Rajalakshmi Engineering College

Name: Prithvi Yogeswaraa M
Email: 241501154@rajalakshmi.edu.in
Roll no: 2116241501154
Phone: 9345507776
Branch: REC
Department: I AIML AD
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 4_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 11

Section 1 : MCQ

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

Answer

map()

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

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

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

Answer

0

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

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

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

Answer

212.032.0

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

4. 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***

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

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

display(10, 20, 30)
```

Answer

102030

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

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

```
def display(b, n):
   while n > 0:
      print(b,end="")
      n=n-1
display('z',3)
```

Answer

An exception is executed

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

7. What is the output of the code shown?

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

Answer

worldhellohello

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

8. 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***

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

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

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

Answer

16hello

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

10. 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***

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

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

print(result)

Answer

43

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

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

Answer

lambda

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

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

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

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

Answer

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

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

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

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

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

Answer

9 5

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

15. 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***

Rajalakshmi Engineering College

Name: Prithvi Yogeswaraa M
Email: 241501154@rajalakshmi.edu.in
Roll no: 2116241501154
Phone: 9345507776
Branch: REC
Department: I AIML AD
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 4_COD_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

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

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

Function Signature: armstrong_number(number)

Input Format

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

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 153
Output: 153 is an Armstrong number.

Answer

```
def is_armstrong_number(number):

    str_number = str(number)
    num_digits = len(str_number)


    armstrong_sum = sum(int(digit) ** num_digits for digit in str_number)


    return armstrong_sum == number

def armstrong_number(number):
    if is_armstrong_number(number):
        print(f"{number} is an Armstrong number.")
    else:
        print(f"{number} is not an Armstrong number.")


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


armstrong_number(n)
```

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

2. Problem Statement

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

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

Input Format

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

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

Output Format

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

Refer to the sample for the formatting specifications.

Sample Test Case

Input: Examly
e
Output: False

Answer

```
main_string = input().strip()


substring = input().strip()

starts_with = lambda s, sub: s.startswith(sub)


result = starts_with(main_string, substring)
print(result)
```

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

3. Problem Statement

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

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

Help her build this program to automate her calculations.

Input Format

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

Output Format

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

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

The third line prints the sum of all these powers.

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

```
def advanced_exponential_calculator(base, exponent):
    result = pow(base, exponent)

    intermediate_powers = [pow(base, i) for i in range(1, exponent + 1)]

    sum_of_powers = sum(intermediate_powers)

    print(result)
    print(intermediate_powers)
    print(sum_of_powers)

base = int(input().strip())
exponent = int(input().strip())

advanced_exponential_calculator(base, exponent)
```

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

4. Problem Statement

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

Input Format

The input consists of a string representing the message.

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: hello!!
Output: 7

Answer

```
def message_length_calculator(message):

    length = len(message)
    return length


message = input().strip()


length_of_message = message_length_calculator(message)
print(length_of_message)
```

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

5. Problem Statement

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

Function Signature: analyze_string(input_string)

Input Format

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

Output Format

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

The second line contains an integer representing the count of lowercase letters

in the format "Lowercase letters: [count]".

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: Hello123
Output: Uppercase letters: 1
Lowercase letters: 4
Digits: 3
Special characters: 0

Answer

```
def analyze_string(input_string):

    uppercase_count = 0
    lowercase_count = 0
    digit_count = 0
    special_count = 0

    for char in input_string:
        if char.isupper():
            uppercase_count += 1
        elif char.islower():
            lowercase_count += 1
        elif char.isdigit():
            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***

Rajalakshmi Engineering College

Name: Prithvi Yogeswaraa M
Email: 241501154@rajalakshmi.edu.in
Roll no: 2116241501154
Phone: 9345507776
Branch: REC
Department: I AIML AD
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 4_PAH_Updated

Attempt : 1
Total Mark : 60
Marks Obtained : 60

Section 1 : Coding

1. Problem Statement

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

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

Function Signature: calculate_bmi(weight, height)

Formula: BMI = Weight/(Height)2

Input Format

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

weight in kilograms.

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

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 70.0
1.75
Output: Your BMI is: 22.86

Answer

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

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***

2. Problem Statement

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

Help Sophia to complete her task.

