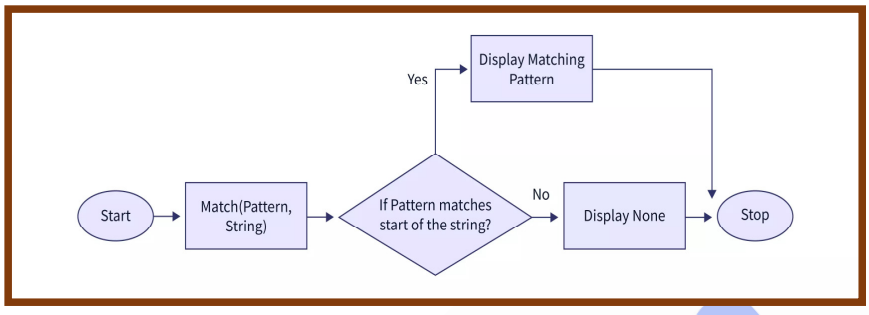
# Regular Expression in Python

## Regular Expressions in Python

In Python, regular expressions (regex) are handled by the built-in re module. Regular expressions are patterns used to match strings or parts of strings. They are incredibly powerful for searching, extracting, replacing, or validating text.




## Why do we use?

- Pattern Matching: To check if a string fits a specific pattern (e.g., email, phone number, URL).
- Searching Text: To find specific words, numbers, or patterns anywhere in a text.
- Extracting Data: To pull out parts of a string, like all digits, dates, or hashtags.
- Replacing Text: To substitute patterns in text (e.g., replace all spaces with underscores).
- Splitting Text: To split strings based on complex rules (not just simple spaces or commas).
- Validation: To ensure input data meets certain rules (e.g., passwords, usernames).
- Efficiency: Handles complex text patterns in one line instead of multiple loops and conditions.
- Automation: Useful in web scraping, log parsing, or data cleaning where patterns repeat.

## What is a Regular Expression (Regex)?

A regular expression (or regex) is a sequence of special characters that defines a search pattern.
It's used to find, match, extract, validate, or replace text in strings.
In Python, regex operations are done using the re module.

## Why do we use Regular Expressions?

- To search for specific patterns in text.
- To validate user input (emails, phone numbers, etc.).
- To extract useful data (numbers, words, dates).

# Regular Expression in Python

- To replace or clean unwanted characters or text.

## Regex Patterns Cheat Sheet

| Symbol | Meaning |
|---|---|
| . | Any character except newline |
| ^ | Start of string |
| $ | End of string |
| * | 0 or more repetitions |
| + | 1 or more repetitions |
| ? | 0 or 1 repetition (optional) |
| {n} | Exactly n repetitions |
| {n,m} | Between n and m repetitions |
| [] | Set of characters |
| \d | Digit [0-9] |
| \D | Non-digit |
| \w | Word character [a-zA-Z0-9_] |
| \W | Non-word character |
| \s | Whitespace |
| \S | Non-whitespace |

## Importing the re Module

**import re**

Once imported, you can use functions like:
- re.match()
- re.search()
- re.findall()
- re.split()
- re.sub()
- re.compile()

# Regular Expression in Python

**1. re.match() - Matches only at the beginning of a string**

```
import re

text = "Hello World"
pattern = r"Hello"

match = re.match(pattern, text)

if match:
    print("Match found:", match.group())
else:
    print("No match.")
```

**Output:**
Match found: Hello

**Explanation:**
- re.match() checks only at the start of the string.
- It will not find "Hello" if it's in the middle of the text.

**2. re.search() - Searches for a match anywhere in the string**

```
import re

text = "Say Hello to Python"
pattern = r"Hello"

result = re.search(pattern, text)
if result:
    print("Found:", result.group())
else:
    print("Not found.")
```

**Output:**
Found: Hello

**Explanation:**
- Unlike match(), search() scans the entire string.

**3. re.findall() - Returns all matches as a list**

```
import re

text = "apple, banana, apple, cherry"
pattern = r"apple"
```

```
matches = re.findall(pattern, text)
print(matches)
```

**Output:**
['apple', 'apple']
• want to count or list all matches.

**Explanation:**
• Finds all occurrences of "apple" in the text.
• Useful when you want to count or list all matches.

**4. re.split() - Splits a string based on a pattern**

```
import re

text = "one, two; three four"
pattern = r"[,; ]+"

result = re.split(pattern, text)
print(result)
```

**Output:**
['one', 'two', 'three', 'four']

**Explanation:**
• The pattern [,; ]+ splits wherever there's a comma, semicolon, or space.
• The + means "one or more occurrences."

**5. re.sub() - Replace matches with something else**

```
import re

text = "I like cats. Cats are cute."
pattern = r"[Cc]ats"

result = re.sub(pattern, "dogs", text)
print(result)
```

**Output:**
I like dogs. dogs are cute.

**Explanation:**
• Replaces "cats" or "Cats" with "dogs".
• Very useful for text cleaning or transformation.