the start and end indices (1-based) of the ranges.

*Output Format*

The output consists of n lines, displaying "X : Y" where X is the median of the

sublist and Y is the negative index in the original list.

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: 10
1 2 3 4 5 7 8 9 10 11
3
1 5
2 6
3 10
Output: 3 : -8
4 : -7
7 : -5

*Answer*

```
n = int(input())
nums = list(map(int, input().split()))
r = int(input())
for _ in range(r):
    start, end = map(int, input().split())
    sublist = nums[start-1:end]
    sorted_sublist = sorted(sublist)
    length = len(sorted_sublist)
    if length % 2 == 1:
        median = sorted_sublist[length // 2]
    else:
        median = sorted_sublist[(length // 2) - 1]
    index_in_original = nums.index(median)
    negative_index = index_in_original - n
    print(f"{median} : {negative_index}")
```

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

3.  Problem Statement

A company is creating email accounts for its new employees. They want to

use a naming convention for email addresses that consists of the first letter of the employee's first name, followed by their last name, followed by @company.com.

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

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

Note:

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

*Input Format*

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

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

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

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

*Output Format*

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: John
Smith
admin

iamNeo
Output: jsmith@admin.iamneo.com

*Answer*

```python
first_name = input()
last_name = input()
role = input()
company = input()
first_initial = first_name[0].lower()
last_name = last_name.lower()
role = role.lower()
company = company.lower()
if role == "admin":
    email = f"{first_initial}{last_name}@admin.{company}.com"
else:
    email = f"{first_initial}{last_name}@{company}.com"
print(email)
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 4_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 14

## Section 1 : MCQ

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

```python
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*


2.  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*

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

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

*Answer*

44

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

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

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

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

*Answer*

10 15

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

5.  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*


6.  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*


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

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

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

**Answer**

True

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


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

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

**Answer**

27

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

9.  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*

10.  What is the output of the code shown?

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

**Answer**

x

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

11.  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*

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

```
num = -5
result = abs(num)
print(result)
```

*Answer*

5

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


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

```
def maximum(x, y):
    if x > y:
        return x
    elif x == y:
        return 'The numbers are equal'
    else:
        return y

print(maximum(2, 3))
```

*Answer*

3

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


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

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

*Answer*

TypeError

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


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

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

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

**Answer**

9 5

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 4_COD_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

## Section 1 : Coding

1.  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*

```python
# You are using Python
main_str = input()
sub_str = input()
starts_with = lambda s, prefix: s.startswith(prefix)
print(starts_with(main_str, sub_str))
```

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

2.  Problem Statement

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

Calculates the result of raising a given base to a specific exponent using Python's built-in pow() function.Displays all intermediate powers from base$^1$ 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
base = int(input())
exponent = int(input())
result = pow(base, exponent)
print(result)
powers = [pow(base, i) for i in range(1, exponent + 1)]
print(powers)
print(sum(powers))
```

*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*

```python
# You are using Python
def armstrong_number(number):
    num_str = str(number)
    num_digits = len(num_str)
    total = sum(int(digit) ** num_digits for digit in num_str)
    if total == number:
        print(f"{number} is an Armstrong number.")
    else:
        print(f"{number} is not an Armstrong number.")
n = int(input())
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):

# You are using Python
    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
        return uppercase_count, lowercase_count, digit_count, special_count

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

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

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


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*

```python
# You are using Python
message = input()
print(len(message))
```

*Status :* Correct

*Marks : 10/10*

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

## 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

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*

```python
num = int(input())
# You are using Python
def digital_root(num):
    while num >= 10:
        sum_digits = 0
        while num > 0:
            sum_digits += num % 10
            num //= 10
        num = sum_digits
    return num

print(digital_root(num))
```

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

2.  Problem Statement

Ella is designing a messaging application that needs to handle long text messages efficiently. To optimize storage and transmission, she plans to implement a text compression feature that replaces consecutive repeated

characters with the character followed by its count, while leaving non-repeated characters unchanged.

