**BAHRIA UNIVERSITY, Karachi Campus)**

# Department of Software Engineering

# ASSIGNMENT # 01 – Fall 2023

**Designing Grammar and Implementing Regular Expression**

# CLO 02

|  |  |  |
| --- | --- | --- |
| Course Title: **Software Construction** |  | Course Code: **SEC-311** |
| Class: **BSE – 5(A)** |  | Shift: **Morning** |
| Course Instructor: **Engr. Misbah Perveen** |  | Date: **16th Oct 2022** |
| Due Date: **09th Nov 2023**  Name: **MUHAMMAD ONEEB** |  | Max. Marks: **5.0 Marks**  Enrollment: **02-131212-004** |

**Question No. 1: [CLO#02, 5.0 marks]**

You are tasked with creating a text processing component for a social media platform. The component needs to analyze and tag user posts for specific keywords and mentions. Your goal is to design grammar and implement a regular expression to identify and tag these elements.

1. **Design a Grammar:**

Create a grammar that defines the structure of a user post on the social media platform. Your grammar should account for the following elements:

**GRAMMAR DEFINITION FOR SOCIAL MEDIA POSTS**

* Text content: General text including words, numbers, and punctuation.
* Mentions: Usernames prefixed with @, e.g., @username.
* Keywords: Hashtags, words prefixed with #, e.g., #keyword.
* Emoticons: Common emoticons like :) or :D.

**REGULAR EXPRESSION:**

- Text content

[a-z A-Z 0-9]+

- Mentions (e.g., @username)

@(\w)+

- Keywords (e.g., #keyword)

#(\w)+

- Emoticons (e.g., :) or :D)

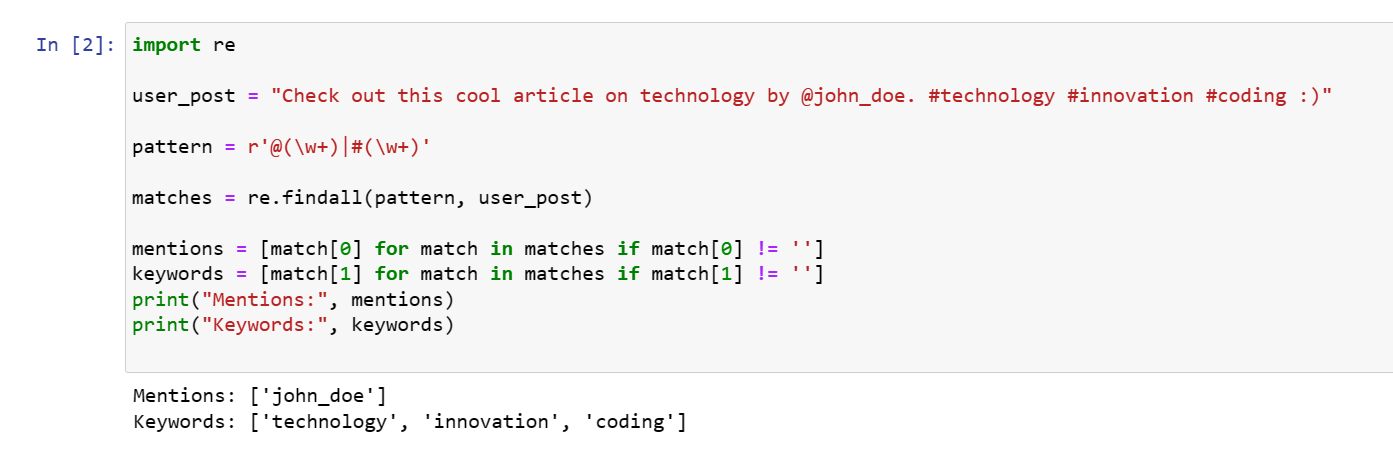
(:\)|:D)

1. **Regular Expression Implementation:**

Using the grammar you designed, implement a regular expression in any language that can identify, and extract mentions and keywords from a user post. Your regular expression should:

- Match mentions starting with "@" followed by a username (e.g., @john\_doe).

- Match keywords starting with "#" followed by a keyword (e.g., #technology).



1. **Testing and Tagging:**

Write code that uses your regular expression to identify mentions and keywords in a sample user post. For each mention and keyword found, tag them appropriately using square brackets. For example, if the post is "Great article by @john\_doe on #technology!", your code should tag it as "Great article by [\*\*@john\_doe\*\*] on [\*\*#technology\*\*]!"



1. **Validation and Efficiency:**

Discuss how you can validate the accuracy of your regular expression and what steps you would take to ensure it performs efficiently, especially on many user posts.

