Validating Passwords and email address using regular expression

Validating Email Address using Regular Expression

Why validate an email?
When users enter an email, you need to ensure it has a **valid format** like:

username@domain.extension

Example: test@example.com
Without validation, people might enter wrong emails like:
- test@.com
- @gmail.com
- abcgmail.com

Pattern Format
A **valid email address** generally follows this structure:
username@domain.extension

Part	Description	Example
username	Contains letters, numbers, dots, underscores, or hyphens	john_doe123
@	Separator between user and domain	@
domain	Website or email provider	gmail, yahoo
extension	Ends with .com, .org, .net, etc.	.com

Solved Example:

```
import re

# Ask user for input
email = input("Enter your email address: ")

# Define regex pattern
pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'

# Validate using re.match()
if re.match(pattern, email):
    print("✅ Valid Email Address!")
else:
    print("❌ Invalid Email Address!")
```

Breaking Down the Pattern

Part	Meaning
^	Start of the string
[\w\.-]+	One or more word characters, dots, or hyphens (for username)
@	The @ symbol — mandatory
[\w\.-]+	One or more characters for the domain
\.	A literal dot before the extension
\w+	One or more letters for the domain extension (com, org, etc.)
$	End of the string

Test Cases

Input	Result
test@example.com	✅ Valid
john_doe@mail.co.in	✅ Valid
abc@.com	❌ Invalid (no domain name)
@gmail.com	❌ Invalid (no username)
test.gmail.com	❌ Invalid (missing @)

Validating Password using Regular Expression

Why validate passwords?

To ensure **strong and secure passwords**, you must check that they meet certain rules, such as:

- Minimum length
- Contains at least one uppercase letter
- Contains at least one lowercase letter
- Contains at least one number
- Contains at least one special character

Validating Passwords and email address using Regular Expression

Common Password Rules Example

Rule	Description
At least 8 characters long	Prevent short, weak passwords
At least one uppercase letter	Example: A-Z
At least one lowercase letter	Example: a-z
At least one digit	Example: 0-9
At least one special character	Example: @, $, #, !, %

Python Code Example

```
import re

# Ask user for password
password = input("Enter your password: ")

# Define regex pattern for strong password
pattern = r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$'

# Validate using re.match()
if re.match(pattern, password):
    print("✅ Strong Password!")
else:
    print("❌ Weak Password! Must contain:\n"
          "- At least 8 characters\n"
          "- One uppercase letter\n"
          "- One lowercase letter\n"
          "- One number\n"
          "- One special symbol (@$!%*?&)")
```

Validating Passwords and email address using Regular Expression

Breaking Down the Password Pattern

Part	Explanation
^	Start of the string
(?=.*[a-z])	Must contain at least one **lowercase** letter
(?=.*[A-Z])	Must contain at least one **uppercase** letter
(?=.*\d)	Must contain at least one **digit (0-9)**
(?=.*[@$!%*?&])	Must contain at least one **special character**
[A-Za-z\d@$!%*?&]{8,}	Must be at least **8 characters long**, containing only these valid symbols
$	End of the string

Test Cases

Password	Result
Abcd123!	✅ Strong
abcd1234	❌ No uppercase, no special symbol
ABCD1234!	❌ No lowercase letter
Abc!	❌ Too short
Abcdefg1@	✅ Strong

In Simple Words:

- **RegEx** helps us define rules for what's "valid."
- **re.match()** checks if input follows those rules.
- It saves time, avoids manual checking, and ensures user input is **correct and safe**.