Rajalakshmi Engineering College

Name: Tanisha Sudhagar
Email: 240701553@rajalakshmi.edu.in
Roll no: 2116240701553
Phone: 6374776137
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

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

```
# You are using Python
n=int(input())
a=min([i for i in range(2,n+1)if(n%i==0)])
print("The smallest positive divisor of ",n,"is : ",a)
```

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

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_highest_even_price(prices_str):
    prices = list(map(int, prices_str.split()))
    even_prices = [price for price in prices if price % 2 == 0]
    if even_prices:
        return max(even_prices)
    else:
        return None

if __name__ == "__main__":
    prices_input = input()
    highest_even_price = find_highest_even_price(prices_input)

    if highest_even_price is not None:
        print(f"The maximum even price is: {highest_even_price}")
    else:
        print("No even prices were found")
```

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

3. 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
def calculate_squares_and_cubes(numbers_str):
    numbers = list(map(int, numbers_str.split()))
    squares = list(map(lambda x: x**2, numbers))
    cubes = list(map(lambda x: x**3, numbers))
    return squares, cubes
if __name__ == "__main__":
    numbers_input = input()
    squares, cubes = calculate_squares_and_cubes(numbers_input)
    print(squares)
    print(cubes)
```

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

```
def password_strength(n):
   length=len(n)
   l=0
   u=0
   d=0
   s=0
```

```
    for char in n:
        if (char.islower()):
            l+=1
        elif (char.isupper()):
            u+=1
        elif (char.isdigit()):
            d+=1
        else:
            s+=1
    char_t=0
    if l>0:
        char_t+=1
    if(u>0):
        char_t+=1
    if(d>0):
        char_t+=1
    if(s>0):
        char_t+=1


    if length < 6 or char_t < 2:
        return f"{n} is Weak"
    elif (length >= 6 and char_t >= 2) and (length <=10 and char_t <4):
        return f"{n} is Moderate"
    elif length >= 10 and char_t == 4:
        return f"{n} is Strong"
    else:
        return f"{n} is Moderate"
n=input()
print(password_strength(n))
```

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