Date: 5-sept-2020

21118

Assignment No: 2
Operation on String.

Name: Shubham Chomate
SE-1 (E-1)

Problem Statement:

Write a Python program to compute the following operations on string:

a> To display word with longest length

b> To determine the frequency of occurence of particular character in the string.

c> To check wheather given string is pallindrome or not

d> To display index of first appearance of the substring

e> To count the occurences of each word in given string.

Objectives:

1> To understand concept and operations of strings

2> To understand primitive functions of string data structure in Python.

Outcomes:

1> To implement string operations using list data structure in python.

2> To write menu driven, modular program in python.

3> To implement user defined functions in Python

Hardware Requirements:

Manufacturer: Acer

Model: Swift SF 314 - 55G

Processor: Intel(R) core(TM) i5 - 8265U CPU @ 1.60 GHz 1.80GHz

Installed Memory (RAM): 8.00 GB, (7.85 GB Available)

System Type: 64-bit Operating System, x64-based processor

Pen and Touch: No pen (or) Touch input is available

Software Requirements:

Operating System: Windows 10 . Home Single Language
(Version: 1903)

Python Version: 3.8.5

VS Code (text editor): Version: 1.48

Theory.

Concepts:

- String: Collection of characters in sequential manner
- String operations:

~~add()~~ add

getInput(): to take input from user

getLargestWord(): Finding Largest word in a string

getCharCount(): Counting how many times particular character apper in the string

isPallindrom(): checking if string is pallindrome (or) not

getSubstrIndex(): To get index of first occurance of substring in the given string

getAllWordsCounts(): Counting how many times each word appeared in the string.

- class: class is a blueprint of objet. It provides way to implement various oop concepts (eg. data hiding, abstraction)
- Object: Object is an instance of class. for same class there can be multiple objects.
- oop concepts: data hiding, abstraction.

data hiding: work through function & don't play directly with actual data

Abstraction: showing important details & hiding unnecessary

things.
Also basic knowledge of python language, list data type & operations of list is required.

ADT

ADT string is,
Data object: A list of character whose end is some special character.

For each string s (actually list of characters) following methods are defined:

1> CreateString (python inbuilt string): list of characters ie. string
// Creates list of characters from python inbuilt string

2> DisplayString (s): void
// displays string s (actually list of charactors)

3> GetLargestWord(s): string (actually list of characters)
// finds largest word in list of characters s.

4> GetCharCount (s, c): integer count
// Finds how many times c appears in the string s.

5> Pallindrome (s): boolean value.
// checks if string s is pallindrome / not.

6> GetSubstrIndex(s, Substr): integer index.
// finds index of first occurance of substr in s.

7> GetAllwordsCount(s): integer count
// counts how many times each words ~~app~~ of s appears
// in s

classmate
Date
Page

class Declaration:

- class String is declared with empty list and another variable strlen to keep count of characters in a list.
- the constructor used for class is parameterized. ~~#~~ It can take python inbuilt string as optional argument & converts it to list of characters. Otherworse it creates empty list.
- other methods in the class are as below:

```
class String():
    def __init__(self, s=""):
        self.mystr = []   #empty list to store characters
        self.strlen = 0   #keeps the track of number of
                                characters in the list.

    def ~~add~~ getInput(self):
        #function takes input from user & appends the
        #characters in self.mystr also increment self.strlen

    def getLargestWord(self):
        #function return largest word from self.mystr.
        #return type is String() object.

    def getcharCount(self,c):
        #function counts how many times c apper in
        # self.mystr.
        # returns the count of c as integer value

    def isPallindrome(self):
        #methode check if self.mystr is pallindrome/not
        #returns true/false accordingly.
```

```
def getSubStrIndex(self, substr):
    # function checks for substr in self.mystr.
    # return index of first occurance else returns -1.

def getAllwordsCount(self):
    # function counts how many times each word is
    # occured in self.mystr
    # return 2-D list which contain string(s) objects &
    # their counts

def areStringEquals(self, anotherstr):
    # Function compares self.mystr with anotherstr
    # returns true/false accordingly

def Print(self):
    # function prints characters in self.mystr.
    # returns void.
```

