8/24/2024

Cryptology – b keerthana

Amruthesh

241059041

M.E - Cyber Security

MSIS, MAHE

**String Operations in Python**

1. **Find the length of the string**

* s = "Hello, World!"

length = len(s)

print(length)

1. **Slice the string as per your choice**

* s = "Hello, World!"

slice\_str = s[0:5]

print(slice\_str)

1. **Concatenate two strings**

* s1 = "Hello"

s2 = "World"

concat\_str = s1 + " " + s2

print(concat\_str)

1. **Convert in to lower case in to uppercase character**

* s = "hello"

upper\_s = s.upper()

print(upper\_s)

1. **Convert upper case into lower case characters**

* s = "HELLO"

lower\_s = s.lower()

print(lower\_s)

1. **convert the character into Unicode ( Ascii values)**

* char = 'A'

unicode\_val = ord(char)

print(unicode\_val)

1. **convert Unicode into character**

* unicode\_val = 65

char = chr(unicode\_val)

print(char)

1. **Check whether the given "substring" exists in the string**

* s = "Hello, World!"

substring = "World"

exists = substring in s

print(exists)

1. **Replace the character 'k' with 'h'**

* s = "kitchen"

replace\_s = s.replace('k', 'h')

print(replace\_s)

1. **Pad the string with "x" at the end**

* s = "Hello"

pad\_s = s.ljust(10, 'x')

print(pad\_s)

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

* # To remove whitespace

s = " Hello, World! "

trim\_s = s.strip()

print(trim\_s)

* # To remove specific characters:

s = "xxHello, World!xx"

trim\_s = s.strip('x')

print(trim\_s)

1. **split the given string in to group of five characters**

* s = "abcdefghij"

split\_s = [s[i:i+5] for i in range(0, len(s), 5)]

print(split\_s)

1. **count total number of words**

* s = "Hello World from Amruthesh"

words = s.split()

word\_count = len(words)

print(word\_count)

1. **Find the frequency of each character in the string**

* s = "Hello"

frequency = {}

for char in s:

if char in frequency:

frequency[char] += 1

else:

frequency[char] = 1

print(frequency)

**STDIN and File operators**

1. **get the file name from the user**

* filename = input("Enter the file name: ")

print("File name is:", filename)

1. **check the file exist or not**

* import os

filename = input("Enter the file name: ")

file\_exists = os.path.exists(filename)

print(file\_exists)

**Looping and File handling**

1. **read the contents from the file**

* filename = "example.txt"

with open(filename, 'r') as file:

content = file.read()

print(content)

1. **reverse the contents from the file**

* filename = "example.txt"

with open(filename, 'r') as file:

content = file.read()

reversed\_content = content[::-1]

print(reversed\_content)

1. **Write into the file**

* filename = "example.txt"

with open(filename, 'w') as file:

file.write("This is a new line.")

**Math operations**

1. **Convert Frequency in to percentage (continuation of 12th Question)**

* total\_chars = len(s)

for char, freq in frequency.items():

percentage = (freq / total\_chars) \* 100

print(f"Character: {char}, Frequency: {freq}, Percentage: {percentage:.2f}%")

1. **Perform modular arithmetic operation**

* a = 9

b = 4

mod\_result = a % b

print(mod\_result)

1. **Find the prime numbers**

* **check the given number is prime or not**
* def is\_prime(n):

if n < 2:

return False

for i in range(2, int(n\*\*0.5) + 1):

if n % i == 0:

return False

return True

number = 11

print(is\_prime(number))

* **print the prime numbers with the given range**
* def print\_primes\_in\_range(start, end):

primes = [ ]

for num in range(start, end + 1):

if is\_prime(num):

primes.append(num)

return primes

print(print\_primes\_in\_range(10, 50))

1. **Check the given two numbers are co-prime or not**

* import math

def are\_coprime(a, b):

return math.gcd(a, b) == 1

a, b = 35, 18

print(are\_coprime(a, b))

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

* def factors(n):

return [i for i in range(1, n + 1) if n % i == 0]

number = 28

print(factors(number))

1. **generate 10 random numbers**

* import random

random\_numbers = [random.randint(1, 100) for \_ in range(10)]

print(random\_numbers)

1. **Explore: Miller-Rabin Test (pen paper method)**

* The **Miller-Rabin primality test** is a probabilistic algorithm used to determine whether a given number is a prime. It is an extension of the **Fermat primality test** and is more effective at identifying composite numbers. Although it is probabilistic, it can be made deterministic for numbers up to a certain size.
* To ascertain whether a given integer is a prime, one can apply the probabilistic Miller-Rabin test algorithm. It is based on prime number features and modular exponentiation.

#### ****Mathematical Background****

Given an odd integer n, the Miller-Rabin test is based on the following concepts:

1. **Fermat's Little Theorem**:

If **n** is a prime number and **a** is any integer such that **1 < a < n-1**, then:

**an−1 ≡ 1 (mod n)**

If this condition does not hold, then **n** is composite.

1. **Witnesses and Strong Witnesses**:

A witness for the compositeness of **n** is an integer **a** such that:

**an−1 ≢ 1 (mod n)**

A strong witness (also called a Miller-Rabin witness) goes further and checks additional conditions derived from writing **n-1** in the form:

**n−1 = 2r⋅d (where d is odd)**

* **Algorithm Steps**

1. Write **n - 1** as **2^r \* d** where **d** is odd.
2. Choose a random integer **a** such that **1 < a < n-1**.
3. Compute:

**x = ad (mod n)**

* + If **x == 1** or **x == n-1**, the test passes for this base.

1. Repeat the squaring:

**x = x2 (mod n)**

* + If **x == n-1**, the test passes for this base.
  + If none of these conditions are met, **n** is composite.

If a base **a** pass the Miller-Rabin test, then n is **probably prime**; otherwise, **n** is **definitely composite**.