UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 08/02/2021
S03018

Ex. No.: 7

1. String Slicing

Aim

To perform string slicing and display characters from a string as required.

Algorithm

Input: Any string
Output: The required part of the string

Pre-condition: The limits should not exceed the index value range of the string

```
Step 1: Start
Step 2: Define a function for slicing the string. The function accepts the string and the required limits for slicing as the arguments. It checks if the limits are reasonable and if yes, returns the sliced string using the slicing operator. If not, it indicates that the limits are out of range.
Step 3: In the main function, the limits and step size are read as input from the user.
Step 4: The function is called and the return value is printed.
Step 5: End
```

Program

```
# Python program to perform string slicing to display parts of the
string as required.

# to get a string as input from the user
st=input("Enter a string: ")
```

```
l=len(st)

# function definition for slicing
def stslice(s,a,ut=l,lt=0):
  if (lt<=ut and lt>=0 and ut<=len(s)):
    return s[lt:ut:a]
  else:
    print("Invalid limits! Slicing through the entire string...")
    return stslice(st,stz)

# to accept suitable slicing specifications from the user
try:
  stz=int(input("Enter step size: "))
  ll=int(input("Enter lower limit: "))
  ul=int(input("Enter upper limit: "))

  print(stslice(st,stz,ul,ll)) #function call

except:
  print("Only integer values accepted.")
```

Output

Input:

```
I am using string slicing. 3 0 10
```

Output:

```
Enter a string: I am using string slicing.
Enter step size: 3
Enter lower limit: 0
Enter upper limit: 10
Imsg
```

Results / Inferences

Program for performing string slicing to display the characters from a string as required is written and executed.

2. String Palindrome

Aim

To check if a given string is a palindrome or not.

Algorithm

Input: Any string
Output: The conclusion of it being a palindrome or not

Step 1: Start
Step 2: Define a function to check if a given word is a palindrome or not. The function reads a word as input from the user and checks if the word is the same as the reverse of the word. If yes, it prints that it is a palindrome and if not, prints that it is not a palindrome.
Step 3: Call the function to check if a word is a palindrome.
Step 6: End

Program

```
# Python program to check if a given string is a palindrome or not.

# function to reverse the string
def rev(w):
  return w[::-1]

# function definition for the palindrome check
def is_palindrome():
  word=input("Enter a word: ")
  wd=rev(word)
  if wd==word:
    print("Palindrome")
  else:
    print("Not a palindrome")

is_palindrome()#function call
```

Output

```
Input:

malayalam
laptop

Output:

Enter a word: malayalam
Palindrome

Enter a word: laptop
Not a palindrome
```

Results / Inferences

Program for checking if a given string is a palindrome or not is written and executed.

Krithika Swaminathan
Date: 08/02/2021
S03018

Ex. No.: 7

3. Caesar Cypher

Aim

To encrypt a message using the Caesar Cypher.

Algorithm

```
Input: Any message as a string
Output: The encrypted message

Pre-condition: The entered string must consist of letters only.

Step 1: Start
Step 2: Define a function to encrypt the message. The function
accepts a word and the shift value for the Caesar shift as
arguments.
Step 3: The function contains a loop that traverses the entire
word and increases the ASCII value of each character by the
shift value and converts it back to letters. If the ASCII value
after addition exceeds 90 for upper case letters and 122 for
lower case letters, then 26 is subtracted from the value, after
which it is converted back to a letter.
Step 4: The changed letters are stored as a new string.
Step 5: The function returns the new string as the encrypted
message.
Step 6: Read the message and the shift value as input from the
user.
Step 7: Call the function and print the return value.
Step 8: End
```

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 08/02/2021
S03018

Ex. No.: 7

Program

```
# Python program to encrypt a message using the Caesar cypher.

# Function definition for the Caesar cypher
def rotate_word(m,n):
  new=''
  if m.isalpha():
    for i in range(len(m)):
      j=ord(m[i])+n
      if (j>=65 and j<=90) or (j>=97 and j<=122):
        new+=chr(j)
      else:
        j=j-26
        new+=chr(j)
  else:
    print("Invalid character(s)")
  if new.isalpha()==0:
    print("Not all characters in the encrypted string are 
alphabets.")
  return new

# to get the message and specifications from the user
word=input("Enter a word: ")
shift=int(input("Enter shift value: "))
# to display the encrypted message
print(rotate_word(word,shift))
```

Output

```
Input:

zebra 2

Output:
```

```
Enter a word: zebra
Enter shift value: 2
bgdtc
```

Results / Inferences

Program for encrypting a message using the Caesar Cypher is written and executed.

4. Menu for String Operations

Aim

To create a menu-driven program to perform string operations.

Algorithm

Input: Any string and choice
Output: A result based on the user’s choice

Pre-condition: The choice should be valid.

```
Step 1: Start
Step 2: Define a function to create a menu for altering or analysing a string based on the user’s choice. The function accepts the string and the choice as arguments.
Step 3: If the choice is option ‘a’, do the following.
    • Read a string(substring) as input from the user.
    • Use the membership operator to check if the substring is present in the string.
    • If yes, return that it’s found. If not, return that it’s not found.
Step 4: Else, if the choice is option ‘b’, do the following.
    • Read a substring as input from the user.
    • Use the slicing operator to reverse the string.
    • Using the membership operator, check if the substring is present in the reversed string.
    • If yes, return that it’s found. If not, return that it’s not found.
Step 5: Else, if the choice is option ‘c’, do the following.
    • Read the required width of the string and a filler as input from the user.
```

- Use the rjust() function to right justify the string with the width and filler as arguments.
- Return the right justified string.