Algorithm for each operation in class:

```
Algorithm getlargestword(string):
{ 1. make two strings largestword & currword.
  2. traverse the string character by character.
  2.1 if deliminator is found :
        2.1.1 if length of currentword is greater than
              largestword. make currentword as
              largest word
  4. return largestword
}

Algorithm getCharCount(string, char):
{ 1. make a counter variable. (let count)
```

```
    2. traverse string character by character
        2.1 if character of string is equal to char. increment
            the counter variable by 1.
    3. return count
}

Algorithm checkPallindrome (string):
{
    1. create a iterator variable to traverse input string (let i)
    2. traverse the string upto middle.
        2.1 if char at i is not equal to character at i
            from end return false
    3. return True // if all matches string is pallindrome
}

Algorithm getSubstringIndex (string, substring):
{
    1. Traverse a string character by character
        1.1. if char of string is equal to first character ~~of~~
             of substring
            1.1.1. compare first k characters ~~of~~ from
                   current character in string with
                   substring (k is length of substring)
            1.1.2 if all character matches
                1.1.2.1. return current character
                         index. ~~in string~~
                  else check for next character of main
                       string
    2. return -1 // as substring is not found in string
}
```

```
Algorithm getAllwordsCount (string)
{
    1. create a 2.D. list, ~~create~~ (let wordcount=[ ])
    2. traverse through word in a string
        2.1 make counter variable cnt=0.
        2.2. check for current word in remaining string
            If word is found
                increment the counter variable &
                    erase the founded word.
        2.3 append the current word & it's count to
            the wordcount list
    3. return wordcount list.
}
```

Analysis of Algorithms.

Algorithm	Time Complexity	Space Complexity
1> getlargestword()	O(n) where n = character count of input string.	
2> getCharCount( )	O(n) where n= length of input string	
3> checkPallindrome()	$O(n/2) \approx O(n)$ ... asymptotic property n = length of input string	
4> getSubstringIndex()	O(n.k) where n=length of main string k=length of substring.	

5) getAllwordsCount()	O(n²) where n = length of input string.	

Test Cases:

Note: '#' is used to represent end of the string.

Test case No.	Input given/ Test case description	Expected output	Actual Output
1)	finding largest word in a string with large number of whitespaces asb . . . mbpgs . . . zasb#.	the string: mbpgs	string: mbpgs#
2)	Finding largest word in a string with special characters. str: @abc &me (xyzw)#	the string: (xyzw)	string: (xyzw)# program doesn't differentiate b/w special characters & latin character except ~~for~~ #
3)	finding count of space in the string: str: a_bc_d_ _ef#	~~output count~~ count of whitespace: 4.	count: 4.
4)	finding count of Tab in the string: str: as ___df___gh#	count of tab: 2	count: 2
5)	finding 'x' character in empty string str: empty string	expected output is 0.	count: 0.

6> checking normal string for pallindrome: str: aba#	pallindrome.	string is pallindrome.
7> checking empty string for pallindrome.	the string is empty.	The string is empty.
8> getting index of substring which is actually present in the string. str: Hi, how are you doing?# substring: are#	expected index: 8	The index of substring is:8
9> getting index of substring which is not present in the string: str: Hi, how are you doing?# substr: hey#	substring is not present	The substring you have entered is not present in given string.
10> getting count of all words in a sentence. str: he he me be he#	he : 3 times be : 1 time me: 1 time	occurrence of each word in given string is: word is: he count : 3 word is : me count: 1 word is : be count : 1
11> getting count of words in empty sentence	The sentence is empty.	The sentence you have entered is empty

classmate
Date
Page

Applications:
The applications of string & it's operations in real world are:
1> Spell checker
2> Spam filters
3> Intrusion Detection System
4> Search engines
5> Plagarism Detection.
7> Bio-informatics: Gene finding algorithms
8> Digital forencies

Conclusion:
At the end of this assignment I am able to implement inbuit python class of string by myself. I have learned to validate input and how to handle different types of string. Also it helped me to develope my logic regarding string operations.

* * *