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

Function Specification: def compress_string(*args)

*Input Format*

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

*Output Format*

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: aaaBBBccc
Output: a3B3c3

*Answer*

```
# You are using Python
def compress_string(s, index=0, compressed=""):
    if index == len(s):
        return compressed
    count = 1
    while index + 1 < len(s) and s[index] == s[index + 1]:
        count += 1
        index += 1
    compressed += s[index] + (str(count) if count > 1 else "")
    return compress_string(s, index + 1, compressed)
input_string = input()
print(compress_string(input_string))
```

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

## 3. 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())
numbers = list(map(int, input().split()))
count_div_3 = len(list(filter(lambda x: x % 3 == 0, numbers)))
count_div_5 = len(list(filter(lambda x: x % 5 == 0, numbers)))
sum_div_3 = sum(filter(lambda x: x % 3 == 0, numbers))
sum_div_5 = sum(filter(lambda x: x % 5 == 0, numbers))
print(count_div_3)
print(count_div_5)
print(sum_div_3)
print(sum_div_5)
```

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

4.   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*

```
# You are using Python
sensor1 = float(input())
sensor2 = float(input())
temperature_diff = abs(sensor1 - sensor2)
print(f"Temperature difference: {temperature_diff:.2f} °C")
```

*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*

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

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

calculate_bmi(weight, height)
```

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

6.  Problem Statement

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

Help Sophia to complete her task.

Function Specification: def sum_digits(num)

*Input Format*

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

*Output Format*

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

Refer to the sample output for formatting specifications.

*Sample Test Case*

Input: 123245
Output: 17

*Answer*

```python
num = int(input())
```

```python
# You are using Python
def sum_digits(num):
    return sum(int(digit) for digit in str(num))
```

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

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 4_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

## Section 1 : Coding

1.  Problem Statement

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*

```python
# You are using Python
import string
password = input()
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)
types_count = sum([has_lower, has_upper, has_digit, has_special])
if len(password) >= 10 and types_count == 4:
    strength = "Strong"
elif len(password) >= 6 and types_count >= 2:
    strength = "Moderate"
else:
    strength = "Weak"
print(f"{password} is {strength}")
```

2.  Problem Statement

Arjun is working on a mathematical tool to manipulate lists of numbers. He needs a program that reads a list of integers and generates two lists: one containing the squares of the input numbers, and another containing the cubes. Arjun wants to use lambda functions for both tasks.

Write a program that computes the square and cube of each number in the input list using lambda functions.

*Input Format*

The input consists of a single line of space-separated integers representing the list of input numbers.

*Output Format*

The first line contains a list of the squared values of the input numbers.

The second line contains a list of the cubed values of the input numbers.

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: 1 2 3
Output: [1, 4, 9]
[1, 8, 27]

*Answer*

```
# You are using Python
numbers = list(map(int, input().split()))
squares = list(map(lambda x: x**2, numbers))
cubes = list(map(lambda x: x**3, numbers))
print(squares)
print(cubes)
```

3.   Problem Statement

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

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

*Input Format*

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

The second line contains n space-separated integers.

*Output Format*

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

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: 7
12 34 56 78 98 65 23
Output: 5
2

*Answer*

```python
# You are using Python
n = int(input())
numbers = list(map(int, input().split()))
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
```

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

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


4.  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*

#

```python
# You are using Python
DOMESTIC_RATE = 5.0
INTERNATIONAL_RATE = 10.0
REMOTE_RATE = 15.0
def calculate_shipping(weight, destination):
    if weight <= 0:
        print("Invalid weight. Weight must be greater than 0.")
        return None
    if destination == "Domestic":
        rate = DOMESTIC_RATE
    elif destination == "International":
        rate = INTERNATIONAL_RATE
    elif destination == "Remote":
        rate = REMOTE_RATE
    else:
        print("Invalid destination.")
        return None
    return weight * rate
try:
    weight = float(input())
    destination = input()
    shipping_cost = calculate_shipping(weight, destination)
except ValueError:
    print("Invalid input.")
```

