Saumya Deep

In [1]:
```
#1.Program to append and insert elements into an array

import array as arr

# Create an array
a = arr.array('i', [1, 2, 3])

# Append element
a.append(4)
print("After append:", a)

# Insert element at position 1
a.insert(1, 5)
print("After insert:", a)
```

```
After append: array('i', [1, 2, 3, 4])
After insert: array('i', [1, 5, 2, 3, 4])
```

In [2]:
```
#1.Program to append and insert elements into an array through user input

import array as arr

# Create an empty integer array
a = arr.array('i', [])

# Take number of elements from the user
n = int(input("How many elements do you want to add to the array? "))
for i in range(n):
    num = int(input(f"Enter element #{i + 1}: "))
    a.append(num)

print("Initial array:", a)

# Append a new element from user
append_element = int(input("Enter an element to append at the end: "))
a.append(append_element)
print("After append:", a)

# Insert an element at a specific position
insert_element = int(input("Enter an element to insert: "))
position = int(input("Enter the position to insert at (starting from 0): "))
```

```
# Check for valid position
if 0 <= position <= len(a):
    a.insert(position, insert_element)
    print("After insert:", a)
else:
    print("Invalid position. No element inserted.")
```

```
How many elements do you want to add to the array? 5
Enter element #1: 1
Enter element #2: 2
Enter element #3: 3
Enter element #4: 4
Enter element #5: 5
Initial array: array('i', [1, 2, 3, 4, 5])
Enter an element to append at the end: 7
After append: array('i', [1, 2, 3, 4, 5, 7])
Enter an element to insert: 5
Enter the position to insert at (starting from 0): 1
After insert: array('i', [1, 5, 2, 3, 4, 5, 7])
```

In [3]:

```
#2.Program to reverse the elements of an array

import array as arr

a = arr.array('i', [1, 2, 3, 4, 5])
print("Original array:", a)

# Reverse the array
a.reverse()
print("Reversed array:", a)
```

```
Original array: array('i', [1, 2, 3, 4, 5])
Reversed array: array('i', [5, 4, 3, 2, 1])
```

In [4]:

```
#3.Program to find the average of array elements

import array as arr

a = arr.array('i', [10, 20, 30, 40, 50])
total = sum(a)
average = total / len(a)
print("Average:", average)
```

```
Average: 30.0
```

Saumya Deep

In [5]:
```
#4.Program to find the index of an element in an array

import array as arr

a = arr.array('i', [10, 20, 30, 40, 50])
element = 30
index = a.index(element)
print(f"Index of {element}:", index)
```

```
Index of 30: 2
```

In [1]:
```
#4.Program to find the index of an element in an array through user input

import array as arr

# Take array elements from the user
n = int(input("How many elements do you want to add to the array? "))
a = arr.array('i', [])

for i in range(n):
    num = int(input(f"Enter element #{i + 1}: "))
    a.append(num)

# Take the element to search
element = int(input("Enter the element to find the index of: "))

# Check if element exists and print index
if element in a:
    index = a.index(element)
    print(f"Index of {element}:", index)
else:
    print(f"{element} is not in the array.")
```

```
How many elements do you want to add to the array? 5
Enter element #1: 10
Enter element #2: 20
Enter element #3: 30
Enter element #4: 40
Enter element #5: 50
Enter the element to find the index of: 30
Index of 30: 2
```

In [6]:
```
#5.Program to find the maximum and minimum values in an array
```

```
import array as arr

a = arr.array('i', [10, 20, 30, 40, 50])
print("Maximum:", max(a))
print("Minimum:", min(a))
```

```
Maximum: 50
Minimum: 10
```

In [4]:

```
#5.Program to find the maximum and minimum values in an array using through user input

import array as arr

# Create an empty array of integers
a = arr.array('i', [])

# Get number of elements from user
n = int(input("How many elements do you want to add to the array? "))
for i in range(n):
    num = int(input(f"Enter element #{i + 1}: "))
    a.append(num)

# Display the array
print("Your array:", a)

# Find and display the maximum and minimum values
print("Maximum:", max(a))
print("Minimum:", min(a))
```

```
How many elements do you want to add to the array? 6
Enter element #1: 10
Enter element #2: 20
Enter element #3: 5
Enter element #4: 30
Enter element #5: 45
Enter element #6: 25
Your array: array('i', [10, 20, 5, 30, 45, 25])
Maximum: 45
Minimum: 5
```

