Module 3: Regex (Regular Expressions) in Python

🔍Introduction to Regex

Regex (Regular Expressions) are patterns used to **search, match, and manipulate text**. They help automate text processing tasks.

📌Common Uses

- Find phone numbers
- Detect email addresses
- Validate user input
- Replace or reformat dates
- Extract useful information from large text

🖌️Using Regex in Python

Python provides the `re` **module** for regex operations:

```
import re
```

📌Raw Strings

Use **raw strings** (`r"pattern"`) so Python does not interpret backslashes as escape characters.

🔧Basic Matching

`re.search()` — Find First Match

```
resp = re.search(r'i', 'Hello i am arman ashraf, i live in lahore')
print(resp)
print(resp[0])
```

✔️Finds the **first** occurrence of "i".

🪥Anchors — ^ (start) and $ (end)

```
text = "Arman is a software engineer."
re.search(r'^Arman', text)      # Match at start
re.search(r'^arman', text)      # Case-sensitive
re.search(r'engineer.$', text)  # Match at end
```

✔️Ensures that text **begins** or **ends** with a specific pattern.

🕐Dot `.` — Match Any One Character

```
re.search(r'a.b', 'anb')
re.search(r'a.b', 'Anb', re.IGNORECASE)
```

✔️Matches: **a + ANY character + b**.

🧱Character Classes `[]`

Character classes match **any one character** inside them.

```
re.search(r'[aeiou]', 'hello')
```

✔️Matches any vowel.

```
re.search(r'[^aeiou]', 'ai is a vowel')
```

✔️ `^` inside brackets = **NOT** a vowel.

🔓 OR Operator `|`

```
re.search(r'cat|dog', 'i have a cat')
```

✔️Matches either **"cat"** or **"dog"**.

Quantifiers

Quantifiers decide **how many times** a pattern repeats.

`*` — Zero or More

```
re.search(r'ab*c', 'ac')
re.search(r'ab.*c', 'abc')
```

`?` — Zero or One

```
re.search(r'ab?c', 'ac')
re.search(r'ab?c', 'abc')
```

`+` — One or More

```
re.search(r'ab+c', 'ac')        # No match
re.search(r'b+c+', 'abcccc')    # Match
```

🕑Matching Literals

To match special characters like `.`, escape them:

```
re.search(r'a\.b', 'a.b')
```

🕑Useful Shortcuts

`\w` — Word characters (letters, digits, underscore)

```
re.search(r'\w+', 'Hello_123!')
```

`\d` — Digits

```
re.search(r'\d+', 'My number is 12345')
```

🧰Groups `( )`

```
re.search(r'(ab)+', 'abababxyz')
```

✔️Matches repeated sequences like **ab ab ab**.

✂ Splitting Text with Regex

Split on multiple spaces

```
re.split(r'\s+', 'Hello   world  this is   regex')
```

Split on letters t, h, i, s

```
re.split(r'[this]', 'Hello world this is regex')
```

Keep matched characters in result

```
re.split(r'([Hthis])', 'Hello   world  this is   regex')
```

📘Summary Definitions

Regex

A pattern used to match or manipulate text.

Character Class

A set of characters to match from.

Quantifiers

Symbols that define how many times a pattern repeats (*, +, ?).

Groups

Used to capture or repeat parts of a pattern.

Anchors

Used to match the **start (^)** or **end ($)** of text.