Validation Types Explanation:

- Email validation: RegExp(r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$')
- Phone number validation: RegExp(r'^\d{10}$')
- Name validation: RegExp(r'^[a-zA-Z\s]+$')

Explanation:

1.Email validation: RegExp(r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'):

The provided Regular Expression (RegExp) validates email addresses. Here's a breakdown of the pattern:

Code Explanation:

RegExp(r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$')

1. **^**

- **Anchor**: Ensures that the match starts at the beginning of the string.

2. **[a-zA-Z0-9._%+-]+**

- **Character Class ([]):** Matches a single character from the specified set:
  - **a-zA-Z:** Allows lowercase and uppercase letters.
  - **0-9:** Allows digits.
  - **._%+-:** Allows the following special characters: . (dot), _ (underscore), % (percent), + (plus), - (dash).
- **Quantifier (+):** Matches one or more of these characters (required for a valid email username).

3. **@**

- Matches the **@** symbol, which separates the username from the domain in an email address.

4. **[a-zA-Z0-9.-]+**

- Matches the domain name:
  - **a-zA-Z:** Allows letters.
  - **0-9:** Allows digits.

    - **.-:** Allows the dot (.) and dash (-) characters.
- **Quantifier (+):** Matches one or more of these characters.

5. **\.**

- **Escaped dot (\.):** Matches a literal dot (.). This ensures that the domain includes a "dot" for the top-level domain (e.g., .com, .org).

6. **[a-zA-Z]{2,}**

- Matches the top-level domain (TLD), like .com or .net:
    - **a-zA-Z:** Ensures only letters are used.
    - **{2,}:** Specifies a minimum length of 2 characters (e.g., .uk) and no maximum limit.

7. **$**

- **Anchor:** Ensures that the match ends at the end of the string.

Usage

The regular expression ensures that:

1. The email starts with a valid username.
2. The username is followed by an @ symbol.
3. A valid domain follows, containing letters, numbers, dots, or dashes.
4. A top-level domain (at least two letters) ends the email.

2. Phone number validation: RegExp(r'^\d{10}$') :

The provided **Regular Expression (RegExp)** validates phone numbers to ensure they are exactly 10 digits long. Here's a breakdown of the pattern:

Code Explanation:

```
RegExp(r'^\d{10}$')
```

1. ^

- **Anchor**: Ensures the match starts at the beginning of the string.
  This ensures that no extra characters are allowed before the phone number.

2. \d

- **Digit Matcher**: Matches a single digit (equivalent to [0-9]).

3. {10}

- **Quantifier**: Matches exactly **10 occurrences** of the preceding pattern (\d).
  This ensures the phone number contains exactly 10 digits.

4. $

- **Anchor**: Ensures the match ends at the end of the string.
  This prevents any extra characters after the 10 digits.

Usage

The regular expression ensures that:

1. The input string starts with a digit (^).
2. The input contains exactly 10 digits (\d{10}).
3. There are no extra characters before or after the digits (^ and $).

3.Name validation: RegExp(r'^[a-zA-Z\s]+$')

The provided **Regular Expression (RegExp)** validates a **name** to ensure it contains only alphabetic characters and spaces. Here's a detailed breakdown of the pattern:

Code Explanation

RegExp(r'^[a-zA-Z\s]+$')

1. ^

- **Anchor**: Ensures the match starts at the beginning of the string.
  This prevents any extra characters before the name.

2. [a-zA-Z\s]

- **Character Class ([])**: Matches a single character from the specified set:
  - a-zA-Z: Allows both **lowercase** (a-z) and **uppercase** (A-Z) letters.
  - \s: Matches a **whitespace character** (spaces, tabs, etc.), allowing names with spaces (e.g., "John Doe").

3. +

- **Quantifier**: Matches **one or more** occurrences of the preceding pattern ([a-zA-Z\s]).
  This ensures the name is not empty and contains at least one valid character.

4. $

- **Anchor**: Ensures the match ends at the end of the string.
  This prevents any extra characters after the name.

<u>Usage</u>

This regular expression ensures that:

1. The input contains only letters (a-z or A-Z) and spaces.
2. The input can include multiple words (e.g., "John Smith").
3. No numbers, special characters, or extra spaces are allowed outside the pattern.