In [7]:

```
#6.Program to reverse a string

string = input("Enter a string:")
```

```
reversed_string = string[::-1]
print("Reversed string:", reversed_string)
```

```
Enter a string:Hello World
Reversed string: dlroW olleH
```

In [10]:
```
#7.Program to checkif a string is an anagram of another

def is_anagram(str1, str2):
    return sorted(str1.lower()) == sorted(str2.lower())

str1 = input("Enter the first string: ")
str2 = input("Enter the second string: ")

if is_anagram(str1, str2):
    print("The strings are anagrams.")
else:
    print("The strings are not anagrams.")
```

```
Enter the first string: listen
Enter the second string: silent
The strings are anagrams.
```

In [11]:
```
#8.Program to remove all whitespaces from a string

string = "  Hello   World  "
no_spaces = string.replace(" ", "")
print("Without spaces:", no_spaces)
```

```
Without spaces: HelloWorld
```

In [12]:
```
#9.Program to capitalize the first letter of each word in a string

string = "hello world"
capitalized = string.title()
print("Capitalized:", capitalized)
```

```
Capitalized: Hello World
```

In [13]:
```
#10.Program to check if all characters in a string are digits

# Take input from the user
string = input("Enter a string: ")

# Check if the string contains only digits
```

```
if string.isdigit():
    print("The string contains only digits.")
else:
    print("The string does not contain only digits.")
```

```
Enter a string: 21376
The string contains only digits.
```

In [14]:
```
#11.Program to count the number of words in a sentence

sentence = input("Enter a sentence:")
words = sentence.split()
print("Number of words:", len(words))
```

```
Enter a sentence:This is a sample sentence.
Number of words: 5
```

In [15]:
```
#12.Program to find all substrings of a given string

# Take input from the user
string = input("Enter a string: ")

# Generate all possible substrings
substrings = [string[i:j] for i in range(len(string))
              for j in range(i + 1, len(string) + 1)]

# Print the substrings
print("Substrings:", substrings)
```

```
Enter a string: abc
Substrings: ['a', 'ab', 'abc', 'b', 'bc', 'c']
```

In [16]:
```
#13.Program to find the most frequent character in a string

# Take input from the user
string = input("Enter a string: ")

# Create a dictionary to store character counts
char_count = {}

# Count character frequencies, ignoring spaces
for char in string:
    if char != ' ':
        if char in char_count:
            char_count[char] += 1
        else:
            char_count[char] = 1

# Find the most frequent character
most_common_char = max(char_count, key=char_count.get)
most_common_count = char_count[most_common_char]

# Print the result
print(f"Most frequent character: '{most_common_char}' with {most_common_count} occurrences")
```

```
Enter a string: Hello World
Most frequent character: 'l' with 3 occurrences
```

In [18]:

```
#13.Program to find the most frequent character in a string using counter

from collections import Counter

string = input("Enter a string: ")
counter = Counter(string)
# Remove space from consideration
del counter[' ']
most_common = counter.most_common(1)[0]
print("Most frequent character:", most_common[0], "with", most_common[1], "occurrences")
```

```
Enter a string: hello world
Most frequent character: l with 3 occurrences
```

In [22]:

```
#14.Program to check if a string contains only unique characters

def unique_check(string):
    return len(set(string)) == len(string)

# Take user input
user_input = input("Enter a string: ")

# Check and print the result
if unique_check(user_input):
    print("The string has all unique characters.")
else:
    print("The string does NOT have all unique characters.")
```

```
Enter a string: abcde
The string has all unique characters.
```

Saumya Deep

In [7]:
```python
# Create a dictionary from two lists

# Take input from the user for keys and values
keys = input("Enter keys (comma-separated): ").split(",")
values = input("Enter values (comma-separated): ").split(",")

# Check if both lists have the same length
if len(keys) != len(values):
    print("Error: Number of keys and values must be the same.")
else:
    # Create the dictionary
    result_dict = {}
    for i in range(len(keys)):
        result_dict[keys[i].strip()] = values[i].strip()

    # Display the result
    print("Created dictionary:", result_dict)
```

```
Enter keys (comma-separated): a,b,c,d
Enter values (comma-separated): 1,2,3,4
Created dictionary: {'a': '1', 'b': '2', 'c': '3', 'd': '4'}
```