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

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 5_MCQ

Attempt : 1
Total Mark : 20
Marks Obtained : 18

## Section 1 : MCQ

1.  What is the output of the following?

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

*Answer*

TrueTrue

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


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

*Answer*

Values of a dictionary must be unique

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

3.  What will be the output?

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

*Answer*

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

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

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

*Answer*

s1&amp;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*

5.  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*

6.   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*

TypeError: unsupported operand type

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

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

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

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

*Answer*

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

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

8.   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*

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

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

**Answer**

False

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

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

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

**Answer**

13

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

11.   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*

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

**Answer**

False

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

13.   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*

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

*Answer*

( )

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

15.  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*

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

*Answer*

t[3] = 45

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

17.  Which of the following is a Python tuple?

*Answer*

(1, 2, 3)

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

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

*Answer*

More than one key isn't allowed

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

19.  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*

20.  What is the output of the following code?

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

*Answer*

False

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

### REC_Python_Week 5_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

## Section 1 : Coding

1.  Problem Statement

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

Write a program to help Gowshik with this task.

Example:

Given list:

[1, 2, 3, 4]

[3, 5, 2, 1]

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

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

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

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

*Output Format*

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

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

*Answer*

```
# You are using Python
n = int(input())
list1 = list(map(int, input().split(',')))
list2 = list(map(int, input().split(',')))
result = tuple(list1[i] + list2[i] for i in range(n))
print(result)
```

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

2.  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

```python
# You are using Python
n = int(input())
result = {}
for _ in range(n):
    data = list(map(int, input().split()))
    customer_id = data[0]
    amounts = data[1:]
    total = sum(amounts)
    maximum = max(amounts)
```

```
    result[customer_id] = [total, maximum]
print(result)
```

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


3.  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*

```python
# You are using Python
N = int(input())
items = []
for _ in range(N):
    items.append(eval(input()))
threshold = int(input())
result = []
for _, quantities in items:
    result.extend(filter(lambda x: x > threshold, quantities))
print(*result)
```

*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
registered = set(map(int, input().split()))
attended = set(map(int, input().split()))
dropped_out = set(map(int, input().split()))
attended_registered = registered & attended
final_students = attended_registered - dropped_out
print(attended_registered)
print(final_students)
```

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

5.  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*

```python
# You are using Python
n = int(input())
freq = {}
for _ in range(n):
    pid = int(input())
    freq[pid] = freq.get(pid, 0) + 1
total_unique = len(freq)
average_freq = sum(freq.values()) / total_unique
print(freq)
print(f"Total Unique IDs: {total_unique}")
print(f"Average Frequency: {average_freq:.2f}")
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

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*

```python
# You are using Python
n = int(input())
elements = list(map(int, input().split()))
result = tuple((elements[i], elements[i+1]) for i in range(n-1)) + ((elements[-1], None),)
print(result)
```

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

2.  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

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

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

## 3. 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.

Input: 4

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

*Answer*

```python
# You are using Python
n = int(input())
primes = {}
num = 2
count = 1
while count <= n:
    for i in range(2, int(num**0.5)+1):
        if num % i == 0:
            break
    else:
        primes[count] = num
        count += 1
    num += 1
print(primes)
```

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


4.  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*

```
n = int(input())
event_ids = [int(input()) for _ in range(n)]
groups = []
group = [event_ids[0]]
for i in range(1, n):
    if event_ids[i] == event_ids[i - 1] + 1:
        group.append(event_ids[i])
    else:
        groups.append(tuple(group))
        group = [event_ids[i]]
groups.append(tuple(group))
for g in groups:
    if len(g) == 1:
        print(f"({g[0]})")
    else:
        print(g)
```

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

5.  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
elements=input().split()
seen=set()
result="
for e in elements:
    if e not in seen:
        seen.add(e)
        result+=e
print(result)
```

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

6.  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*

```
s=set(map(int,input().split()))
ch=int(input())
print(f"{{{' ,'.join(map(str,sorted(s,reverse=True)))}}}")
if ch==1:
    print(max(s))
elif ch==2:
    print(min(s))
elif ch==3:
    n1=int(input())
    s.discard(n1)
    print(f"{{{' ,'.join(map(str,sorted(s,reverse=True)))}}}")
else:
    print("Invalid choice")
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 5_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 37.5

## Section 1 : Coding

1.  Problem Statement

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

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

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

Your task is to help Samantha in implementing the same.

