String Operations in Python

1. Find the length of the string

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string = "Cryptology"
length = len(string)
print(length)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
10
```

2. Slice the string as per your choice

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string = "Cryptology"
slice1 = string[0:5]
slice2 = string[7:]
slice3 = string[0:5:2]
print(slice1)
print(slice2)
print(slice3)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Crypt
ogy
Cyt
```

3. Concatenate two strings

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string1 = "Crypt"
string2 = "ology"
result = string1 + string2
print(result)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Cryptology
```

4. Convert in to lower case in to uppercase character

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string = "Kruthi"
lowercase = string.lower()
uppercase = string.upper()
print(lowercase)
print(uppercase)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
kruthi
KRUTHI
```

5. Convert upper case into lower case characters

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string = "KRUTHI"
lowercase = string.lower()
print(lowercase)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
kruthi
```

6. convert the character into Unicode (Ascii values)

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string = "Kruthi"
ascii_values = [ord(char) for char in string]
print(ascii_values)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
[75, 114, 117, 116, 104, 105]
```

7. convert Unicode into character

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
ascii_values = [75, 114, 117, 116, 104,105]
characters = ''.join(chr(value) for value in ascii_values)
print(characters)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Kruthi
```

8. Check whether the given "substring" exists in the string

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string = "Kruthi Somalli"
substring = "Somalli"
if substring in string:
    print("Substring exists!")
else:
    print("Substring does not exist.")
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Substring exists!
```

9. Replace the character 'k' with 'h'

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string = "Kitty"
result = string.replace('k', 'h').replace('K', 'h')
print(result)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
hitty
```

10. Pad the string with "x" at the end

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string = "Kruthi"
padded_string = string.ljust(10, 'x')
print(padded_string)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Kruthixxxx
```

11. remove leading and trailing whitespace or specified characters from the string

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string = "    Kruthi    "
trimmed_string = string.strip()
print(trimmed_string)
trimmed_specific = string.strip(" K!")
print(trimmed_specific)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Kruthi
ruthi
```

12. split the given string in to group of five characters

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string = "Kruthi Somalli."
groups = [string[i:i + 5] for i in range(0, len(string), 5)]
print(groups)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
['Kruth', 'i Som', 'alli.']
```

13. count total number of words

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
string = "Kruthi Somalli"
word_count = len(string.split())
print(word_count)
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
2
```

14. Find the frequency of each characters in the string STDIN and File operators

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
from collections import Counter
input_string = input("Enter a string: ")
frequency = Counter(input_string)
print("Character Frequency:")
for char, count in frequency.items():
    print(f"{char}: {count}")
with open('input.txt', 'r') as file:
    file_content = file.read()
    frequency_file = Counter(file_content)
    print("\nFile Character Frequency:")
    for char, count in frequency_file.items():
        print(f"{char}: {count}")
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Enter a string: Kruthi Somalli
Character Frequency:
K: 1
r: 1
u: 1
t: 1
h: 1
i: 2
 : 1
S: 1
o: 1
m: 1
a: 1
l: 2
```

15. get the file name from the user

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
filename = input("Enter the filename (with extension): ")
try:
    with open(filename, 'r') as file:
        content = file.read()
        print("File Content:")
        print(content)
except FileNotFoundError:
    print(f"The file '{filename}' does not exist.")
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Enter the filename (with extension): lab2
The file 'lab2' does not exist.
```

16. check the file exist or not Looping and File handling

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
import os
def check_file_exists(filename):
    """Check if the specified file exists."""
    return os.path.isfile(filename)
def main():
    while True:
        filename = input("Enter the filename (with extension) or 'exit' to quit: ")
        if filename.lower() == 'exit':
            print("Exiting the program.")
            break
        if check_file_exists(filename):
            print(f"The file '{filename}' exists.")
        else:
            print(f"The file '{filename}' does not exist. Please try again.")
if __name__ == "__main__":
    main()
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Enter the filename (with extension) or 'exit' to quit: lab2
The file 'lab2' does not exist. Please try again.
Enter the filename (with extension) or 'exit' to quit:
```

17. read the contents from the file

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
import os
def read_file_contents(filename):
    """Read and print the contents of the specified file."""
    if os.path.isfile(filename):
        with open(filename, 'r') as file:
            content = file.read()
            print("File Contents:")
            print(content)
    else:
        print(f"The file '{filename}' does not exist.")

def main():
    while True:
        filename = input("Enter the filename (with extension) to read or 'exit' to quit: ")
        if filename.lower() == 'exit':
            print("Exiting the program.")
            break
        read_file_contents(filename)
if __name__ == "__main__":
    main()
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Enter the filename (with extension) to read or 'exit' to quit: kru
```

