NAME: Stuti Sharma

ENROLLMENT NUMBER: 20103043

BATCH: B2

Python MCQ Generator

This Python script generates Multiple Choice Questions (MCQs) based on a dataset stored in a CSV file. It uses the pandas library to read the data from the CSV file and generates random variations of the questions and options to create a question paper with shuffled MCQs.

USAGE:

- **IMPORT REQUIRED LIBRARIES:**

```
import pandas as pd

import random

import re

import string

import nltk
```

- **LOAD THE DATASET:**

```
data = pd.read_csv("finally_final_spyder.csv")
```

- **EXTRACT COLUMNS FROM THE DATASET:**

```
question = data.loc[:, "Question"]

answer = data.loc[:, "Answer"]

distractor1 = data.loc[:, "Distractor 1"]

distractor2 = data.loc[:, "Distractor 2"]

distractor3 = data.loc[:, "Distractor 3"]

distractor4 = data.loc[:, "Distractor 4"]

dl = data.loc[:, "Difficulty"]
```

- **DEFINE THE QUESTION_PAPER FUNCTION TO PRINT A SET OF MCQS:**

```
def question_paper(num):
  for i in range(0, num):
    # Generate and print the shuffled MCQs
```

- **REGEX EXPRESSIONS USED TO MATCH PATTERNS AND APPLY RULES:**

r":(\d)"

pattern1: This pattern is used to find and replace numbers that appear after colons (':') in the text.

r"'(?!%s)(.*?)'",

r'"(?!%s)(.*?)"'

pattern2: This pattern is used to find and replace single-quoted or double-quoted strings in the text, excluding certain characters.

r"[A-Z]"

pattern3: This pattern is used to find uppercase letters in the text.

r"[a-z]"

pattern4: This pattern is used to find lowercase letters in the text.

r"\d"

pattern5: This pattern is used to find digits in the text.

r'\$\d+'

pattern6: This pattern is used to find dollar amounts (e.g., $100) in the text.

r"\d+:\d+",

r"^(\w+)\[(\d+):(\d+):(\d+)\]$"

pattern7: This pattern is used to find strings in the format '<digit1>:<digit2>' or '[word][<digit1>:<digit2>:<digit3>]'.

r"x = (\d+)"

pattern8: This pattern is used to find and replace 'x = <digit>' expressions with random values.

```
r"(\d):",
r'print\((\w+)\[-1:\]\)'
```

pattern9: This pattern is used to find and replace digit colons and 'print' expressions in the text.

```
r"::-(\d+)"
```

pattern10: This pattern is used to find and replace '::-<digit>' expressions with random values.

```
r'\.([a-zA-Z_]+\(\))'
```

pattern11: This pattern is used to find and replace method calls (e.g., '.<method_name>()') in the text.

```
r"\w+\[:\d+\] \+ '\w+' \+ \w+\[\d+:\]"
```

pattern12: This pattern is used to find and replace expressions of the form '<word>[:<digit>] + '<word>' + <word>[<digit>:]'.

```
r'"([^"]+)"\s*\+\s*"([^"]+)"',
r'(".*?"|\'.*?\'|\b\w+\b)\s*\+\s*(\w+)'
```

pattern13: This pattern is used to find and replace string concatenation expressions in the text.

```
r'\.count',
r'"(\w+)"\.count\("(\w+)",(\d+),(\d+)\)'
```

pattern14: This pattern is used to find and replace '.count' method calls on strings and count occurrences of a substring.

```
r',\s?(\d+)'
```

pattern15: This pattern is used to find and replace comma-separated integers in the text.

```
r'len\(\[.*?\]\)'
```

pattern16: This pattern is used to find and replace 'len([<elements>])' expressions in the text.

```
r'len\(\s*\[(.*?)\]\s*\)'
```

pattern17: This pattern is used to find and replace 'len([<elements>])' expressions with modified lists.

```
r'len\((.*?)\)'
```

pattern18: This pattern is used to find and replace 'len(<string>)' expressions with the length of the string.

```
r'print\s+(\w+)'
```

pattern19: This pattern is used to find and replace 'print(<word>)' expressions with <word> enclosed in curly braces.

```
r'print\((\w+)\*(\d+)\)',
r'print\((\w+)\s*\[([+-]?\d+)\]\)',
r'print\((\w+)\[(?:-?\d+)\] \+ \1\[(?:-?\d+)\]\)',
r'print\((\w+)\[(\d+):(\d+)\] \* (\d+)\)'
```

pattern20: This pattern is used to find and replace 'print(<word>*<digit>)' expressions with repetitions of <word>.

- **DEFINE HELPER FUNCTIONS TO MODIFY THE VALUES OF MCQS RANDOMLY:**

```
def modify_values(text, dl, d1, d2, d3, d4, a1):
   # Modify the values of the MCQ text, distractors, and answer based on specific rules
```

Define helper functions to generate random values for MCQs:

```
def get_random1():
   # Generate a random integer value between 1 and 1000
def get_random2():
   # Generate a random uppercase or lowercase letter
def get_random3():
   # Generate a random lowercase letter
def get_random4():
   # Generate a random integer value between 2 and 5
def get_random5():
   # Return a fixed value "Rs"
def get_random6(c):
    # Generate a random integer value between 1 and (c-1)
def get_random7():
    # Generate a random word from the NLTK words corpus
def get_random8():
    # Return a random string from a list of predefined options
```

- **DEFINE THE NEW_MCQS FUNCTION TO GENERATE NEW MCQS:**

```
def new_mcqs(q1, dl, d11, d12, d13, d14, a1):
    # Generate new question, distractors, and answer by modifying the values
```

- **LOOP THROUGH THE MCQS AND GENERATE A QUESTION PAPER:**

```
for i in range(0, n):
    new_q1, dl, new_d11, new_d12, new_d13, new_d14, new_a1 = new_mcqs(q1, dl, d11,
d12, d13, d14, a1)
    print("Question: ", new_q1)
    print("Difficulty Level: ", dl)
    print("A. ", new_d11)
    print("B. ", new_d12)
    print("C. ", new_d13)
    print("D. ", new_d14)
    print("ANS. ", new_a1)
```

DEPENDENCIES

- **PANDAS:** To read and manipulate the CSV data
- **NLTK:** To access the words corpus for generating random words