First Semester – Assignment (2)

DMC-6102: Python Programming

Date: 17-Apr-23

Register Number: 2232MCA0058

Question 1

Syntax used in this program:

The major syntax used in the given program are:

1. import json
2. def paragraph_stats(paragraph)
3. vowels = consonants = digits = symbols = words = 0
4. for char in paragraph.
5. if char.isalpha():
6. if char in 'AEIOUaeiou':
7. else: consonants += 1
8. elif char.isdigit(): digits += 1
9. elif char.isspace(): words += 1:
10. else: symbols += 1:
11. words += 1
12. sentences = paragraph.count('.') + paragraph.count('!') + paragraph.count('?')
13. freq_dict = {}
14. for word in paragraph.lower().split():
15. word = ''.join(char for char in word if char.isalpha()):
16. if word: freq_dict[word] = freq_dict.get(word, 0) + 1:
17. palindrome = 0:
18. for word in freq_dict:
19. if word == word[::-1]: palindrome += 1:
20. anagrams = 0:
21. words_list = [word for word in freq_dict if len(word) > 1]:
22. for i in range(len(words_list)):
23. for j in range(i+1, len(words_list)):
24. if sorted(words_list[i]) == sorted(words_list[j]): anagrams += 1:
25. return {...}:
26. stats = paragraph_stats(paragraph)

Problem statement:

Write a function paragraph_stats(paragraph) which performs on the below paragraph: paragraph = "There is no strife, no prejudice, no national conflict in outer space as yet. Its hazards are hostile to us all. Its conquest deserves the best of all mankind, and its opportunity for peaceful cooperation many never come again. But why, some say, the moon? Why choose this as our goal? And they may well ask why climb the highest mountain? Why, 35 years ago, fly the Atlantic? Why does Rice play Texas? We choose to go to the moon. We choose to go to the moon in this decade and do the other things, not because they are easy, but because they are hard, because that goal will serve to organise and measure the best of our energies and skills, because that challenge is one that we are willing to accept, one we are unwilling to postpone, and one which we intend to win, and the others, too. -- John F Kennedy"

Determines the following:

- vowels - number of vowels in the paragraph
- consonants - number of consonants in the paragraph
- words - number of words in the paragraph
- sentences - number of sentences in the paragraph
- digits - number of digits in the paragraph
- symbols - number of special symbols or characters in the paragraph
- frequency - frequency of each distinct word in the paragraph
- palindrome - number of palindrome words in the paragraph
- anagrams - number of anagrams in the paragraph (anagrams are different words formed with same alphabets)

Solution:

```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 25 13:23:48 2023

@author: Haryish Elangumaran
"""
import json #json format on the output is to be generated to promote

def paragraph_stats(paragraph):
    # Counting vowels, consonants, digits, symbols, and words
    vowels = consonants = digits = symbols = words = 0
    for char in paragraph:
        if char.isalpha():
            if char in 'AEIOUaeiou':
                vowels += 1
            else:
                consonants += 1
        elif char.isdigit():
            digits += 1
        elif char.isspace():
            words += 1
        else:
            symbols += 1
    words += 1
    # Counting sentences
    sentences = paragraph.count('.') + paragraph.count('!') +
paragraph.count('?')

    # Counting frequency of each distinct word
    freq_dict = {}
    for word in paragraph.lower().split():
        word = ''.join(char for char in word if char.isalpha())
        if word:
            freq_dict[word] = freq_dict.get(word, 0) + 1

    # Counting palindrome words
    palindrome = 0
    for word in freq_dict:
        if word == word[::-1]:
            palindrome += 1

    # Counting anagrams
    anagrams = 0
    words_list = [word for word in freq_dict if len(word) > 1]
    for i in range(len(words_list)):
        for j in range(i+1, len(words_list)):
            if sorted(words_list[i]) == sorted(words_list[j]):
                anagrams += 1

    return {'vowels': vowels, 'consonants': consonants, 'digits':
digits, 'symbols': symbols, 'words': words,
            'sentences': sentences, 'frequency': freq_dict,
'palindrome': palindrome, 'anagrams': anagrams}

def print_paragraph_stats(dictionary, level=0):
    print("The below entites are the analysis of a given paragraph,")
    i=0
    for i, (key,value) in enumerate(dictionary.items()):
        print("{}.) {}: {}".format(i+1, key,value))

paragraph = "There is no strife, no prejudice, no national conflict in
outer space as yet. Its hazards are hostile to us all. Its conquest
deserves the best of all mankind, and its opportunity for peaceful
cooperation many never come again. But why, some say, the moon? Why
choose this as our goal? And they may well ask why climb the highest
mountain? Why, 35 years ago, fly the Atlantic? Why does Rice play Texas?
We choose to go to the moon. We choose to go to the moon in this decade
and do the other things, not because they are easy, but because they are
hard, because that goal will serve to organise and measure the best of
our energies and skills, because that challenge is one that we are
willing to accept, one we are unwilling to postpone, and one which we
intend to win, and the others, too. -- John F Kennedy"

stats = paragraph_stats(paragraph)

print_paragraph_stats(stats)
#print("Raw data:- \n {}".format(stats))
```

