**Experiment No. 1**

**Title: Implementation of string handling**

**Batch: B1 Roll No: 1914078 Experiment No.:1**

### Aim: To implement program for string handling

**Resources needed:** Python IDE

### Theory:

Besides numbers, Python can also manipulate strings, which can be expressed in several ways. They can be enclosed in single quotes ('...') or double quotes ("...") with the same result . \ can be used to escape quotes:

'spam eggs' *# single quotes*

'spam eggs'

'doesn**\'**t' *# use \' to escape the single quote...*

"doesn't"

"doesn't" *# ...or use double quotes instead*

"doesn't"

'"Isn**\'**t," she said.'

'"Isn\'t," she said.'

String literals can span multiple lines. One way is using triple-quotes: """...""" or '''...'''. End of lines are automatically included in the string, but it’s possible to prevent this by adding a \ at the end of the line. The following example:

print("""**\**

Usage: thingy [OPTIONS]

-h Display this usage message

-H hostname Hostname to connect to

""")

produces the following output (note that the initial newline is not included):

Usage: thingy [OPTIONS]

-h Display this usage message

-H hostname Hostname to connect to

Strings can be concatenated (glued together) with the + operator, and repeated with \*:

*#3 times 'un', followed by 'ium'*

3 \* 'un' + 'ium'

'unununium'

Strings can be *indexed* (subscripted), with the first character having index 0. There is no separate character type; a character is simply a string of size one. In addition to indexing, *slicing* is also supported. While indexing is used to obtain individual characters, *slicing* allows you to obtain substring:

word = 'Python'

word[0] *# character in position 0*

'P'

word[3:5] *# slice from 3 to 5-1*

'ho'

'

Python strings cannot be changed — they are [immutable](https://docs.python.org/3/glossary.html#term-immutable). Therefore, assigning to an indexed position in the string results in an error:

word[0] = 'J'

...

TypeError: 'str' object does not support item assignment

If you need a different string, you should create a new one:

'J' + word[1:]

'Jython'

The built-in function [len()](https://docs.python.org/3/library/functions.html" \l "len) returns the length of a string:

s = 'supercalifragilisticexpialidocious'

len(s)

34

## Built-In String Functions In Python.

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Description** | **Example Code** |
| capitalize() | Returns the String with first character capitalized and rest of the characters in lower case. | var = ‘PYTHON’ print (var.capitalize()) *# Python* |
| lower() | Converts all the characters of the String to lowercase. | var = ‘TechBeamers’ print (var.lower()) *# techbeamers* |
| upper() | Converts all the characters of the String to uppercase. | var = ‘TechBeamers’ print (var.upper()) *# TECHBEAMERS* |
| swapcase() | Swaps the case of every character in the String means that lowercase characters are changed to uppercase and vice-versa. | var = ‘TechBeamers’ print (var.swapcase()) *# tECHbEAMERS* |
| title() | Returns the ‘titlecased’ version of String which means that all words start with uppercase and rest of the characters in the words are in lowercase. | var = ‘welcome to Python programming’ print (var.title()) *# Welcome To Python Programming* |
| count( str[,beg [,end]]) | Returns the number of times substring ‘str’ occurs in range [beg, end] if beg and end index are given. If it is not given then substring is searched in whole String. Search is case-sensitive. | var=’TechBeamers’ str=’e’ print (var.count(str)) *# 3* var1=’Eagle Eyes’ print (var1.count(‘e’)) *# 2* var2=’Eagle Eyes’ print (var2.count(‘E’,0,5)) *# 1* |

### Python String Comparison Functions.

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Description** | **Example Code** |
| islower() | Returns ‘True’ if all the characters in the String are in lowercase. If any one character is in uppercase it will return ‘False’. | var=’Python’ print (var.islower()) *# False* var=’python’ print (var.islower()) *# True* |
| isupper() | Returns ‘True’ if all the characters in the String are in uppercase. If any one character is in lowercase it will return ‘False’. | var=’Python’ print (var.isupper()) *# False* var=’PYTHON’ print (var.isupper()) *# True* |
| isdecimal() | Returns ‘True’ if all the characters in String are decimal. If anyone character in the String is of other data-type, it will return ‘False’. Decimal characters are those from Unicode category ‘Nd’. Complete list of ‘Nd’ is present at following link: http://www.fileformat.info/info/unicode/category/Nd/list.htm | num=u’2016′ print (num.isdecimal()) *# True* |
| isdigit() | Returns ‘True’ for any character for which isdecimal() would return ‘True and some characters in ‘No’ category. If there are any characters other than these, it will return ‘False’. Precisely, digits are the characters for which Unicode property includes: Numeric\_Type=Digit or Numeric\_Type=Decimal. For example, superscripts are digits but fractions not. Complete list of ‘No’ is present at following link: http://www.fileformat.info/info/unicode/category/No/list.htm | print (‘2’.isdigit()) *# True* print (‘²’.isdigit()) *# True* |