*Input Format*

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

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

*Output Format*

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

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

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

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: program
Peace
Output: ['a', 'p']
['c', 'e', 'g', 'm', 'o', 'r']
False
False

*Answer*

```python
# You are using Python
word1 = input().strip().lower()
word2 = input().strip().lower()
common_letters = sorted(set(word1) & set(word2))
unique_letters = sorted((set(word1) - set(word2)) | (set(word2) - set(word1)))
is_superset = set(word1) >= set(word2)
no_common_letters = len(common_letters) == 0
print(common_letters)
print(unique_letters)
```

```
print(is_superset)
print(no_common_letters)
```

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


2.   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*

```python
# You are using Python
k = int(input())
clubs = [set(map(int, input().split())) for _ in range(k)]
symmetric_diff = clubs[0]
for club in clubs[1:]:
    symmetric_diff ^= club
print(symmetric_diff)
print(sum(symmetric_diff))
```

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

3.  Problem Statement

Noah, a global analyst at a demographic research firm, has been tasked with identifying which country experienced the largest population growth over a two-year period. He has a dataset where each entry consists of a country code and its population figures for two consecutive years. Noah needs to determine which country had the highest increase in population and present the result in a specific format.

Help Noah by writing a program that outputs the country code with the largest population increase, along with the increase itself.

*Input Format*

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

Each of the following N blocks contains three lines:

1. The first line is a country code.
2. The second line is an integer representing the population of the country in the first year.
3. The third line is an integer representing the population of the country in the second year.

*Output Format*

The output displays the country code and the population increase in the format

{code: difference}, where code is the country code and difference is the increase in population.

Refer to the sample output for formatting specifications.

*Sample Test Case*

Input: 3
01
1000
1500
02
2000
2430
03
1500
3000
Output: {03:1500}

*Answer*

```python
# You are using Python
N = int(input())
max_increase = 0
country_code = ""
for _ in range(N):
    code = input().strip()
    pop1 = int(input())
    pop2 = int(input())
    increase = pop2 - pop1
    if increase > max_increase:
        max_increase = increase
        country_code = code
print(f"{{{country_code}:{max_increase}}}")
```

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

4.  Problem Statement

James is an engineer working on designing a new rocket propulsion

system. He needs to solve a quadratic equation to determine the optimal launch trajectory. The equation is of the form $ax^2 + bx + c = 0$.

Your task is to help James find the roots of this quadratic equation. Depending on the discriminant, the roots might be real and distinct, real and equal, or complex. Implement a program to determine and display the roots of the equation based on the given coefficients.

### Input Format

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

The second line contains three space-separated integers a,b, and c representing the coefficients of the quadratic equation.

### Output Format

The output displays:

1. If the discriminant is positive, display the two real roots.
2. If the discriminant is zero, display the repeated real root.
3. If the discriminant is negative, display the complex roots as a tuple with real and imaginary parts.

Refer to the sample output for formatting specifications.