```
kruthi@kali: ~
File  Actions  Edit  View  Help
        fence[row][i] = '*'
        if row == 0:
            step = 1
        elif row == rails - 1:
            step = -1
        row += step

    index = 0
    for r in range(rails):
        for c in range(len(ciphertext)):
            if fence[r][c] == '*' and index < len(ciphertext):
                fence[r][c] = ciphertext[index]
                index += 1

    result = []
    row, step = 0, 1
    for i in range(len(ciphertext)):
        result.append(fence[row][i])
        if row == 0:
            step = 1
        elif row == rails - 1:
            step = -1
        row += step

    return ''.join(result)

ciphertext = "AAIUJ SIHBE KTEAO TEADE SNUTF EAEMR TAHSA RHROA YHNFO AITTE EHCBO FVCAT RNMNS NUTFE RASHL WFHLN HIUJS IHTKS OEHNI FISAE FNTIG RMR
rails = 3
plaintext = rail_fence_decrypt(ciphertext.replace(" ", ""), rails)
print(f"Decrypted message: {plaintext}")

Enter the filename (with extension) to read or 'exit' to quit:
```

Type here to search Links Desktop 25°C

18. reverse the contents from the file

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
import os

def reverse_file_contents(filename):
    """Reverse the contents of the specified file and print them."""
    if os.path.isfile(filename):
        with open(filename, 'r') as file:
            content = file.read()
        reversed_content = content[::-1]
        print("Reversed File Contents:")
        print(reversed_content)
    else:
        print(f"The file '{filename}' does not exist.")

def main():
    while True:
        filename = input("Enter the filename (with extension) to reverse its contents or 'exit' to quit: ")
        if filename.lower() == 'exit':
            print("Exiting the program.")
            break
        reverse_file_contents(filename)
if __name__ == "__main__":
    main()
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Enter the filename (with extension) to reverse its contents or 'exit' to quit: kru
```

```
kruthi@kali: ~
File  Actions  Edit  View  Help
)tluser(nioj.'' nruter

pets =+ wor
1- = pets
:1 - sliar == wor file
1 = pets
:0 == wor fi
)]i[]wor[ecnef(dneppa.tluser
:))txetrehpic(nel(egnar ni i rof
1 ,0 = pets ,wor
][ = tluser


1 =+ xedni
]xedni[txetrehpic = ]c[]r[ecnef
:)txetrehpic(nel < xedni dna '*' == ]c[]r[ecnef fi
:))txetrehpic(nel(egnar ni c rof
:)sliar(egnar ni r rof
0 = xedni


pets =+ wor
1- = pets
:1 - sliar == wor file
1 = pets
:0 == wor fi
'*' = ]i[]wor[ecnef
:))txetrehpic(nel(egnar ni i rof


1 ,0 = pets ,wor
])sliar(egnar ni _ rof ]))txetrehpic(nel(egnar ni _ rof 'n\'[[ = ecnef
:)sliar ,txetrehpic(tpyrced_ecnef_liar fed
Enter the filename (with extension) to reverse its contents or 'exit' to quit:
```

19. Write into the file Math operations

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
def math_operations_results.txt :
    a = 10
    b = 5
    addition = a + b
    subtraction = a - b
    multiplication = a * b
    division = a / b if b ≠ 0 else "Division by zero"
    modulus = a % b
    exponentiation = a ** b
    results = {
        "Addition": addition,
        "Subtraction": subtraction,
        "Multiplication": multiplication,
        "Division": division,
        "Modulus": modulus,
        "Exponentiation": exponentiation,
    }
    return results
def write_results_to_file(results, filename):
    """Write the math operation results to the specified file."""
    with open(filename, 'w') as file:
        file.write("Math Operations Results:\n")
        for operation, result in results.items():
            file.write(f"{operation}: {result}\n")
    print(f"Results written to '{filename}'.")
def main():
    results = perform_math_operations()
    filename = "math_operations_results.txt"
    write_results_to_file(results, filename)
if __name__ == "__main__":
    main()
```

Output:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
Math Operations Results:
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
Modulus: 0
Exponentiation: 100000
```

20. convert Frequency in to percentage (continuation of 12th Question)

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
from collections import Counter
def calculate_character_frequency(string):
    """Calculate the frequency of each character in the string."""
    return Counter(string)
def convert_frequency_to_percentage(frequency):
    """Convert character frequency to percentage."""
    total_characters = sum(frequency.values())
    percentages = {char: (count / total_characters) * 100 for char, count in frequency.items()}
    return percentages
def main():
    string = input("Enter a string to analyze: ")
    frequency = calculate_character_frequency(string)

    percentages = convert_frequency_to_percentage(frequency)
    print("\nCharacter Frequencies and Percentages:")
    for char, count in frequency.items():
        percentage = percentages[char]
        print(f"Character: '{char}' | Frequency: {count} | Percentage: {percentage:.2f}%")
if __name__ == "__main__":
    main()
```

Output:

```
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Enter a string to analyze: Kruhti Somalli