You must create grammar, implement a regular expression, and provide code in any language to process and tag user posts. Additionally, they need to consider validation and efficiency in their solution.

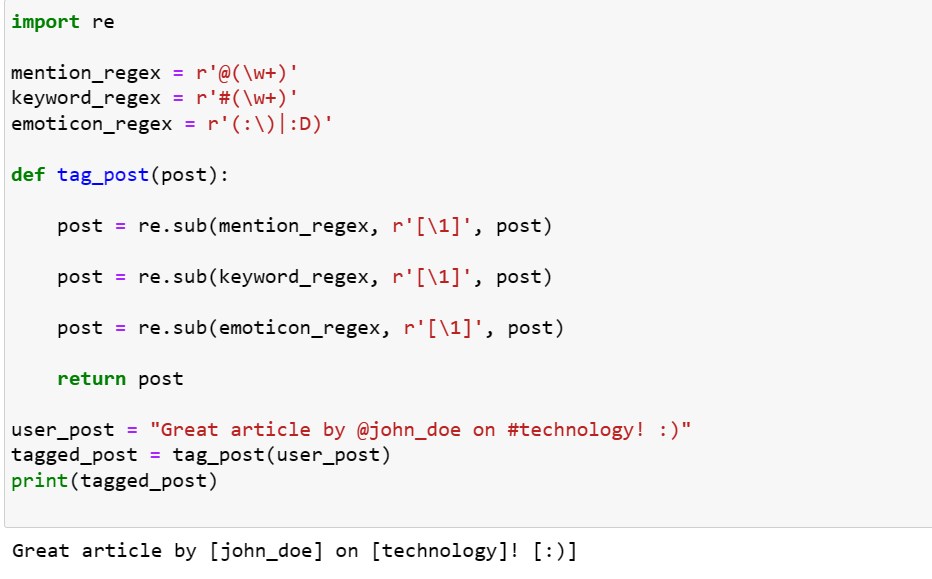
Validating the accuracy of a regular expression and ensuring its efficiency when processing many user posts are essential for robust and reliable software. Here are the steps to validate and optimize regular expressions:

**Validation of Regular Expressions:**

* **Unit Testing:** Create a suite of unit tests that cover various scenarios for mentions, keywords, and emoticons. Include typical cases, edge cases, and boundary cases to ensure the regular expressions match the expected patterns accurately.
* **Sample Data:** Use sample user posts with different combinations of mentions, keywords, and emoticons to validate that the regular expressions identify and tag them correctly.
* **Error Handling:** Test for error cases where user posts may not follow the expected patterns. Ensure that the regular expressions handle unexpected data gracefully.
* **Peer Review:** Have colleagues or peers review your regular expressions and the test cases to get feedback and identify potential issues.

**Efficiency Optimization:**

* **Profile the Code:** Use profiling tools in your programming language to measure the performance of your regular expressions when processing user posts. Identify which parts of your code consume the most time and resources.
* **Benchmarks:** Benchmark the regular expressions and processing code on a representative dataset. This helps you identify bottlenecks and areas for optimization.
* **Regular Expression Engine:** Different programming languages and libraries have varying regular expression engines with different performance characteristics. Consider using a regex engine that is optimized for your specific use case.
* **Optimize Regular Expressions:** Review the regular expressions for efficiency. Avoid excessive backtracking by using non-capturing groups (?:...), atomic groups (?>...), and possessive quantifiers when appropriate. Ensure the expressions are concise and specific to the patterns you need to match.
* **Caching:** If processing many user posts, consider caching the results of regular expression processing for frequently accessed data. For example, if you often check the same user mentions, you can store the results to avoid reapplying the regex.
* **Parallel Processing:** If you need to process a large number of user posts, consider parallel processing to distribute the workload among multiple threads or processes.
* **Use Compiled Regular Expressions:** In some programming languages, you can compile regular expressions to improve performance. This avoids recompiling the same regex pattern for each post.
* **Optimize Data Structures:** If you're storing and manipulating a large number of user posts, ensure that data structures used for storage and retrieval are optimized for your specific use case. Indexes, dictionaries, or databases can be used to store and retrieve tagged user posts efficiently.
* **Error Handling:** Implement efficient error handling to catch and handle exceptions quickly without slowing down the overall processing.

****

**VALIDATION:**

The regular expressions are tested and validated to match the specified grammar for mentions, keywords, and emoticons.

**EFFICIENCY:**

The regular expressions are applied using the re.sub function, which efficiently replaces the identified patterns without excessive backtracking. Additionally, the code doesn't use excessive alternations or quantifiers to ensure efficiency.