### Sample Test Case

Input: 3
1 5 6
Output: (-2.0, -3.0)

### Answer

```
import math
def find_roots(a, b, c):
    discriminant = b**2 - 4*a*c
    if discriminant > 0:
        root1 = (-b + math.sqrt(discriminant)) / (2*a)
        root2 = (-b - math.sqrt(discriminant)) / (2*a)
        return (max(root1, root2), min(root1, root2))
```

```python
        elif discriminant == 0:
            root = -b / (2*a)
            return (root,)
        else:
            real_part = -b / (2*a)
            imaginary_part = math.sqrt(abs(discriminant)) / (2*a)
            return ((real_part, imaginary_part), (real_part, -imaginary_part))
N = int(input())
a, b, c = map(int, input().split())
roots = find_roots(a, b, c)
print(roots)
```

*Status :* Partially correct                                    *Marks : 7.5/10*

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 6_PAH

Attempt : 1
Total Mark : 30
Marks Obtained : 30

## Section 1 : Coding

1.   Problem Statement

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

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

### Input Format

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

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

*Output Format*

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

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

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

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

Refer to the sample outputs for the formatting specifications.

*Sample Test Case*

Input: 3
16 Sam
87 Sabari
43 Dani
Output: Student with ID 16 added to the database.
Student with ID 87 added to the database.
Student with ID 43 added to the database.

*Answer*

```python
MAX_CAPACITY = 30
def main():
    n = int(input())
    database = set()
    count = 0
    for _ in range(n):
        sid, name = input().split(maxsplit=1)
        sid = int(sid)
        try:
```

```
        if count >= MAX_CAPACITY:
            raise Exception("Student database is full.")
        if sid in database:
            raise Exception("Student ID already exists.")
        database.add(sid)
        count += 1
        print(f"Student with ID {sid} added to the database.")
    except Exception as e:
        print(f"Exception caught. Error: {e}")
        break
if __name__ == "__main__":
    main()
```

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


2.   Problem Statement

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

File Name: user_input.txt

*Input Format*

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

These numbers may be integers or decimals.

*Output Format*

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

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




Refer to the sample output for format specifications.

*Sample Test Case*

Input: 1 2 3 4 5

Output: Average of the numbers is: 3.00

*Answer*

```python
def main():
    try:
        line = input().strip()
    except EOFError:
        print("Invalid data in the input.")
        return

    if not line:
        print("Invalid data in the input.")
        return

    tokens = line.split()
    try:
        nums = [float(t) for t in tokens]
    except ValueError:
        print("Invalid data in the input.")
        return

    avg = sum(nums) / len(nums)
    print(f"Average of the numbers is: {avg:.2f}")

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

*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

```python
def main():
    filename = input().strip()
    content  = input()
    char     = input()
    if not char:
        print("Character not found in the file.")
        return
    target = char[0]
    try:
        with open(filename, 'w', encoding='utf-8') as f:
            f.write(content)
```

```python
    except IOError:
        pass
    try:
        with open(filename, 'r', encoding='utf-8') as f:
            data = f.read()
    except IOError:
        data = content
    count = sum(1 for c in data if c.lower() == target.lower())
    if count > 0:
        print(f"The character '{target}' appears {count} times in the file.")
    else:
        print("Character not found in the file.")

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

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 6_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

## Section 1 : Coding

1.  Problem Statement

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*

```python
# You are using Python
sentence = input().strip()
words = sentence.split()
print(len(words))
```

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


2.  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*

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

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

## 3. Problem Statement

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

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

### Input Format

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

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

### Output Format

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

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

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

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

Refer to the sample output for formatting specifications.

### Sample Test Case

Input: 20.0
5
Output: 100.0

### Answer

```
# You are using Python
try:
```

```
    price = float(input())
    quantity = int(input())
    if price <= 0:
        raise ValueError("Invalid Price")
    if quantity <= 0:
        raise ValueError("Invalid Quantity")
    total_cost = price * quantity
    if total_cost > 1000:
        raise RuntimeError("Excessive Cost")
    print(round(total_cost, 1))
except ValueError as ve:
    print(ve)
except RuntimeError as re:
    print(re)
```

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


4.  Problem Statement

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

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

File Name: text_file.txt

*Input Format*

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

*Output Format*

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

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

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

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

*Answer*

```python
# You are using Python
input_string = input()
with open('text_file.txt', 'w') as file:
    file.write(input_string)
with open('text_file.txt', 'r') as file:
    content = file.read()
print(f"Original String: {content}")
print(f"Upper-Case String: {content.upper()}")
print(f"Lower-Case String: {content.lower()}")
```