Character Frequencies and Percentages:
Character: 'K' | Frequency: 1 | Percentage: 7.14%
Character: 'r' | Frequency: 1 | Percentage: 7.14%
Character: 'u' | Frequency: 1 | Percentage: 7.14%
Character: 'h' | Frequency: 1 | Percentage: 7.14%
Character: 't' | Frequency: 1 | Percentage: 7.14%
Character: 'i' | Frequency: 2 | Percentage: 14.29%
Character: ' ' | Frequency: 1 | Percentage: 7.14%
Character: 'S' | Frequency: 1 | Percentage: 7.14%
Character: 'o' | Frequency: 1 | Percentage: 7.14%
Character: 'm' | Frequency: 1 | Percentage: 7.14%
Character: 'a' | Frequency: 1 | Percentage: 7.14%
Character: 'l' | Frequency: 2 | Percentage: 14.29%
```

21. Perform modular arithmetic operation

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
def modular_arithmetic(a, b, mod):
    addition = (a + b) % mod
    subtraction = (a - b) % mod
    multiplication = (a * b) % mod
    exponentiation = pow(a, b, mod)
    return addition, subtraction, multiplication, exponentiation
def main():
    a = int(input("Enter the first integer (a): "))
    b = int(input("Enter the second integer (b): "))
    mod = int(input("Enter the modulus (mod): "))
    addition, subtraction, multiplication, exponentiation = modular_arithmetic(a, b, mod)
    print("\nModular Arithmetic Results:")
    print(f"Addition (a + b) % mod = {addition}")
    print(f"Subtraction (a - b) % mod = {subtraction}")
    print(f"Multiplication (a * b) % mod = {multiplication}")
    print(f"Exponentiation (a^b) % mod = {exponentiation}")

if __name__ == "__main__":
    main()
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Enter the first integer (a): 1
Enter the second integer (b): 2
Enter the modulus (mod): 3

Modular Arithmetic Results:
Addition (a + b) % mod = 0
Subtraction (a - b) % mod = 2
Multiplication (a * b) % mod = 2
Exponentiation (a^b) % mod = 1
```

22. Find the prime numbers check the given number is prime or not print the prime numbers with the given range

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True
def find_primes_in_range(start, end):
    """Find and return a list of prime numbers in a given range."""
    primes = []
    for num in range(start, end + 1):
        if is_prime(num):
            primes.append(num)
    return primes
def main():
    start = int(input("Enter the start of the range: "))
    end = int(input("Enter the end of the range: "))
    primes = find_primes_in_range(start, end)
    print(f"Prime numbers between {start} and {end}: {primes}")
    number_to_check = int(input("Enter a number to check if it is prime: "))
    if is_prime(number_to_check):
        print(f"{number_to_check} is a prime number.")
    else:
        print(f"{number_to_check} is not a prime number.")
if __name__ == "__main__":
    main()
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Enter the start of the range: 1
Enter the end of the range: 9
Prime numbers between 1 and 9: [2, 3, 5, 7]
Enter a number to check if it is prime: 4
4 is not a prime number.
```

23. Check the given two numbers are co prime or not

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
import math
def are_coprime(a, b):
    return math.gcd(a, b) == 1
def main():
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    if are_coprime(num1, num2):
        print(f"{num1} and {num2} are co-prime numbers.")
    else:
        print(f"{num1} and {num2} are not co-prime numbers.")
if __name__ == "__main__":
    main()
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Enter the first number: 5
Enter the second number: 9
5 and 9 are co-prime numbers.
```

24. find the factors for the given number (can use python library)

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
import math
def find_factors(num):
    factors = []
    for i in range(1, int(math.sqrt(num)) + 1):
        if num % i == 0:
            factors.append(i)
            if i ≠ num // i:
                factors.append(num // i)
    factors.sort()
    return factors
def main():
    number = int(input("Enter a number to find its factors: "))
    factors = find_factors(number)
    print(f"The factors of {number} are: {factors}")
if __name__ == "__main__":
    main()
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Enter a number to find its factors: 6
The factors of 6 are: [1, 2, 3, 6]
```

25. generate 10 random numbers

Sol:

```
kruthi@kali: ~
File  Actions  Edit  View  Help
import random
def generate_random_numbers(count, lower_bound, upper_bound):
    return [random.randint(lower_bound, upper_bound) for _ in range(count)]
def main():
    count = 10
    lower_bound = int(input("Enter the lower bound: "))
    upper_bound = int(input("Enter the upper bound: "))
    random_numbers = generate_random_numbers(count, lower_bound, upper_bound)
    print(f"Generated {count} random numbers between {lower_bound} and {upper_bound}:")
    print(random_numbers)
if __name__ == "__main__":
    main()
```

Output:

```
┌──(kruthi㉿kali)-[~]
└─$ vi lab2.py

┌──(kruthi㉿kali)-[~]
└─$ python lab2.py
Enter the lower bound: 1
Enter the upper bound: 9
Generated 10 random numbers between 1 and 9:
[3, 1, 1, 3, 9, 8, 9, 7, 1, 8]
```