Group-A
Assignment-2

Shrikrushna S Zirape

DATE 21286

PAGE NO. 1

* Problem statement :-

Write a python program to compute following operations on string.

a) To display the word with longest length.
b) To determine freq of occurance
c) To check whether the string is palindrome or not.
d) To display First index of substring.
e) To Count occurance of each word in string.

* Learning objective :-

a) To learn to write simple python program and execute it.
b) To learn implimentation of string data structure.
c) To learn & impliment class and obiect Concept.

* learning outcome :-
a) He'll able to write program in python & execute it.
b) He will be able to impliment string data structure.
c) He will be able to implimend class & obiect.

21286
2

Software requirement:-

1) Pycharm IDE Community Version 2020
2) OS Windows 10.

Theory:-

In python string are array of byte representing unicode characters a string is collection of one or more character put in a single quote, double quote or triple quote, in python there is no char data type a character is string of length one.

Syntax:-

```
name = "string1"
name = 'string2'
name = '''string3'''
```

DATE

* Algorithm/pseudocode:-

ADT for class String

```
Class String
    Str = ""          # empty string
    getlen(String)    // returns the
    char_freq(string,chr)  length of given string
                      // returns freq of character
    longest_(Str,Str)  // longest string
    ispalindrome(Str)
                      // Check is string is
                         palindrome.
    First_occur (Str)
                      // determine first occurance
                         of substring in string.
    all-occur (Str)
                      // determine all occurances
                         of substring in string.
```

```
1 Start
2 String S
3 S.redd()
4 read(a)
5.1 if a=1
5.1.1      S.longest_length()
5.2 elif a=2
5.2.1      S.char-freq()
5.3 elif a=3
5.3.1      S.ispalindrome()
5.4. elif  a=4
           S. First-occure.()
```

```
5.5. elif a=s
5.5.2       s.all-occur(str)
6.  else
7.      break.
8  end.
```

* getlen (str)
  ~~Count~~

```
1  start
2  Count=0
3  for i in str:
4      Count += 1
5  return Count
6  end
```

* longest_len (str1, str2)

```
1  start
2  n1 = self.getlen(str1)
3  n2 = self.getlen(str2)
4  if (n1 > n2)
5      return str1
6  return str2
7. end
```

* char Freq (str, char) start.

```
2  for i to range len(str)
3      if str[i] = char:
4      Count += 1
5.     return Count.
6      end.
```

21286
(5)

* ispalindrome (str)

```
1    start
2    for i=0 to(getlen (str)/2)
3       if str[i] != str[(n-i)-1]
4          return not_palindrome break.
5    ; return palindrome
6    end.
```

* first occur (string, substring)

```
1  start.
2  occur = []
3  for i=0  to (-getlen(substring) +getlen(str
4                                  -2)
5       if string[i:(i+getlen(substring)]==
6                substring:
7            occure.append(i)
8  return occur[0]
9   end
```

* all_occur (str, substr)

```
1 start
2 len1,len2 = getlen(str), getlen(sub_str)
3 occur=[]
4 for i=0 to (len1-len2+1)
5       if string[i:(i+len2)]==substring:
6            occur.append(i)
7 return occur.
8   end.
```

24286

(6)

Test Case	Description	I/P	expected Output	Actual O/P	Result.
1	1. longest len 2. freq of char 3. pallindrome. 4. First appear. 5. Count of occur 6. end.	abcba	Pallindrome	Pallindrome	Pass.
2		1. abcdef abc	abcdef	abcdef	Pass.

* ~~Comp~~