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Description** | **Examples** |
| isnumeric() | Returns ‘True’ if all the characters of the Unicode String lie in any one of the category ‘Nd’,’No’ and ‘NI’. If there are any characters other than these, it will return ‘False’. Precisely, Numeric characters are those for which Unicode property includes Numeric\_Type=Digit, Numeric\_Type=Decimal or Numeric\_Type=Numeric. Complete list of ‘NI’ is present at following link: http://www.fileformat.info/info/unicode/category/Nl/list.htm | num=u’2016′ print (num.isnumeric()) *# True* num=u’year2016′ print (num.isnumeric()) *# False* |
| isalpha() | Returns ‘True’ if String contains at least one character (non-empty String) and all the characters are alphabetic, ‘False’ otherwise. | print (‘python’.isalpha()) *# True* print (‘python3’.isalpha()) *# False* |
| isalnum() | Returns ‘True’ if String contains at least one character (non-empty String) and all the characters are either alphabetic or decimal digits, ‘False’ otherwise. | print (‘python’.isalnum()) *# True* print (‘python3’.isalnum()) *# True* |

### Activities:

1. Write Python program for removing i-th character from a string
2. WAP that print all words with even length and odd length separately from the given string
3. WAP that accepts string from user and redisplays it by removing vowel sounds from it.(without using inbuilt functions)
4. WAP to find the first appearance of the substring 'not' and 'poor' from a given string, if 'not' follows the 'poor', replace the whole 'not'...'poor' substring with 'good'. Return the resulting string.

### Result: (script and output)

### A)

line = input("Enter a string : ")

i = int(input("Enter element position to be removed : "))

newString = ""

if(0 <= i < len(line)):

    print("The new string is : "+line[:i]+line[i+1:])

elif (-len(line) < i < 0):

    print("The new string is : "+line[:i+len(line)]+line[i+1+len(line):])

else:

    print("Index not found")

### 

### B)

line = input("Enter a string : ")

even = []

odd = []

for i in line.split():

    if(len(i) % 2 == 0):

        even.append(i)

    else:

        odd.append(i)

print("Even length words : ")

for i in even:

    print(i)

print("Odd length words : ")

for i in odd:

    print(i)

### 

### C)

line = input("Enter a string : ")

for i in line:

    if(i not in ["a", "e", "i", "o", "u", "A", "E", "I", "O", "U"]):

        print(i, end="")

### 

### D)

line = input("Enter a string : ").split()

posNot = line.index("not") if "not" in line else -1

posPoor = line.index("poor") if "poor" in line else -1

if(posPoor > posNot):

    print(" ".join(line[: posNot]), "good", " ".join(line[posPoor+1:]))

else:

    print("Invalid String")

### 

Outcomes: Use of Basic Data Structures in Python

### 

### Questions:

### Explore different string functions(ljust(), rjust(), center(), zfill(), lstrip(), rstrip() etc)

rjust() :  The string rjust() method returns a new string of given length after substituting a given character in left side of original string.

ljust(): The string ljust() method returns a new string of given length after substituting a given character in right side of original string.

center(): This function center aligns the string according to the width specified and fills remaining space of line with blank space if ‘ fillchr ‘ argument is not passed.

The lstrip() (left strip) function removes leading whitespace (on the left) in the string. The rstrip() (right strip) function removes the trailing whitespace (on the right).

The zfill() method adds zeros (0) at the beginning of the string, until it reaches the specified length.

**Conclusion:** We implemented programs for string handling operations.

**References:**

* 1. **Reema Thareja , “Python Programming: Using Problem Solving Approach”, Oxford University Press, First Edition 2017, India**
  2. **Sheetal Taneja and Naveen Kumar,” Python Programing: A Modular Approach”, Pearson India, Second Edition 2018, India**