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

5.  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*

```python
# You are using Python
class NotEligibleToVote(Exception):
    pass
age=int(input())
try:
    if age<18:
        raise NotEligibleToVote
    print("Eligible to vote")
except NotEligibleToVote:
    print("Not eligible to vote")
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 6_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

## Section 1 : Coding

1.   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*

```python
# You are using Python
sentence = input().strip()
words = sentence.split()
print(len(words))
```

*Status :* <span style="color:green">Correct</span>                                           *Marks : 10/10*


2.   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*

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

# 3.  Problem Statement

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

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

### *Input Format*

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

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

### *Output Format*

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

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

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

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

Refer to the sample output for formatting specifications.

### *Sample Test Case*

Input: 20.0
5
Output: 100.0

### *Answer*

```
# You are using Python
try:
```

```
    price = float(input())
    quantity = int(input())
    if price <= 0:
        raise ValueError("Invalid Price")
    if quantity <= 0:
        raise ValueError("Invalid Quantity")
    total_cost = price * quantity
    if total_cost > 1000:
        raise RuntimeError("Excessive Cost")
    print(round(total_cost, 1))
except ValueError as ve:
    print(ve)
except RuntimeError as re:
    print(re)
```

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


4.  Problem Statement

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

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

File Name: text_file.txt

*Input Format*

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

*Output Format*

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

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

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

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

*Answer*

```python
# You are using Python
input_string = input()
with open('text_file.txt', 'w') as file:
    file.write(input_string)
with open('text_file.txt', 'r') as file:
    content = file.read()
print(f"Original String: {content}")
print(f"Upper-Case String: {content.upper()}")
print(f"Lower-Case String: {content.lower()}")
```

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


5.  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*

```python
# You are using Python
class NotEligibleToVote(Exception):
    pass
age=int(input())
try:
    if age<18:
        raise NotEligibleToVote
    print("Eligible to vote")
except NotEligibleToVote:
    print("Not eligible to vote")
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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.  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*

2.  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*

3. 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*

4. 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*

5. 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*

6. How do you rename a file?

*Answer*

os.rename(existing_name, new_name)

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


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

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

Meri,25

John,21

Raj,20

Ouptput:

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


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

*Answer*

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

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


8. 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*


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

*Answer*

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

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

10. 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*

11. 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*

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

*Answer*

raise Exception()

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

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

*Answer*

error

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

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

```
# 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*

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

*Answer*

finally

16.   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

17.   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)

18. 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*

19. 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*

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

*Answer*

To handle exceptions during code execution

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 7_CY

Attempt : 1
Total Mark : 50
Marks Obtained : 40

## Section 1 : Coding

1.   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
sensors, samples = map(int, input().split())
data = [list(map(float, input().split())) for _ in range(sensors)]
arr = np.array(data)
means = arr.mean(axis=1, keepdims=True)
stds = arr.std(axis=1, keepdims=True)
normalized = (arr - means) / stds
print("Normalized Sensor Data:", normalized)
```

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

2.  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

data = [list(map(float, input().split())) for _ in range(5)]
arr = np.array(data)

yearly_totals = arr.sum(axis=1)
max_rainfall_months = arr.argmax(axis=1)

print(yearly_totals, max_rainfall_months)
```

*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*

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

n = int(sys.stdin.readline().strip())
header = sys.stdin.readline().strip().split(',')

data = [sys.stdin.readline().strip().split(',', len(header)-1) for _ in range(n)]
df = pd.DataFrame(data, columns=header)

df['Purchase Date'] = pd.to_datetime(df['Purchase Date'], errors='coerce')
df['Purchase Month'] = df['Purchase Date'].dt.month
```

```
df['Purchase Day'] = df['Purchase Date'].dt.day

print("Transformed E-commerce Transaction Data:")
print(df)
```

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

4.  Problem Statement

Arjun is monitoring hourly temperature data recorded continuously for multiple days. He needs to calculate the average temperature for each day based on 24 hourly readings.