In [5]:
```python
# Create a dictionary from two lists through different methods

keys = ['a', 'b', 'c', 'd']
values = [1, 2, 3, 4]

# Method 1: Using zip() with dict()
result_dict = dict(zip(keys, values))
print("Method 1:", result_dict)

# Method 2: Using dictionary comprehension
result_dict = {k: v for k, v in zip(keys, values)}
print("Method 2:", result_dict)

# Method 3: Using a loop (more explicit)
result_dict = {}
for i in range(len(keys)):
    result_dict[keys[i]] = values[i]
print("Method 3:", result_dict)

# Handling unequal length lists
keys = ['a', 'b', 'c', 'd', 'e']  # Extra key
```

```
values = [1, 2, 3]                # Fewer values

# Using zip() which stops at shortest list
result_dict = dict(zip(keys, values))
print("\nWith unequal lists (zip stops at shortest):", result_dict)

# Using itertools.zip_longest to fill missing values
from itertools import zip_longest
result_dict = dict(zip_longest(keys, values, fillvalue=None))
print("With zip_longest (fills missing values):", result_dict)
```

```
Method 1: {'a': 1, 'b': 2, 'c': 3, 'd': 4}
Method 2: {'a': 1, 'b': 2, 'c': 3, 'd': 4}
Method 3: {'a': 1, 'b': 2, 'c': 3, 'd': 4}

With unequal lists (zip stops at shortest): {'a': 1, 'b': 2, 'c': 3}
With zip_longest (fills missing values): {'a': 1, 'b': 2, 'c': 3, 'd': None, 'e': None}
```

In [6]:
```
# Create a dictionary from two lists through user input

keys = input("Enter keys separated by commas: ").split(',')
values = input("Enter values separated by commas: ").split(',')

# Convert values to integers if possible
try:
    values = [int(v.strip()) if v.strip().isdigit() else v.strip() for v in values]
except ValueError:
    values = [v.strip() for v in values]

# Create dictionary
keys = [k.strip() for k in keys]
result_dict = dict(zip(keys, values))

print("\nCreated dictionary:")
for key, value in result_dict.items():
    print(f"{key}: {value}")

# Handle unequal lengths
if len(keys) != len(values):
    print("\nWarning: Number of keys and values don't match!")
    print(f"Keys: {len(keys)}, Values: {len(values)}")
    print("Dictionary created only for matching pairs")
```

```
Enter keys separated by commas: a,b,c,d
Enter values separated by commas: 1,2,3,4

Created dictionary:
a: 1
b: 2
c: 3
d: 4
```

In [23]:

```
#15.Program to create a list of the first 10 even numbers

even_numbers = [2*i for i in range(1, 11)]
print("First 10 even numbers:", even_numbers)
```

```
First 10 even numbers: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

In [25]:

```
#16.Program to find the largest and smallest elements in a list using map

# Take user input
user_input = input("Enter numbers separated by spaces: ")

# Convert input string into a list of integers
numbers = list(map(int, user_input.split()))

# Find and print the largest and smallest number
print("Largest:", max(numbers))
print("Smallest:", min(numbers))
```

```
Enter numbers separated by spaces: 10 20 5 40 30
Largest: 40
Smallest: 5
```

In [26]:

```
#16.Program to find the largest and smallest elements in a list using loop

# Take user input
user_input = input("Enter numbers separated by spaces: ")

# Initialize an empty list
numbers = []

# Split the input and convert each item using a loop
for i in user_input.split():
    numbers.append(int(i))
```

```
# Find and print the largest and smallest number
print("Largest:", max(numbers))
print("Smallest:", min(numbers))
```

```
Enter numbers separated by spaces: 10 20 5 40 30
Largest: 40
Smallest: 5
```

In [27]:
```
#17.Program to reverse a list without using built-in methods

def reverse_list(lst):
    return lst[::-1]

numbers = [1, 2, 3, 4, 5]
print("Reversed list:", reverse_list(numbers))
```

```
Reversed list: [5, 4, 3, 2, 1]
```

In [28]:
```
#18.Program to count the number of occurrences of an element in a list using count

