Vidya Vikas Education Trust's
Universal College of Engineering, Kaman Road, Vasai-401208
Accredited B+ Grade by NAAC

Experiment 4

Name of the Student: - Sanket Belekar
Roll No. : 67 Batch: A3 Branch: TE COMPS-A
Date of Practical Performed: - Staff Signature with Date & Marks

Aim: Write a program to perform Regular Expression/Perform Morphological Analysis & word generation for any given text.

Theory: A regular expression (RE) is **a language for specifying text search strings**. RE helps us to match or find other strings or sets of strings, using a specialized syntax held in a pattern. Regular expressions are used to search texts in UNIX as well as in MS WORD in identical way.

Function Description

Findall: Returns a list containing all matches
Search: Returns a Match object if there is a match anywhere in the string
Split: Returns a list where the string has been split at each match
Sub: Replaces one or many matches with a string

Regular Expression:

RE is a string that defines a text matching pattern. In NLP, RE is used to find strings having certain patterns in given text. A regular expression is built up using defining rules.

Simple operation in Regular Expressions:
Kleene Closure: If E is a regular expression, then E* is a regular expression
Positive Closure: If E is a regular expression, then E+ is a regular expression
or: If E1and E2are regular expressions, then E1 | E2 is a regular expression
concatenation: If E1and E2are regular expressions, then E1E2 is a regular expression.

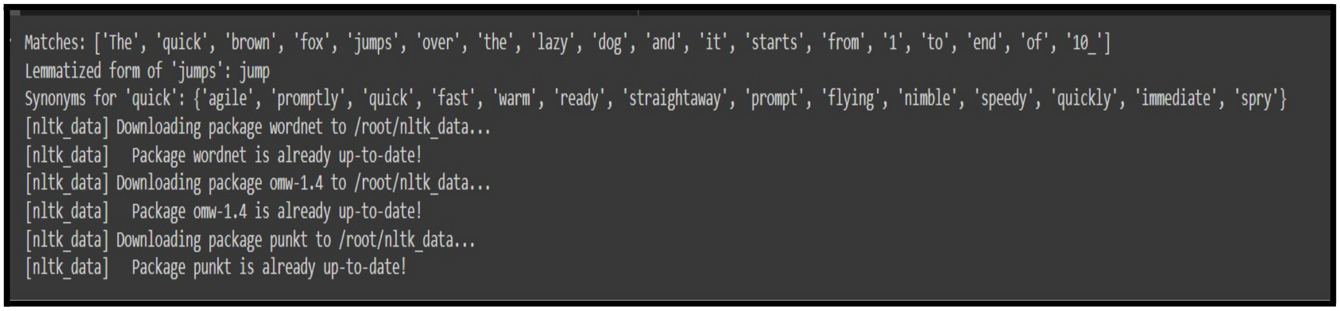
Some ways to represent REs are as follows:

Name	Regular Expression	Matched Strings
Disjunction of characters	[wW]oodchuck	woodchuck, Woodchuck
Range	[A-Z]	All uppercase letters
Disjunction negation	[^sS]	Except "s" and "S"
Disjunction-operator(pipe symbol)	fl(y|ies)	"fly" or "flies"
Kleene Closure	ba*	b, ba, baa, …
Positive Closure	ba+	ba, baa, baaa, …
Wildcard Expression (.)	beg.n	begin, begun, began, etc.

Code:

```
import re
import nltk

# Download the necessary NLTK data files
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer

#OMW is a open multilingual wordnet :-it is multilingual lexical database that extends the WordNet database to multiple languages.
#It provides translations and mappings between different languages' synsets (sets of synonyms) and the English WordNet synsets.
nltk.download('punkt')


def perform_regex_operations(text, pattern):
    """
    Perform regular expression operations on the given text.
    """
    matches = re.findall(pattern, text)
    return matches


def perform_morphological_analysis(word):
    """
    Perform simple morphological analysis (lemmatization) on the given word.
    """
    lemmatizer = WordNetLemmatizer()
    lemma = lemmatizer.lemmatize(word)
    return lemma


def generate_synonyms(word):
    """
    Generate synonyms for the given word using WordNet.
    """
    synonyms = []          #initializes an empty list named synonyms that will be used to collect
synonyms for the given word.
    for syn in wordnet.synsets(word):
        for lemma in syn.lemmas():
            synonyms.append(lemma.name())
    return set(synonyms)


# Example usage
if __name__== "__main__":
    # Sample text
    text = "The quick brown fox jumps over the lazy dog and it starts from 1 to end of 10_"
    # Regular Expression Pattern (find all words)
     pattern = r'\b\w+\b'             # In regex, \w matches any alphanumeric character (letters and
digits) and underscores (_).
     # Word boundary anchor. This asserts a position where a word starts or ends. It ensures that
the match occurs at the boundary of a word.


    # Perform regex operations
    matches = perform_regex_operations(text, pattern)
    print("Matches:", matches)


    # Perform morphological analysis
    word = 'jumps' #"scouts" #'Jumps','boys,','girls','runs','meaningfull'
    lemma = perform_morphological_analysis(word)
    print(f"Lemmatized form of '{word}': {lemma}")


    # Generate synonyms
    word = "quick"
    synonyms = generate_synonyms(word)
    print(f"Synonyms for '{word}': {synonyms}")
```

Output



```
Matches: ['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', 'and', 'it', 'starts', 'from', '1', 'to', 'end', 'of', '10_']
Lemmatized form of 'jumps': jump
Synonyms for 'quick': {'agile', 'promptly', 'quick', 'fast', 'warm', 'ready', 'straightaway', 'prompt', 'flying', 'nimble', 'speedy', 'quickly', 'immediate', 'spry'}
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Conclusion:

Thus, we have successfully studied and performed a program to perform Regular Expression/Perform Morphological Analysis & word generation for any given text.