Help him to implement the task using the numpy package.

Formula:

Reshape the temperature readings into rows where each row has 24 readings (one day).

Average temperature per day = mean of 24 hourly readings in each row.

*Input Format*

The first line of input consists of an integer value, n, representing the total number of temperature readings.

The second line of input consists of n floating-point values separated by spaces, representing hourly temperature readings.

*Output Format*

The output prints: avg_per_day

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: 30
30.0 30.0 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*

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

n = int(input().strip())
temps = list(map(float, input().strip().split()))
arr = np.array(temps).reshape(-1, 24)
avg_per_day = arr.mean(axis=1)
print(avg_per_day
```

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


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*

```
import pandas as pd
n = int(input())
products = []
for _ in range(n):
    product_id, quantity = input().split()
    quantity = int(quantity)
    products.append((product_id, quantity))

df = pd.DataFrame(products, columns=["Product_ID", "Quantity"])
zero_stock_df = df[df["Quantity"] == 0]

print("Products with Zero Stock:")
if zero_stock_df.empty:
    print("No products with zero stock.")
else:
    print(zero_stock_df.to_string(index=False))
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 7_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 40

## Section 1 : Coding

1.   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*

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

rows = int(input())
cols = int(input())
pixels = [list(map(int, input().split())) for _ in range(rows)]

img = np.array(pixels, dtype=float)
min_pixel = img.min()
max_pixel = img.max()

if max_pixel == min_pixel:
    normalized_img = np.zeros_like(img)
else:
```

```
normalized_img = (img - min_pixel) / (max_pixel - min_pixel)

print(normalized_img)
```

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

2.  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*

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

rows, cols = map(int, input().split())
data = [list(map(float, input().split())) for _ in range(rows)]
arr = np.array(data)

positive_days = np.all(arr > 0, axis=1)

print("Days where all asset returns were positive:", positive_days)
```

*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).

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*

```python
# You are using Python
import numpy as np
n, *rest = map(float, input().split())
if len(rest) < n:
    rest += list(map(float, input().split()))
wait_times = np.array(rest[:int(n)])

max_time = wait_times.max()
bucket_size = 5
bins = np.arange(0, bucket_size * (int(np.ceil(max_time / bucket_size)) + 1),
bucket_size)

indices = np.digitize(wait_times, bins, right=False) - 1

print("Wait Time Buckets and Counts:", end=" ")
for i in range(len(bins) - 1):
count = np.sum(indices == i)
print(f"{int(bins[i])}-{int(bins[i+1])}: {count}", end=" ")
```

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

4.  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*

```python
import pandas as pd

n = int(input())
data = [input().split() for _ in range(n)]

names = [row[0] for row in data]
years = [float(row[1]) for row in data]
```

```
df = pd.DataFrame({'Name': names, 'Years at Company': years})

def classify_years(y):
    if y < 3:
        return 'Junior'
    elif 3 <= y < 6:
        return 'Mid'
    else:
        return 'Senior'

df['Experience Level'] = df['Years at Company'].apply(classify_years)

print("Employee Data with Experience Level:", end=' ')
print(df.to_string(index=False))
```

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


5.   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*

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

n = int(input())
data = []
for _ in range(n):
    line = sys.stdin.readline().rstrip('\n')
    parts = line.split(maxsplit=1)
    rid = parts[0]
    fb = parts[1] if len(parts) > 1 else ""
    data.append([rid, fb])

df = pd.DataFrame(data, columns=["RespondentID", "Feedback"])
df["Feedback"] = df["Feedback"].replace("", "No Response")
df["Feedback"] = df["Feedback"].fillna("No Response")