Function Specification: def sum_digits(num)

Input Format

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 123245
Output: 17

Answer

```
num = int(input())

# You are using Python
def sum_digits(num):
    digit_sum = 0
    while num > 0:
        digit_sum += num % 10
        num //= 10
    return digit_sum


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

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

3. 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):
    # Base case: if the string is empty, return an empty string
    if not s:
        return ""

    # Initialize variables
    compressed = ""
    count = 1

    # Iterate through the string
    for i in range(1, len(s)):
        if s[i] == s[i - 1]:
            count += 1  # Increment count for repeated characters
        else:
            # Append the character and its count (if greater than 1) to the
compressed string
```

```
            compressed += s[i - 1] + (str(count) if count > 1 else "")
            count = 1  # Reset count for the new character

    # Handle the last character and its count
    compressed += s[-1] + (str(count) if count > 1 else "")

    return compressed

# Input reading
if __name__ == "__main__":
    input_string = input().strip()
    result = compress_string(input_string)
    print(result)
```

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

```
sensor1 = float(input())
sensor2 = float(input())


difference = abs(sensor1 - sensor2)


print(f"Temperature difference: {difference:.2f} °C")
```

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

5. Problem Statement

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

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

Additionally, the program should calculate the total sum of all numbers divisible by 3 and divisible by 5 separately.

Input Format

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

The second line contains n space-separated integers.

Output Format

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

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

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 6
3 5 6 10 15 20
Output: 3
4
24
50

Answer

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


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


print(len(div_by_3))
print(len(div_by_5))
print(sum(div_by_3))
print(sum(div_by_5))
```

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

6. Problem Statement

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

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

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

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

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

Function prototype: def digital_root(num)

Input Format

The input consists of an integer num.

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 451110

Output: 3

Answer

```
num = int(input())

def digital_root(num):
    while num >= 10:
        digit_sum = 0
        while num > 0:
            digit_sum += num % 10
            num //= 10
```

```
        num = digit_sum
    return num

# Header snippet

print(digital_root(num))
```

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

Rajalakshmi Engineering College

Name: Prithvi Yogeswaraa M
Email: 241501154@rajalakshmi.edu.in
Roll no: 2116241501154
Phone: 9345507776
Branch: REC
Department: I AIML AD
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 4_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

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

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

Function Signature: count_substrings(text, substring)

Input Format

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

The second line consists of a string representing the substring.

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

```
# You are using Python
def count_substrings(text, substring):
    # Counting occurrences of the substring in the text
    count = text.count(substring)

    # Printing the result as per the required format
    print(f"The substring '{substring}' appears {count} times in the text.")
```

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

2. Problem Statement

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

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

Input Format

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

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

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10 15 24 8 37 16
Output: The maximum even price is: 24

Answer

```
def find_max_even_price(prices):

    price_list = list(map(int, prices.split()))


    even_prices = [price for price in price_list if price % 2 == 0]


    if even_prices:
        max_even_price = max(even_prices)
        print(f"The maximum even price is: {max_even_price}")
    else:
        print("No even prices were found")


prices = input()


find_max_even_price(prices)
```

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

3. Problem Statement

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

Input Format

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

Output Format

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

Refer to the sample output for the exact format.

Sample Test Case

Input: 24

Output: The smallest positive divisor of 24 is: 2

Answer

```
def smallest_divisor(n):
    # Start from 2 because 1 is always a divisor of any number
    divisors = [i for i in range(2, n+1) if n % i == 0]

    # The smallest divisor will be the first element in the divisors list
    print(f"The smallest positive divisor of {n} is: {min(divisors)}")

# Input reading
n = int(input())

# Call the function
smallest_divisor(n)
```

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

4. Problem Statement

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

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

Character Types Considered:

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

Input Format

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

Output Format

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

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

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: password123
Output: password123 is Moderate

Answer

```
import string

def password_strength(password):

    has_lower = any(c.islower() for c in password)
    has_upper = any(c.isupper() for c in password)
    has_digit = any(c.isdigit() for c in password)
    has_special = any(c in string.punctuation for c in password)

    # Calculate how many of the character types are present
    types_count = sum([has_lower, has_upper, has_digit, has_special])


    if len(password) >= 10 and types_count == 4:
        print(f"{password} is Strong")
    elif len(password) >= 6 and types_count >= 2:
        print(f"{password} is Moderate")
    else:
        print(f"{password} is Weak")

# Input
password = input()

# Call the function
password_strength(password)
```

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