# Take list input from the user
user_input = input("Enter numbers separated by spaces: ")

# Convert the input into a list of integers
numbers = []
for i in user_input.split():
    numbers.append(int(i))

# Take the element to count
element = int(input("Enter the number you want to count: "))

# Count the occurrences
count = numbers.count(element)

# Display the result
print(f"Number of {element}'s:", count)
```

```
Enter numbers separated by spaces: 1 2 3 2 4 2 5
Enter the number you want to count: 2
Number of 2's: 3
```

In [29]:
```
#18.Program to count the number of occurrences of an element in a list

# Take list input from the user
user_input = input("Enter numbers separated by spaces: ")
```

```
# Convert the input into a list of integers
numbers = []
for i in user_input.split():
    numbers.append(int(i))

# Take the element to count
element = int(input("Enter the number you want to count: "))

# Manual count using a loop
count = 0
for num in numbers:
    if num == element:
        count += 1

# Display the result
print(f"Number of {element}'s:", count)
```

```
Enter numbers separated by spaces: 1 2 3 2 4 2 5
Enter the number you want to count: 2
Number of 2's: 3
```

In [30]:

```
#19.Program to remove all negative numbers from a list

numbers = [1, -2, 3, -4, 5, -6]
positive_numbers = [x for x in numbers if x >= 0]
print("Positive numbers:", positive_numbers)
```

```
Positive numbers: [1, 3, 5]
```

In [31]:

```
#19.Program to remove all negative numbers from a list using append

# Original list
numbers = [1, -2, 3, -4, 5, -6]

# Initialize an empty list to store positive numbers
positive_numbers = []

# Use a loop to filter positive numbers
for x in numbers:
    if x >= 0:
        positive_numbers.append(x)
```

```
# Print the result
print("Positive numbers:", positive_numbers)
```

```
Positive numbers: [1, 3, 5]
```

In [32]:

```
#20.Program to insert an element at the beginning, middle, and end of a list

lst = [2, 3, 4]
element = 1

# Insert at beginning
lst.insert(0, element)

# Insert at middle
middle = len(lst) // 2
lst.insert(middle, element)

# Insert at end
lst.append(element)

print("Modified list:", lst)
```

```
Modified list: [1, 2, 1, 3, 4, 1]
```

In [33]:

```
#21.Program to find the common elements between two lists

list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]
common = list(set(list1) & set(list2))
print("Common elements:", common)
```

```
Common elements: [4, 5]
```

In [34]:

```
#22.Program to split a list into two halves

def split_list(lst):
    mid = len(lst) // 2
    return lst[:mid], lst[mid:]

numbers = [1, 2, 3, 4, 5, 6]
first_half, second_half = split_list(numbers)
print("First half:", first_half)
print("Second half:", second_half)
```

```
First half: [1, 2, 3]
Second half: [4, 5, 6]
```

In [35]:
```
#23.Program to create a tuple with different data types

mixed_tuple = (1, "hello", 3.14, True, [1, 2, 3])
print("Mixed tuple:", mixed_tuple)
```

```
Mixed tuple: (1, 'hello', 3.14, True, [1, 2, 3])
```

In [36]:
```
#24.Program to slice a tuple and access specific elements

t = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
print("Slice from index 2 to 5:", t[2:6])
print("Every second element:", t[::2])
print("Last 3 elements:", t[-3:])
```

```
Slice from index 2 to 5: (2, 3, 4, 5)
Every second element: (0, 2, 4, 6, 8)
Last 3 elements: (7, 8, 9)
```

In [37]:
```
#25.Program to convert a list of tuples into a dictionary

list_of_tuples = [("a", 1), ("b", 2), ("c", 3)]
dictionary = dict(list_of_tuples)
print("Dictionary:", dictionary)
```

```
Dictionary: {'a': 1, 'b': 2, 'c': 3}
```

In [38]:
```
#26.Program to check if a tuple is empty

empty_tuple = ()
non_empty_tuple = (1,)

print("Is empty_tuple empty?", len(empty_tuple) == 0)
print("Is non_empty_tuple empty?", len(non_empty_tuple) == 0)
```

