Rajalakshmi Engineering College

Name: Nishal P
Email: 241501130@rajalakshmi.edu.in
Roll no: 241501130
Phone: 9790965308
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 4_COD_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

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

Function Signature: analyze_string(input_string)

Input Format

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

Output Format

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

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

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: Hello123

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

Answer

```
def analyze_string(input_string):
    uppercase_count=sum(1 for char in input_string if char.isupper())
    lowercase_count=sum(1 for char in input_string if char.islower())
    digit_count=sum(1 for char in input_string if char.isdigit())
    special_count=sum(1 for char in input_string if not char.isalnum())
    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***

2. Problem Statement

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

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

Input Format

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

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

Output Format

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

Refer to the sample for the formatting specifications.

Sample Test Case

Input: Examly
e
Output: False

Answer

```
# You are using Python
def check_start(main_string, substring):
    starts_with = lambda s, sub: s.startswith(sub)
    return starts_with(main_string, substring)

main_string = input().strip()
substring = input().strip()

result = check_start(main_string, substring)
```

print(result)

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

3. Problem Statement

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

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

Function Signature: armstrong_number(number)

Input Format

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

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 153
Output: 153 is an Armstrong number.

Answer

```
# You are using Python
def is_armstrong_number(number):
    digits = [int(d) for d in str(number)]
    num_digits = len(digits)
    armstrong_sum = sum(d ** num_digits for d in digits)
    return armstrong_sum == number
```

```
number = int(input().strip())

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

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

```
# You are using Python
message = input().strip()
length_of_message = len(message)
print(length_of_message)
```

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

5. Problem Statement

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

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

Help her build this program to automate her calculations.

Input Format

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

Output Format

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

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

The third line prints the sum of all these powers.

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

```
# You are using Python
base = int(input().strip())
exponent = int(input().strip())
```

```
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)
```

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

Rajalakshmi Engineering College

Name: Nishal P
Email: 241501130@rajalakshmi.edu.in
Roll no: 241501130
Phone: 9790965308
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 4_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 38.5

Section 1 : Coding

1. Problem Statement

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

#

```
DOMESTIC_RATE = 5.0
INTERNATIONAL_RATE = 10.0
REMOTE_RATE = 15.0

weight = float(input())
destination = input()
shipping_cost=0
```

```
if weight>=0 and destination!="known":
    if destination == "Domestic":
        shipping_cost = weight * DOMESTIC_RATE
    elif destination == "International":
        shipping_cost = weight * INTERNATIONAL_RATE
    elif destination == "Remote":
        shipping_cost = weight * REMOTE_RATE
elif weight>=0 and destination=='known':
    shipping_cost=None
    print("Invalid destination.")

else:
    shipping_cost=None
    print("Invalid weight. Weight must be greater than 0.")
if shipping_cost is not None:
    print(f"Shipping cost to {destination} for a {weight} kg package:
${shipping_cost:.2f}")
```

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

2. 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

```
def check_pass(password):
    length=len(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(not c.isalnum() for c in password)

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

    if length<6 or char_types <2:
        return f"{password} is weak"
    elif length>=10 and char_types==4:
        return f"{password} is Strong"
    else:
        return f"{password} is Moderate"

password=input()
print(check_pass(password))
```

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

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

```
# You are using Python
n = int(input().strip())
numbers = list(map(int, input().strip().split()))

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

```
print(count_even)
print(count_odd)
```

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

4. Problem Statement

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

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

Input Format

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

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

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

```
# You are using Python
prices = input().strip().split()
```

```
prices = list(map(int, prices))

even_prices = [price for price in prices 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")
```

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