Possible Outputs:

From the Code:

```
The below entites are the analysis of a given paragraph,
1.) vowels: 250
2.) consonants: 378
3.) digits: 2
4.) symbols: 27
5.) words: 153
6.) sentences: 10
7.) frequency: {'there': 1, 'is': 2, 'no': 3, 'strife': 1, 'prejudice': 1, 'national': 1, 'conflict': 1, 'in': 2,
'outer': 1, 'space': 1, 'as': 2, 'yet': 1, 'its': 3, 'hazards': 1, 'are': 5, 'hostile': 1, 'to': 9, 'us': 1, 'all': 2,
'conquest': 1, 'deserves': 1, 'the': 9, 'best': 2, 'of': 2, 'mankind': 1, 'and': 7, 'opportunity': 1, 'for': 1,
'peaceful': 1, 'cooperation': 1, 'many': 1, 'never': 1, 'come': 1, 'again': 1, 'but': 2, 'why': 5,
'some': 1, 'say': 1, 'moon': 3, 'choose': 3, 'this': 2, 'our': 2, 'goal': 2, 'they': 3, 'may': 1, 'well': 1, 'ask':
1, 'climb': 1, 'highest': 1, 'mountain': 1, 'years': 1, 'ago': 1, 'fly': 1, 'atlantic': 1, 'does': 1, 'rice': 1,
'play': 1, 'texas': 1, 'we': 5, 'go': 2, 'decade': 1, 'do': 1, 'other': 1, 'things': 1, 'not': 1, 'because': 4,
'easy': 1, 'hard': 1, 'that': 3, 'will': 1, 'serve': 1, 'organise': 1, 'measure': 1, 'energies': 1, 'skills': 1,
'challenge': 1, 'one': 3, 'willing': 1, 'accept': 1, 'unwilling': 1, 'postpone': 1, 'which': 1, 'intend': 1,
'win': 1, 'others': 1, 'too': 1, 'john': 1, 'f': 1, 'kennedy': 1}
8.) palindrome: 1
9.) anagrams: 0
```

Possible Test Inputs and Outputs:

Test Summary	Input	Output
Test on one word	Paragraph=" India"	{'vowels': 3, 'consonants': 2, 'digits': 0, 'symbols': 0, 'words': 1, 'sentences': 0, 'frequency': {'india': 1}, 'palindrome': 0, 'anagrams': 0}
Test on one sentence	paragraph = "To deposit money in a bank for the first time, you need to create an account"	{'vowels': 26, 'consonants': 34, 'digits': 0, 'symbols': 1, 'words': 16, 'sentences': 0, 'frequency': {'to': 2, 'deposit': 1, 'money': 1, 'in': 1, 'a': 1, 'bank': 1, 'for': 1, 'the': 1, 'first': 1, 'time': 1, 'you': 1, 'need': 1, 'create': 1, 'an': 1, 'account': 1}, 'palindrome': 1, 'anagrams': 0}
Test on null text	paragraph = " "	{'vowels': 0, 'consonants': 0, 'digits': 0, 'symbols': 0, 'words': 1, 'sentences': 0, 'frequency': {}, 'palindrome': 0, 'anagrams': 0}
Test on Random special characters and letters(say mathematical formulae)	paragraph = "B(n,p)-->P(X=x)=nC_p.p^x.q^(n-x)"	{'vowels': 0, 'consonants': 14, 'digits': 0, 'symbols': 18, 'words': 1, 'sentences': 2, 'frequency':

		{'bnppxxncppxqnx': 1}, 'palindrome': 0, 'anagrams': 0}

Question 2

Syntax used in this program:

The program uses the following syntax:

1. Function definition:
    a. def rotate_matrix(matrix, rotation): """Function to rotate the matrix by 90, 180, 270, or 360 degrees.""" ...
    b. def print_matrix(matrix): """Function to print the matrix in a user-friendly way.""" ...
2. Matrix declaration:
    a. matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
3. While loop:
    a. while True: ...
4. User menu:
    a. print("Matrix Rotation Menu")
    b. print("1. Rotate by 90 degrees")
    c. print("2. Rotate by 180 degrees")
    d. print("3. Rotate by 270 degrees")
    e. print("4. Rotate by 360 degrees")
    f. print("5. Exit")
5. Input statement:
    a. choice = int(input("Enter your choice: "))
6. Conditional statements:
    a. if choice == 1: ... elif choice == 2: ... elif choice == 3: ... elif choice == 4: ... elif choice == 5: ... else: ...
7. List manipulation:
    a. rotated_matrix.append(row)
    b. rotated_matrix = matrix matrix[i][j] matrix[i][::-1]
    c. row.append(matrix[i][j]) row = []

Problem statement:

Write a Menu driven program using list to perform Matrix Rotation such as 90, 180, 270 & 360.

Solution:

```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 25 13:26:44 2023

@author: Haryish Elangumaran
"""

def rotate_matrix(matrix, rotation):
    """Function to rotate the matrix by 90, 180, 270, or 360 degrees."""
    rows = len(matrix)
    cols = len(matrix[0])
    rotated_matrix = []

    #rotates matrix to 90 degree
    if rotation == 90:
        for j in range(cols):
            row = []
            for i in range(rows - 1, -1, -1):
                row.append(matrix[i][j])
            rotated_matrix.append(row)

    #rotates matrix to 180 degree
    elif rotation == 180:
        for i in range(rows - 1, -1, -1):
            rotated_matrix.append(matrix[i][::-1])

    #rotates matrix to 270 degree
    elif rotation == 270:
        for j in range(cols - 1, -1, -1):
            row = []
            for i in range(rows):
                row.append(matrix[i][j])
            rotated_matrix.append(row)

    #Rotates batrix to 360 degree
    elif rotation == 360:
        rotated_matrix = matrix

    return rotated_matrix

#To print the resultant matrix
def print_matrix(matrix):
    """Function to print the matrix in a user-friendly way."""
    for row in matrix:
        print(row)
    print("\n")

#Prescribing the matrix
matrix = [[1, 2, 3],
          [4, 5, 6],
          [7, 8, 9]]

print("Given Matrix:")
print_matrix(matrix)

#Menu list
print("Matrix Rotation Menu
\n  1. Rotate by 90 degrees
\n  2. Rotate by 180 degrees
\n  3. Rotate by 270 degrees
\n  4. Rotate by 360 degrees
\n  5. Exit\n")

#Processing the options
while True:
    choice = int(input("Enter your choice: "))

    if choice == 1:
        matrix = rotate_matrix(matrix, 90)
        print_matrix(matrix)
    elif choice == 2:
        matrix = rotate_matrix(matrix, 180)
        print_matrix(matrix)
    elif choice == 3:
        matrix = rotate_matrix(matrix, 270)
        print_matrix(matrix)
    elif choice == 4:
        matrix = rotate_matrix(matrix, 360)
        print_matrix(matrix)
    elif choice == 5:
        print("closed")
        break
    else:
        print("Invalid choice. Please try again.")
```

Possible Outputs:

From the Code:

```
Given Matrix:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]


Matrix Rotation Menu
 1. Rotate by 90 degrees
 2. Rotate by 180 degrees
 3. Rotate by 270 degrees
 4. Rotate by 360 degrees
 5. Exit

Enter your choice: 1
[7, 4, 1]
[8, 5, 2]
[9, 6, 3]


Enter your choice: 2
[3, 6, 9]
[2, 5, 8]
[1, 4, 7]


Enter your choice: 3
[9, 8, 7]
[6, 5, 4]
[3, 2, 1]
```

```
Enter your choice: 4
[9, 8, 7]
[6, 5, 4]
[3, 2, 1]


Enter your choice: 5
"closed"
```

Possible Test Inputs and Outputs:

Test Summary	Input	Output
Rotating twice to get back to same	Menu options → 2,2	Enter your choice: 2 [9, 8, 7] [6, 5, 4] [3, 2, 1] Enter your choice: 2 [1, 2, 3] [4, 5, 6] [7, 8, 9]
Exiting directly at first instance	Menu options → 5	"closed"
Rotating 90 and 270 degrees	Menu options → 1,3	Enter your choice: 1 [7, 4, 1] [8, 5, 2] [9, 6, 3] Enter your choice: 3 [1, 2, 3] [4, 5, 6] [7, 8, 9] Enter your choice: 5 closed
Using 360 at first instance	Menu options → 4	Enter your choice: 4 [1, 2, 3] [4, 5, 6] [7, 8, 9]
Invalid input	Menu options → 6	Enter your choice: 6 Invalid choice. Please try again. Enter your choice: 5 closed