Cryptography Assignment

**String Operations in Python**

1. Find the length of the string

s = "Hi good morning!"

print("Length of the string:", len(s))

2. Slice the string as per your choice

s = "Hello goodmorning!"

start, end = 7, 12

print("Sliced string:", s[start:end])

3. Concatenate two strings

s1 = "Hello, "

s2 = "good evening"`

print("Concatenated string:", s1 + s2)

4. Convert in to lower case in to uppercase character

s = "Hi Everyone"

print("Lower case string:", s.lower())

5. Convert upper case into lower case characters

s = " Hi Everyone "

print("Upper case string:", s.upper())

6. convert the character into Unicode ( Ascii values)

s = "Hii"

print("Unicode (ASCII) values:", [ord(c) for c in s])

7. convert Unicode into character

unicode\_list = [70, 99, 108, 108, 111]

print("String from Unicode values:", ''.join(chr(u) for u in unicode\_list))

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

s = "Hello dear"

substring = "dear"

print("Substring exists:", substring in s)

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

s = "chicken"

print("Replaced string:", s.replace('k', 'h'))

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

s = "Hurry"

length = 10

print("Padded string:", s.ljust(length, 'x'))

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

s = " Hutt, house! "

print("Stripped string:", s.strip())

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

s = "Higoodmorning"

group\_size = 5

print("Grouped string:", [s[i:i+group\_size] for i in range(0, len(s), group\_size)])

13. count total number of words

s = "good night! Bye sweet dreams?"

print("Number of words:", len(s.split()))

14. Find the frequency of each characters in the string

s = "Hello"

print("Character frequency:", dict(Counter(s)))

**STDIN and File operators**

15. get the file name from the user

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

16. check the file exist or not

filename = "a1.txt"

print("File exists:", os.path.isfile(filename))

**Looping and File handling**

17. read the contents from the file

filename = "a1.txt"

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

print("File contents:\n", file.read())

18. reverse the contents from the file

filename = "a1.txt"

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

contents = file.read()

print("Reversed contents:\n", contents[::-1])

19. Write into the file

filename = "a1.txt"

content = "Hii, Hello"

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

file.write(content)

**Math operations**

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

frequencies = {'a': 6, 'b': 2, 'c': 5}

total = sum(frequencies.values())

percentages = {k: (v / total) \* 100 for k, v in frequencies.items()}

print("Frequency percentages:", percentages)

21. Perform modular arithmetic operation

a, b = 15, 2

operation = 'add'

if operation == 'add':

result = (a + b) % b

elif operation == 'subtract':

result = (a - b) % b

elif operation == 'multiply':

result = (a \* b) % b

elif operation == 'divide':

result = (a // b) % b

else:

raise ValueError("Invalid operation")

print("Modular result:", result)

22. Find the prime numbers

check the given number is prime or not

print the prime numbers with the given range

def is\_prime(n):

if n <= 1:

return False

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

if n % i == 0:

return False

return True

start, end = 10, 20

primes = [i for i in range(start, end + 1) if is\_prime(i)]

print("Prime numbers in range:", primes)

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

a, b = 15, 30

print("Are co-prime:", gcd(a, b) == 1) # Output: Are co-prime: True

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

n = 32

factors = []

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

if n % i == 0:

factors.append(i)

if i != n // i:

factors.append(n // i)

print("Factors:", sorted(factors))

25. generate 10 random numbers

count = 10

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

print("Random numbers:", random\_numbers)

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

The Miller–Rabin primality test or Rabin–Miller primality test is a probabilistic [primality test](https://en.wikipedia.org/wiki/Primality_test): an [algorithm](https://en.wikipedia.org/wiki/Algorithm) which determines whether a given number is [likely to be prime](https://en.wikipedia.org/wiki/Probable_prime).