Day 2 – Regex & Text Handling

Objective: Understand regex deeply, master text handling in Python, and prepare for real-world NLP preprocessing.

1. Introduction to Regex (Regular Expressions)

Regex is a sequence of characters forming a search pattern. It's used to search, match, or manipulate text. Think of it as a microscope for strings, helping you filter and extract specific patterns.

```
Example: import re pattern = r"\d{3}-\d{2}-\d{4}" text = "My SSN is 123-45-6789."
print(re.findall(pattern, text)) # Output: ['123-45-6789']
```

2. Regex Building Blocks

2.1. Characters

Pattern	**Meaning**
a	Literal character
.	Any character except newline
\d	Digit (0-9)
\w	Word character (letters, digits, underscore)
\s	Whitespace
\D, \W, \S	Negated versions

2.2. Quantifiers

Pattern	**Meaning**
+	One or more
*	Zero or more
?	Zero or one
{n}	Exactly n times
{n,}	n or more times
{n,m}	Between n and m times

2.3. Anchors

Pattern	**Meaning**
^	Start of string
$	End of string
\b	Word boundary
\B	Not a word boundary

2.4. Groups and Alternation

Pattern	**Meaning**
(abc)	Capturing group
(?:abc)	Non-capturing group
a|b	OR

3. Text Handling in Python

Function	**Purpose**
re.findall()	Returns all matches
re.search()	Returns first match object
re.match()	Match at start
re.split()	Split by pattern
re.sub()	Replace

```
import re text = "Email me at hello@example.com or world@test.org" emails =
re.findall(r"[\w\.-]+@[\w\.-]+\.\w+", text) print(emails) # ['hello@example.com',
'world@test.org']
```

4. Real-World Regex Patterns

- Email – [\w\.-]+@[\w\.-]+\.\w+
- Phone number – \+?\d[\d -]{8,12}\d
- URL – https?://[^\s]+
- Date (YYYY-MM-DD) – \d{4}-\d{2}-\d{2}
- Hashtags – #\w+
- HTML tag – <[^>]+>

5. Practice Exercises

- Match all capitalized words in a sentence.
- Extract all emails from text.
- Extract all URLs from HTML content.
- Extract all hashtags from a tweet.
- Extract all numbers from a paragraph.
- Validate if a string is a phone number.
- Find all words starting with ‘a’.
- Replace multiple spaces with a single space.
- Remove all punctuation from text.
- Split a paragraph into sentences.
- Find all dates in format DD/MM/YYYY.
- Extract domain name from an email.
- Replace all digits with #.
- Find duplicate words in a sentence.
- Extract all words ending with ‘ing’.

6. Mini Project – Data Extraction Pipeline

```
Task: Extract and clean structured information from messy text. Example Input:
John's email is john.doe99@mail.com, contact: +1-202-555-0143. Meeting is
scheduled for 2025-08-09 at https://zoom.us/j/123456789. Expected Output: {
"emails": ["john.doe99@mail.com"], "phones": ["+1-202-555-0143"], "dates":
["2025-08-09"], "urls": ["https://zoom.us/j/123456789"] }
```