```
def merge_sort(a):
    if len(a) <= 1:
        return a
    mid = len(a) // 2
    left = merge_sort(a[:mid])
    right = merge_sort(a[mid:])
    return merge(left, right)
def merge(l, r):
    res = []
    while l and r:
        res.append((l if l[0] < r[0] else
r).pop(0))
    res += l + r
    return res
arr = [38, 27, 43, 3, 9, 82, 10]
print("Original:", arr)
arr = merge_sort(arr)
print("Sorted:", arr)
output
Original: [38, 27, 43, 3, 9, 82, 10]
Sorted: [3, 9, 10, 27, 38, 43, 82]
```

```
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] =
arr[min_idx], arr[i]

arr = [64, 25, 12, 22, 11]
print("Original:", arr)
selection_sort(arr)
print("Sorted:", arr)
OUTPUT
Original: [64, 25, 12, 22, 11]
Sorted: [11, 12, 22, 25, 64]
```

```
def insertion_sort(arr):
    for i in range(1,
len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and
arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

arr = [12, 11, 13, 5, 6]
print("Original:", arr)
insertion_sort(arr)
print("Sorted:", arr)
OUTPUT------
Original: [12, 11, 13, 5,
6]
Sorted: [5, 6, 11, 12, 13]
```

```
def nearly_equal(a, b):
    if a == b or abs(len(a) - len(b)) > 1:
return False
    if len(a) > len(b): a, b = b, a
    i = j = diff = 0
    while i < len(a) and j < len(b):
        if a[i] != b[j]:
            if diff: return False
            diff = 1
            if len(a) == len(b): i += 1
        else: i += 1
        j += 1
    return True
print(nearly_equal("cat", "cut"))   #
True
print(nearly_equal("cat", "cats"))  #
True
print(nearly_equal("cat", "at"))    #
True
print(nearly_equal("cat", "dog"))   #
False
print(nearly_equal("cat", "cat"))   #
False
OUTPUT-------
True
True
True
False
False
```

```
from math import gcd

lcm = lambda a, b: abs(a * b) //
gcd(a, b)

# Example usage
a, b = 12, 18
print("GCD:", gcd(a, b))
print("LCM:", lcm(a, b))
OUTPUT----
GCD: 6
LCM: 36
```

program to print each line of a file in reverse order.

```
Def
print_reverse_lines(filename):
    with open(filename, 'r') as
file:
        for line in file:
            print(line.strip()[::-1])

# Example usage:
filename = 'your_file.txt'  #
Replace with your file path
print_reverse_lines(filename)
OUTPUT----
Hello world
Python is great
Reverse this------ABC.TXT
dlrow olleH
taerg si nohtyP
siht esreveR
```

count the frequency of characters

```
from collections import Counter
def count_char_frequency(filename):
    with open(filename, 'r') as file:
        content = file.read()

    # Count frequency of each
character
    char_frequency = Counter(content)

    return char_frequency

# Example usage:
filename = 'your_file.txt'  # Replace
with your file path
char_frequency =
count_char_frequency(filename)
print(char_frequency)
OUTPUT------
def hello():
    print("Hello, World!")------AB.TXT
Counter({' ': 5, 'e': 3, 'o': 3, 'l': 3, 'H': 2,
'd': 1, 'f': 1, 'h': 1, '(': 1, ')': 1, '"': 2, ',':
1, '!': 1})
```

Write a program to count the numbers of characters in the string and store them in a dictionary data structure

```
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

# Example usage:
input_string = "hello world"
char_count =
count_characters(input_string)
print(char_count)
OUTPUT------
{'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'w': 1,
'r': 1, 'd': 1}
```

sum of all the primes below two million

```
def sum_of_primes(limit):
    sieve = [True] * limit
    sieve[0] = sieve[1] = False  #
0 and 1 are not prime
    for start in range(2, int(limit
** 0.5) + 1):
        if sieve[start]:
            for i in range(start *
start, limit, start):
                sieve[i] = False
    return sum(i for i in
range(limit) if sieve[i])

# Sum of all primes below
two million
limit = 2000000
result = sum_of_primes(limit)
print(result)
OTUPUT---
142913828922
```

program to use split and join methods in the string and trace a birthday of a person with a dictionary data structure

```
def trace_birthday(birthday_str):
    day, month, year =
birthday_str.split('-')
    return {'day': day, 'month': month,
'year': year}

def format_birthday(birthday_dict):
    return '-'.join([birthday_dict['day'],
birthday_dict['month'],
birthday_dict['year']])

# Example usage:
birthday_str = "12-04-1990"
birthday_dict =
trace_birthday(birthday_str)
print("Birthday as dictionary:",
birthday_dict)
print("Formatted Birthday:",
format_birthday(birthday_dict))
OUTPUT-----
Birthday as dictionary: {'day': '12',
'month': '04', 'year': '1990'}
Formatted Birthday: 12-04-1990
```

Fibonacci sequence whose values do not exceed four million, WAP to find the sum of the even-valued terms.

```
def sum_even_fibonacci(limit):
    a, b = 0, 1
    total = 0
    while b <= limit:
        if b % 2 == 0:
            total += b
        a, b = b, a + b
    return total

# Find the sum of even Fibonacci
numbers below four million
print(sum_even_fibonacci(4000000))
OUTPUT------
4613732
```

```
def countdown():
    number = int(input("Enter a
number: "))
    while number >= 0:
        print(number)
        number -= 1

countdown()
OUTPUT------
Enter a number: 5
5
4
3
2
1
0
```

loops over a sequence

```
def
loop_over_sequence():
    sequence = [1, 2, 3, 4,
5]
    for number in
sequence:
        print(number)

loop_over_sequence()
OUTPUT----
1
2
3
4
5
```

Write a program to compute the number of characters, words and lines in a file.

```
def file_stats(filename):
    with open(filename, 'r') as file:
        lines = file.readlines()

    num_lines = len(lines)
    num_words = sum(len(line.split())
for line in lines)
    num_chars = sum(len(line) for line
in lines)

    return num_lines, num_words,
num_chars

# Example usage:
filename = 'your_file.txt'  # Replace
with your file path
lines, words, chars =
file_stats(filename)
print(f"Lines: {lines}, Words: {words},
Characters: {chars}")
OUTPUT---
Hello world
This is Python
Reverse this-----
Lines: 3, Words: 5, Characters: 26
```

Using a for loop, write a program that prints out the decimal equivalents of 1/2, 1/3, 1/4, . . . , 1/10 def **print_decimal_equivalents():** **for i in range(2, 11):** **print(f"1/{i} = {1/i}")** **print_decimal_equivalents()** **OUTPUT---** 1/2 = 0.5 1/3 = 0.3333333333333333 1/4 = 0.25 1/5 = 0.2 1/6 = 0.16666666666666666 1/7 = 0.14285714285714285 1/8 = 0.125 1/9 = 0.1111111111111111 1/10 = 0.1			