```
Is empty_tuple empty? True
Is non_empty_tuple empty? False
```

In [39]:
```
#27.Program to add an element to a tuple (workaround since tuples are immutable)

original_tuple = (1, 2, 3)
new_element = 4
```

```
new_tuple = original_tuple + (new_element,)
print("New tuple:", new_tuple)
```

```
New tuple: (1, 2, 3, 4)
```

In [40]:

```
#28.Program to remove an item from a tuple

def remove_item(t, item):
    return tuple(i for i in t if i != item)

t = (1, 2, 3, 4, 5)
new_t = remove_item(t, 3)
print("After removal:", new_t)
```

```
After removal: (1, 2, 4, 5)
```

In [41]:

```
#28.Program to remove an item from a tuple through user input.

def remove_item(t, item):
    return tuple(x for x in t if x != item)

# Take tuple input from user
user_input = input("Enter numbers for the tuple, separated by spaces: ")
t = tuple(int(x) for x in user_input.split())

# Take the element to remove
item = int(input("Enter the element you want to remove: "))

# Call the function
new_t = remove_item(t, item)

# Display results
print(f"Element to remove: {item}")
print("Original tuple:", t)
print("Modified tuple:", new_t)
```

```
Enter numbers for the tuple, separated by spaces: 1 2 3 4 5
Enter the element you want to remove: 3
Element to remove: 3
Original tuple: (1, 2, 3, 4, 5)
Modified tuple: (1, 2, 4, 5)
```

In [42]:

```
#29.Program to find repeated elements in a tuple using counter

from collections import Counter
```

```
t = (1, 2, 3, 2, 4, 5, 4)
counter = Counter(t)
repeats = [item for item, count in counter.items() if count > 1]
print("Repeated elements:", repeats)
```

```
Repeated elements: [2, 4]
```

In [43]:
```
#29.Program to find repeated elements in a tuple using loop

# Input from user
user_input = input("Enter numbers for the tuple, separated by spaces: ")
t = tuple(int(i) for i in user_input.split())

# Dictionary to store frequencies
frequency = {}

# Count occurrences using a loop
for item in t:
    if item in frequency:
        frequency[item] += 1
    else:
        frequency[item] = 1

# Find repeated elements
repeats = []
for item in frequency:
    if frequency[item] > 1:
        repeats.append(item)

# Display result
print("Repeated elements:", repeats)
```

```
Enter numbers for the tuple, separated by spaces: 1 2 3 2 4 5 4
Repeated elements: [2, 4]
```

In [44]:
```
#30.Program to add a key-value pair to a dictionary

d = {"a": 1, "b": 2}
d["c"] = 3
print("Updated dictionary:", d)
```

```
Updated dictionary: {'a': 1, 'b': 2, 'c': 3}
```

Saumya Deep

In [45]:
```
#31.Program to check if a key exists in a dictionary

d = {"a": 1, "b": 2, "c": 3}
key = "b"
print(f"Does '{key}' exist?", key in d)
```

```
Does 'b' exist? True
```

In [46]:
```
#31.Program to check if a key exists in a dictionary through user input

# Create dictionary from user input
d = {}
n = int(input("How many key-value pairs do you want to enter? "))

for i in range(n):
    key = input(f"Enter key #{i+1}: ")
    value = int(input(f"Enter value for key '{key}': "))
    d[key] = value

# Take the key to check
key_to_check = input("Enter the key you want to check: ")

# Check if the key exists
print(f"Does '{key_to_check}' exist?", key_to_check in d)
```

```
How many key-value pairs do you want to enter? 3
Enter key #1: a
Enter value for key 'a': 1
Enter key #2: b
Enter value for key 'b': 2
Enter key #3: c
Enter value for key 'c': 3
Enter the key you want to check: b
Does 'b' exist? True
```

In [47]:
```
#32.Program to sum all values in a dictionary

d = {"a": 10, "b": 20, "c": 30}
total = sum(d.values())
print("Sum of values:", total)
```