print("Survey Responses with Missing Feedback Filled:", end=" ")
print(df.to_string(index=False))
```

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

# Rajalakshmi Engineering College

Name: Karthik  Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 7_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

## Section 1 : Coding

1.  Problem Statement

Sita works as a sales analyst and needs to analyze monthly sales data for different cities. She receives lists of cities, months, and corresponding sales values and wants to create a pandas DataFrame using a MultiIndex of cities and months.

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

### Input Format

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

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

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

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

**Output Format**

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

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

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

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

Refer to the sample output for the formatting specifications.

**Sample Test Case**

Input: 4
NYC NYC LA LA
Jan Feb Jan Feb
100 200 300 400

Output: Monthly Sales Data with MultiIndex:
         Sales
City Month
NYC  Jan    100.0
     Feb    200.0
LA   Jan    300.0
     Feb    400.0

Total Sales Per City:
     Sales
City
LA    700.0
NYC   300.0

**Answer**

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

```
n = int(input())
cities = input().split()
months = input().split()
sales = list(map(float, input().split()))
index = pd.MultiIndex.from_tuples(list(zip(cities, months)), names=['City',
'Month'])
df = pd.DataFrame(data={'Sales': sales}, index=index)
print("Monthly Sales Data with MultiIndex:")
print(df)
total_sales = df.groupby(level='City').sum()
print("Total Sales Per City:")
print(total_sales)
```

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


## 2.  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*

```
import numpy as np
array1 = np.array(list(map(float, input().split())))
array2 = np.array(list(map(float, input().split())))
cross_corr = np.correlate(array1, array2, mode='full')
print("Cross-correlation of the two time series:", cross_corr)
```

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


3.   Problem Statement

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

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

*Input Format*

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

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

*Output Format*

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

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

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: 2 4
10 20 30 40
5 15 25 35
Output: Cumulative Monthly Sales:
[[ 10  30  60 100]
 [  5  20  45  80]]

*Answer*

```
# You are using Python
import numpy as np
products, months = map(int, input().split())
data = [list(map(int, input().split())) for _ in range(products)]
arr = np.array(data)
cumulative_array = np.cumsum(arr, axis=1)
print("Cumulative Monthly Sales:", cumulative_array)
```

*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*

```python
# You are using Python
import numpy as np
n = int(input())
sales = np.array(list(map(float, input().split())))
filtered_sales = sales[(sales % 5 == 0) & (sales > 100)]
print(filtered_sales)
```

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


5.   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*

```python
# You are using Python
import pandas as pd
import sys
import io
n = int(input())
header = input()
data = [input() for _ in range(n)]
csv_data = '\n'.join([header] + data)
df = pd.read_csv(io.StringIO(csv_data))
mean_age = df['Age'].mean()
df['Age'].fillna(mean_age, inplace=True)
df = df.dropna(subset=['Diagnosis']).reset_index(drop=True)
print("Cleaned Hospital Records:")
print(df)
```

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

# Rajalakshmi Engineering College

Name: Karthik Subramanian
Email: 240701233@rajalakshmi.edu.in
Roll no: 240701233
Phone: 9384661541
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 7_MCQ

Attempt : 1
Total Mark : 20
Marks Obtained : 17

## Section 1 : MCQ

1.  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*

2.  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*


3.   What does the np.arange(10) function in NumPy do?

*Answer*

Creates an array with values from 0 to 10

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


4.   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*


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

*Answer*

numpy.std()

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


6.   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*


7.  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*


8.  The important data structure of pandas is/are ____.

*Answer*

Both Series and Data Frame

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


9.  Which NumPy function is used to create an identity matrix?

*Answer*

numpy.eye()

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

10.   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*

11.   What does NumPy stand for?

*Answer*

Numerical Python

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

12.   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*

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

*Answer*

pd.DataFrame()

14. 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*

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

**Answer**

Array

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

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

**Answer**

import numpy as np

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

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

**Answer**

0

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

18. 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*

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

**Answer**

arr[0]

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

20. What is the primary purpose of Pandas DataFrame?

**Answer**

To store data in tabular form for analysis and manipulation

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