

* 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 occurrence
- c) To check whether the string is palindrome or not.
- d) To display First index of substring.
- e) To Count occurrence of each word in string.

* Learning objective :-

- a) To learn to write simple python program and execute it.
- b) To learn implementation of string data structure.
- c) To learn & implement class and object concept.

* Learning outcome :-

- a) He'll be able to write program in python & execute it.
- b) He will be able to implement string data structure.
- c) He will be able to implement class & object.

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"""
```


* Algorithm/pseudo code:-

ADT For class String

Class String

```

str = ""           # empty string
getlen(str)        // returns the
char-freq(str, 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 occurrence
                    of string in string.
all-occur(str)
                    // determine all occurrences
                    of substring in string.
  
```

```

1 Start
2 String s
3 s.redd()
4 read(a)
  s.1 if a=1
    s.1.1 s.longest-length()
  s.2 elif a=2
    s.2.1 s.char-freq()
  s.3 elif a=3
    s.3.1 s.ispalindrome()
  s.4. elif a=4
    s.4.1 s.first-occur()
  
```

6.5. elif a = s

6.5.2 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)

2 for i in range len(str)

3 if str[i] == char:

4 Count += 1

5 return Count

6 end.

* is palindrome (str)

```

1  Start
2  For i = 0 to (getlen(str) / 2)
3      if str[i] != str[getlen(str) - 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          occur.append(i)
8  return occur[0]
9  end

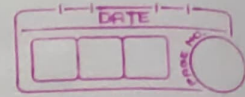
```

* all occur (str, substr)

```

1  Start
2  len1, len2 = getlen(str), getlen(substr)
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

```



Test Case	Description	I/P	expected Output	Actual O/P	Result
-----------	-------------	-----	-----------------	------------	--------

1	1. longest len 2. freq of char. 3. palindrome. 4. First upper. 5. Count of Occur 6. end.	abcba	Palindrome	Palindrome	Pass.
---	---	-------	------------	------------	-------

2	1. abcdef abc	abcdef	abcedf	Pass.
---	---------------------	--------	--------	-------

* ~~Comp~~