Rajalakshmi Engineering College

Name: DEVISHREE J
Email: 240701702@rajalakshmi.edu.in
Roll no: 240701702
Phone: 7397766309
Branch: REC
Department: I CSE FF
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

```
pas=input()
def check(pas):
    lower=False
    upper=False
    digit=False
    special=False
    for ch in pas:
        if ch.islower():
            lower=True
        elif ch.isupper():
            upper=True
        elif ch.isdigit():
            digit=True
        else:
            special=True
    if len(pas)>=10 and lower and upper and digit and special:
```

```
        strength="Strong"
    elif(lower or upper)and digit:
        strength="Moderate"
    else:
        strength="Weak"
    print(f"{pas} is {strength}")
check(pas)
```

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

2. 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

```
n=int(input())
divisors=[i for i in range(2,n+1)if n%i==0]
smallest=min(divisors)
print(f"The smallest positive divisor of {n} is:{smallest}")
```

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

3. Problem Statement

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

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

Function Signature: count_substrings(text, substring)

Input Format

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

The second line consists of a string representing the substring.

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

```
def count_substrings(text,substring):
    count=text.count(substring)
    print(f"The substring '{substring}' appears {count} times in the text.")

text=input()
```

```
substring=input()
count_substrings(text,substring)
```

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

```
def find_max_even_price(prices):
    priceslist=list(map(int,prices.split()))
```

```
    evenprices=[price for price in priceslist if price%2==0]
    if evenprices:
        max_even_price=max(evenprices)
        print(f"The maximum even price is: {max_even_price}")

    else:
        print("No even prices were found")

input_prices=input()
find_max_even_price(input_prices)
```

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