Rajalakshmi Engineering College

Name: Kaviya B J
Email: 240701246@rajalakshmi.edu.in
Roll no: 240701246
Phone: 9345986507
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_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Alex is creating an account and needs to set up a password. The program prompts Alex to enter their name, mobile number, chosen username, and desired password. Password validation criteria include:

Length between 10 and 20 characters.At least one digit.At least one special character from !@#$%^&amp;* set. Display "Valid Password" if criteria are met; otherwise, raise an exception with an appropriate error message.

Input Format

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

The second line of the input consists of the mobile number as a string.

The third line of the input consists of the username as a string.

The fourth line of the input consists of the password as a string.

Output Format

If the password is valid (meets all the criteria), it will print "Valid Password"

If the password is weak (fails any one or more criteria), it will print an error message accordingly.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: John
9874563210
john
john1#nhoj

Output: Valid Password

Answer

```
def validate_password(password):
    # Check length first
    if len(password) < 10 or len(password) > 20:
        return "Should be a minimum of 10 characters and a maximum of 20
characters"

    # Check for at least one digit
    if not any(char.isdigit() for char in password):
        return "Should contain at least one digit"

    # Check for at least one special character
    special_characters = set("!@#$%^&*")
    if not any(char in special_characters for char in password):
        return "It should contain at least one special character"

    return "Valid Password"

def main():
    name = input().strip()
    mobile_number = input().strip()
    username = input().strip()
    password = input().strip()

    # Validate the password
    result = validate_password(password)
    print(result)

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

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

2. Problem Statement

Write a program to obtain the start time and end time for the stage event show. If the user enters a different format other than specified, an exception occurs and the program is interrupted. To avoid that, handle the exception and prompt the user to enter the right format as specified.

Start time and end time should be in the format 'YYYY-MM-DD HH:MM:SS'If the input is in the above format, print the start time and end time.If the input does not follow the above format, print "Event time is not in the format "

Input Format

The first line of input consists of the start time of the event.

The second line of the input consists of the end time of the event.

Output Format

If the input is in the given format, print the start time and end time.

If the input does not follow the given format, print "Event time is not in the format".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2022-01-12 06:10:00
2022-02-12 10:10:12

Output: 2022-01-12 06:10:00
2022-02-12 10:10:12

Answer

```
from datetime import datetime

start_time = input().strip()
end_time = input().strip()
time_format = "%Y-%m-%d %H:%M:%S"

try:
   datetime.strptime(start_time, time_format)
   datetime.strptime(end_time, time_format)
   print(f"{start_time} {end_time}")
except ValueError:
   print("Event time is not in the format")
```

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

3. Problem Statement

Write a program to read the Register Number and Mobile Number of a student. Create user-defined exception and handle the following:

If the Register Number does not contain exactly 9 characters in the specified format(2 numbers followed by 3 characters followed by 4 numbers) or if the Mobile Number does not contain exactly 10 characters, throw an IllegalArgumentException. If the Mobile Number contains any character other than a digit, raise a NumberFormatException.If the Register Number contains any character other than digits and alphabets, throw a NoSuchElementException.If they are valid, print the message 'valid' or else print an Invalid message.

Input Format

The first line of the input consists of a string representing the Register number.

The second line of the input consists of a string representing the Mobile number.

Output Format

The output should display any one of the following messages:

If both numbers are valid, print "Valid".

If an exception is raised, print "Invalid with exception message: ", followed by the specific exception message.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 19ABC1001
9949596920

Output: Valid

Answer

```
class IllegalArgumentException(Exception):
    pass

class NumberFormatException(Exception):
    pass

class NoSuchElementException(Exception):
    pass

def validate_register_number(reg_no):
    if len(reg_no) != 9:
        raise IllegalArgumentException("Register Number should have exactly 9 characters.")
    if not reg_no[:2].isdigit() or not reg_no[2:5].isalpha() or not reg_no[5:].isdigit():
        raise IllegalArgumentException("Register Number should have the format: 2 numbers, 3 characters, and 4 numbers.")
    if not reg_no.isalnum():
        raise NoSuchElementException("Register Number should only contain digits and alphabets.")

def validate_mobile_number(mob_no):
    if len(mob_no) != 10:
        raise IllegalArgumentException("Mobile Number should have exactly 10 characters.")
    if not mob_no.isdigit():
        raise NumberFormatException("Mobile Number should only contain digits.")

reg_no = input().strip()
mob_no = input().strip()

try:
    validate_register_number(reg_no)
    validate_mobile_number(mob_no)
    print("Valid")
except (IllegalArgumentException, NumberFormatException, NoSuchElementException) as e:
    print(f"Invalid with exception message: {e}")
```

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

4. Problem Statement

Implement a program that checks whether a set of three input values can form the sides of a valid triangle. The program defines a function is_valid_triangle that takes three side lengths as arguments and raises a ValueError if any side length is not a positive value. It then checks whether the sum of any two sides is greater than the third side to determine the validity of the triangle.

Input Format

The first line of input consists of an integer A, representing side1.

The second line of input consists of an integer B, representing side2.

The third line of input consists of an integer C, representing side3.

Output Format

The output prints either "It's a valid triangle" if the input side lengths form a valid triangle,

or "It's not a valid triangle" if they do not.

If there is a ValueError, it should print "ValueError: <error_message>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
4
5
Output: It's a valid triangle

Answer

```
def is_valid_triangle(a, b, c):
   if a <= 0 or b <= 0 or c <= 0:
      raise ValueError("Side lengths must be positive")
   return a + b > c and a + c > b and b + c > a
try:
   A = int(input())
   B = int(input())
   C = int(input())
   if is_valid_triangle(A, B, C):
      print("It's a valid triangle")
   else:
      print("It's not a valid triangle")
except ValueError as ve:
   print(f"ValueError: {ve}")
```

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