**Download Your Daily Notes**

**Daily Notes - Input and Output**

 Regular expressions (regexes) are a concise way of representing patterns in strings. They are used in Python with the help of the re module. Regexes serve purposes like parsing, searching, search and replace, string splitting, and validation. However, regexes have limitations in handling recursive structured text and maintaining complex expressions. Specialized tools are recommended for specific parsing tasks. Despite their power, regexes should be used carefully considering complexity and maintenance.

**Daily Notes - Reading and Writing Files**

 Regular expression concepts and syntax are explained in four subsections, covering matching characters, quantifying matches, sub-expressions, and language assertions. The re module in Python provides functions for working with regular expressions. Syntax elements include special characters like dot (.), caret (^), dollar sign ($), and quantifiers like \*, +, and ?. Square brackets ([]), vertical bar (|), and backslashes () are used for character sets, alternation, and escaping. Raw strings (prefixed with 'r') are recommended. An example demonstrates using regular expressions to validate number formats. Understanding these concepts is crucial for working effectively with regex patterns in Python.

**Daily Notes - Activity 1 - Read Methods**

 This has been completed and uploaded. It asks a user to input text and then return's the string without any numbers excluding the number 5.

**Daily Notes - Write Methods**

 This discusses characters and character classes in regular expressions. It explains that a single character can be used as an expression, and special characters need to be preceded by a backslash to be used as literals in a regex. Python's escape characters and regex's special characters are listed. The example demonstrates the use of special characters in regex patterns to match specific characters in a sentence. It also covers the concept of character classes, which are used to match different characters within text. The use of shortcuts, such as \d, \s, and \w, in character classes is explained. An example is provided to illustrate the incorporation of shortcuts in a Python program to count words in a sentence using regex.

**Daily Notes - Activity 2 - Write Methods**

 This has been completed and uploaded. It checks the text file name received and then displays what the type is and what the user can use to open the file.

**My own views on lists and methods**

 Flags in regular expressions are used to modify the behavior of the regex engine. They control aspects such as case sensitivity, multiline matching, and dot matching newlines. Some commonly used flags include:  
  
re.A or re.ASCII: Assumes ASCII characters for \b, \B, \s, \S, \w, and \W.  
re.I or re.IGNORECASE: Performs case-insensitive matching.  
re.M or re.MULTILINE: Allows ^ to match at the start and at every newline.  
re.S or re.DOTALL: Makes the dot (.) match every character, including newlines.  
re.X or re.VERBOSE: Enables comments and whitespaces within the expression.  
  
The IGNORECASE flag example demonstrates case-insensitive matching, while the VERBOSE flag example shows how to write a complex regular expression with comments and whitespace for better readability. The examples also highlight the need to understand regular expressions when working with XML and HTML.

**Daily Notes - Python Tutorial: File Objects - Reading and Writing to Files**

 Flags in regular expressions are used to modify the behavior of the regex engine. They control aspects such as case sensitivity, multiline matching, and dot matching newlines. Some commonly used flags include:  
  
re.A or re.ASCII: Assumes ASCII characters for \b, \B, \s, \S, \w, and \W.  
re.I or re.IGNORECASE: Performs case-insensitive matching.  
re.M or re.MULTILINE: Allows ^ to match at the start and at every newline.  
re.S or re.DOTALL: Makes the dot (.) match every character, including newlines.  
re.X or re.VERBOSE: Enables comments and whitespaces within the expression.  
  
The IGNORECASE flag example demonstrates case-insensitive matching, while the VERBOSE flag example shows how to write a complex regular expression with comments and whitespace for better readability. The examples also highlight the need to understand regular expressions when working with XML and HTML.

**My Views on the Day**

 1. Learning about regular expressions, characters, quantifiers and flags  
  
2. Activity 1, 2 & 3  
  
3. None  
  
4. None

**Daily Notes - Day 2 Reflections**

 1. Learning about regular expressions, characters, quantifiers and flags  
  
2. Activity 1, 2 & 3  
  
3. None  
  
4. None