Step 6: Else, if the choice is option ‘d’, do the following.

- Using the capitalize() function, return the capitalized string.

Step 7: Else, if the choice is option ‘e’, do the following.

- Use the isalnum() function to check if the string is alphanumeric.
- If yes, return that it’s alphanumeric. If not, return that it is not alphanumeric.

Step 8: Read a string and the user’s choice as input from the user.
Step 9: Pass them as the arguments and call the function.
Step 10: End

Program

```
# Python program to create a menu to choose and perform string
operations.
# Function definition for the menu
def menu(s,ch):
  if ch=='a':
    sub=input("Enter the substring: ")
    if sub in s:
      return "Found"
    else:
      return "Not found"

  elif ch=='b':
    sub=input("Enter the substring: ")
    if sub in s[::-1]:
      return "Found"
    else:
      return "Not found"
  elif ch=='c':
    width=int(input("Enter reqd width: "))
    fill=input("Enter filler: ")
    return s.rjust(width,fill)

  elif ch=='d':
    return s.capitalize()

  elif ch=='e':
    if s.isalnum():
      return "Alphanumeric"
    else:
      return "Not alphanumeric"

  else:
    return "Invalid choice!"
# to get the string from the user
s=input("Enter a string: ")
# to display the choices in the menu
print("a.occurence of substring\nb.occurence of substring from the
end\nc.right justify a string\nd.capitalize the first letter of a
string\ne.check if alphanumeric")
ch=input("Choice?: ")
print(menu(s,ch))#function call to call the required menu operation
```

Output

```
Enter a string: The Wonderful Wizard of Oz
a.occurence of substring
b.occurence of substring from the end
c.right justify a string
d.capitalize the first letter of a string
e.check if alphanumeric
Choice?: a
```

```
Enter the substring: Wizard
Found

Enter a string: The Wonderful Wizard of Oz
a.occurence of substring
b.occurence of substring from the end
c.right justify a string
d.capitalize the first letter of a string
e.check if alphanumeric
Choice?: b
Enter the substring: red
Found

Enter a string: the hills are alive
a.occurence of substring
b.occurence of substring from the end
c.right justify a string
d.capitalize the first letter of a string
e.check if alphanumeric
Choice?: c
Enter reqd width: 35
Enter filler: #
################the hills are alive

Enter a string: emperor
a.occurence of substring
b.occurence of substring from the end
c.right justify a string
d.capitalize the first letter of a string
e.check if alphanumeric
Choice?: d
Emperor

Enter a string: Flight627
a.occurence of substring
b.occurence of substring from the end
c.right justify a string
```

```
d.capitalize the first letter of a string
e.check if alphanumeric
Choice?: e
Alphanumeric

Enter a string: Doesn't matter
a.occurence of substring
b.occurence of substring from the end
c.right justify a string
d.capitalize the first letter of a string
e.check if alphanumeric
Choice?: 6f
Invalid choice!
```

Results / Inferences

Program for creating a menu to perform string operations is written and executed.

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 08/02/2021
S03018

Ex. No.: 7

5. Replacing a Character

Aim

To replace all other occurrences except the first of the first character in a string with $.

Algorithm

Input: Any string
Output: The altered string

```
Step 1: Start
Step 2: Read any string as input from the user.
Step 3: Store the first character in a variable.
Step 4: Create an empty new string.
Step 5: Traverse the string, starting from the second character. If the character is the same as the first character, add '$' to the new string. If it isn't, then add the original character to the new string.
Step 6: Print the new string.
Step 7: End
```

Program

```
# Python program to replace every following occurrence of the first character of a string with $.

# to get the string as input from the user
s=input("Enter a string: ")
# to take note of the first character
f=s[0]
new=f
for ch in s[1::]:
  if f is ch: #to check if the character matches the first
    new+='$'
  else:
    new+=ch
# to print the new altered string
print(new)
```

Output

```
Enter a string: peter purchased pumpkins, papayas and
apples
peter $urchased $um$kins, $a$ayas and a$$les
```

Results / Inferences

Program for replacing characters as required is written and executed.

6. Occurrence of a Word in a Sentence

Aim

To count the occurrence of a given word in a given sentence.

Algorithm

Input: Any two numbers, say, m and n
Output: The result of the Ackermann function

Pre-condition: The numbers should be greater than or equal to 0

```
Step 1: Start
Step 2: Read a sentence as input from the user.
Step 3: Read a word as input from the user and introduce a variable to count the words.
Step 4: Split the sentence into words using the split() function and traverse the list of words using a loop.
Step 5: If the word is equal to the input word, add one to the count.
Step 6: Print the occurrence of the required word.
Step 7: End
```

Program

```python
# Python program to count the occurrence of a given word in a given sentence.

# To get the sentence as input from the user
sen=input("Enter a sentence: ")
# To input the word to be counted
word=input("Enter a word: ")
count=0
```

```
for w in sen.split():
  if w==word: #to check if the word matches the given word
    count=count+1
print("Occurrence:",count)
```

Output

```
Enter a sentence: You know that I know that you know that
I know the code to the safe.
Enter a word: know
Occurrence: 4
```

Results / Inferences

Program for counting the occurrence of a given word in a given sentence is written and executed.