Shubham Sharma
2021-22BCSE046

Lab 4: Strings & Functions

1. Write a Python program to calculate the length of a string (using both inbuilt method and without using the inbuilt method).

```
def length_inbuilt_method(s):
    return len(s)
def length_manual_method(s):
    count = 0
    for char in s:
        count += 1
    return count

test_string = "Hello, World!"

print(f"Length of string using inbuilt method: {length_inbuilt_method(test_string)}")

print(f"Length of string without using inbuilt method: {length_manual_method(test_string)}")
```

```
Length of string using inbuilt method: 13
Length of string without using inbuilt method: 13
```

2. Write a Python program to count the number of occurrences of a given character in a string (Using both inbuilt method and without the inbuilt method).

```
def count_inbuilt_method(s, char):
    return s.count(char)

def count_manual_method(s, char):
    count = 0
    for c in s:
        if c == char:
            count += 1
    return count

test_string = "Hello, World!"
char_to_count = 'o'

print(f"Occurrences of '{char_to_count}' using inbuilt method: {count_inbuilt_method(test_string, char_to_count)}")

print(f"Occurrences of '{char_to_count}' without using inbuilt method: {count_manual_method(test_string, char_to_count)}")
```

```
Occurrences of 'o' using inbuilt method: 2
Occurrences of 'o' without using inbuilt method: 2
```

3. Write a Python program to count the number of occurrences of each character of a given string.

```
def count_characters(s):
    char_count = {}

    for char in s:
        if char in char_count:
            char_count[char] += 1
        else:
            char_count[char] = 1

    return char_count


test_string = "Hello, World!"
char_occurrences = count_characters(test_string)

for char, count in char_occurrences.items():
    print(f"'{char}' : {count}")
```

```
H' : 1 'e' : 1 'l' : 3 'o' : 2 ',' : 1 ' ' : 1 'W' : 1 'r' : 1 'd' : 1 '!' : 1 (.venv)
```

4. Given two strings, s1 and s2. Write a program to create a new string s3 by appending s2 in the middle of s1

```
def append_in_middle(s1, s2):
    middle_index = len(s1) // 2

    s3 = s1[:middle_index] + s2 + s1[middle_index:]

    return s3

s1 = "HelloWorld"
s2 = "Python"

s3 = append_in_middle(s1, s2)
print(f"The new string is: {s3}")
```

```
The new string is: HelloPythonWorld
```

5. Count all letters, digits, and special symbols from a given string.

```
def count_elements(s):
    letters = 0
    digits = 0
    special_symbols = 0

    for char in s:
        if char.isalpha():
            letters += 1
        elif char.isdigit():
            digits += 1
        else:
            special_symbols += 1

    return letters, digits, special_symbols

test_string = "Hello123! Welcome @2024."

letters, digits, special_symbols = count_elements(test_string)
print(f"Letters: {letters}")
print(f"Digits: {digits}")
print(f"Special symbols: {special_symbols}")
```

```
Letters: 12
Digits: 7
Special symbols: 5
```

6. Write a program to check if two strings are balanced. For example, strings s1 and s2 are balanced if all the characters in the s1 are present in s2. The character's position doesn't matter.

```
def are_strings_balanced(s1, s2):
    for char in s1:
        if char not in s2:
            return False
    return True

s1 = "abc"
s2 = "cbadef"

if are_strings_balanced(s1, s2):
    print(f"The strings '{s1}' and '{s2}' are balanced.")
else:
    print(f"The strings '{s1}' and '{s2}' are not balanced.")
```

```
The strings 'abc' and 'cbadef' are balanced.
```

7. Find all occurrences of a substring in a given string by ignoring the case

```
def find_all_occurrences(main_string, sub_string):
    main_string_lower = main_string.lower()
    sub_string_lower = sub_string.lower()

    indices = []

    index = main_string_lower.find(sub_string_lower)
    while index != -1:
        indices.append(index)
        index = main_string_lower.find(sub_string_lower, index + 1)

    return indices

main_string = "Hello World, welcome to the world of programming. WORLD is vast!"
sub_string = "world"

occurrences = find_all_occurrences(main_string, sub_string)

if occurrences:
    print(f"Substring '{sub_string}' found at indices: {occurrences}")
else:
    print(f"Substring '{sub_string}' not found.")
```

```
Substring 'world' found at indices: [6, 28, 50]
```

8. Calculate the sum and average of the digits present in a string.

```
def sum_and_average_of_digits(s):
    digits = [int(char) for char in s if char.isdigit()]

    total_sum = sum(digits)
    average = total_sum / len(digits) if digits else 0

    return total_sum, average

test_string = "Hello123! 456"
total_sum, average = sum_and_average_of_digits(test_string)
print(f"Sum of digits: {total_sum}, Average of digits: {average}")
```

```
Sum of digits: 21, Average of digits: 3.5
```

9. WAP to reverse a string using slicing operator

```
def reverse_string_slicing(s):
    return s[::-1]
test_string = "Hello, World!"
reversed_string = reverse_string_slicing(test_string)
print(f"Reversed string using slicing: {reversed_string}")
```

```
Reversed string using slicing: !dlroW ,olleH
```

10. WAP to reverse a string using loops

```
def reverse_string_loop(s):
    reversed_str = ""
    for char in s:
        reversed_str = char + reversed_str
    return reversed_str
test_string = "Hello, World!"
reversed_string = reverse_string_loop(test_string)
print(f"Reversed string using loop: {reversed_string}")
```

```
Reversed string using slicing: !dlroW ,olleH
```

11. Removal all characters from a string except integers

```
def remove_non_integers(s):
    return ''.join(char for char in s if char.isdigit())

test_string = "Hello123! 456"
cleaned_string = remove_non_integers(test_string)
print(f"String with only integers: {cleaned_string}")
```

```
String with only integers: 123456
```

12. Apply all possible operators (+, *, in, is, ==, >) on strings, and print the output.

```
s1 = "Hello"
s2 = "World"

concat = s1 + " " + s2

repeat = s1 * 2

membership = "Hello" in s1

identity = (s1 is s1)

equality = (s1 == s2)

greater_than = (s1 > s2)

print(f"Concatenation: {concat}")
print(f"Repetition: {repeat}")
print(f"Membership: {membership}")
print(f"Identity: {identity}")
print(f"Equality: {equality}")
print(f"Greater than: {greater_than}")
```

```
Concatenation: Hello World
Repetition: HelloHello
Membership: True
Identity: True
Equality: False
Greater than: False
```

13. Create a function with TWO arguments (one of them is the default argument), and call the function.

```
def greet(name, greeting="Hello"):
    return f"{greeting}, {name}!"

print(greet("Alice"))
print(greet("Bob", "Hi"))
```

```
Hello, Alice!
Hi, Bob!
```

14. WAP to reverse a string using Recursion

```python
def reverse_string_recursion(s):
    if len(s) == 0:
        return s
    else:
        return s[-1] + reverse_string_recursion(s[:-1])

test_string = "Hello, World!"
reversed_string = reverse_string_recursion(test_string)
print(f"Reversed string using recursion: {reversed_string}")
```

```
Reversed string using recursion: !dlroW ,olleH
```