```
Sum of values: 60
```

Saumya Deep

In [48]:
```
#33.Program to remove a key from a dictionary

d = {"a": 1, "b": 2, "c": 3}
del d["b"]
print("After removal:", d)
```

```
After removal: {'a': 1, 'c': 3}
```

In [49]:
```
#33.Program to remove a key from a dictionary through user input

# Create dictionary from user input
d = {}
n = int(input("How many key-value pairs do you want to enter? "))

for i in range(n):
    key = input(f"Enter key #{i+1}: ")
    value = int(input(f"Enter value for key '{key}': "))
    d[key] = value

# Take the key to delete
key_to_delete = input("Enter the key you want to delete: ")

# Delete the key if it exists
if key_to_delete in d:
    del d[key_to_delete]
    print(f"Key '{key_to_delete}' has been removed.")
else:
    print(f"Key '{key_to_delete}' does not exist.")

# Display updated dictionary
print("After removal:", d)
```

```
How many key-value pairs do you want to enter? 3
Enter key #1: a
Enter value for key 'a': 1
Enter key #2: b
Enter value for key 'b': 2
Enter key #3: c
Enter value for key 'c': 3
Enter the key you want to delete: b
Key 'b' has been removed.
After removal: {'a': 1, 'c': 3}
```

Saumya Deep

In [50]:
```
#34.Program to iterate over a dictionary and print its keys and values

d = {"a": 1, "b": 2, "c": 3}
for key, value in d.items():
    print(f"Key: {key}, Value: {value}")
```

```
Key: a, Value: 1
Key: b, Value: 2
Key: c, Value: 3
```

In [51]:
```
#35.Program to create a dictionary with numbers from 1 to n as keys and their squares as values

n = int(input("Enter the limit: "))
squares = {i: i**2 for i in range(1, n+1)}
print("Number squares:", squares)
```

```
Enter the limit: 5
Number squares: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

In [53]:
```
#36.Program to open a text file and read its contents line by line, printing each line

with open('sample.txt', 'r') as file:
    for line in file:
        print(line.strip())
```

```
Hello, this is a sample text file.
It contains multiple lines.
Each line will be printed without extra spaces.
This is useful for reading file content line by line.
Have a great day!
```

In [54]:
```
#37.Program to write a list of strings to a file, each string on a new line

lines = ["First line", "Second line", "Third line"]
with open('output1.txt', 'w') as file:
    for line in lines:
        file.write(line + '\n')
print("File written successfully.")
```

```
File written successfully.
```

In [55]:
```
#37.Program to write a list of strings to a file, each string on a new line through user input

# Ask the user how many lines they want to enter
```

Saumya Deep

```
n = int(input("How many lines do you want to write to the file? "))

# Collect lines from the user
lines = []
for i in range(n):
    line = input(f"Enter line #{i + 1}: ")
    lines.append(line)

# Write lines to the file
with open('output2.txt', 'w') as file:
    for line in lines:
        file.write(line + '\n')

print("File written successfully.")
```

```
How many lines do you want to write to the file? 3
Enter line #1: First line
Enter line #2: Second line
Enter line #3: Third line
File written successfully.
```

In [56]:

```
#38.Program to read a file and count the number of words and lines in it

with open('sample.txt', 'r') as file:
    lines = file.readlines()
    line_count = len(lines)
    word_count = sum(len(line.split()) for line in lines)

print(f"Lines: {line_count}, Words: {word_count}")
```

```
Lines: 5, Words: 33
```

In [58]:

```
#38.Program to read a file and count the number of words and lines in it. Path entered by user.

# Ask the user for the file path or name
file_path = input("Enter the file path or name (e.g., sample.txt): ")

try:
    with open(file_path, 'r') as file:
        lines = file.readlines()
        line_count = len(lines)
        word_count = sum(len(line.split()) for line in lines)

    print(f"Lines: {line_count}, Words: {word_count}")
```

```
except FileNotFoundError:
    print("Error: The file was not found. Please check the path and try again.")
except Exception as e:
    print(f"An error occurred: {e}")
```

```
Enter the file path or name (e.g., sample.txt): sample.txt
Lines: 5, Words: 33
```

In [60]:
```
#39.Program to copy the contents of one file to another without using built-in shutil module

with open('sample.txt', 'r') as source, open('destination.txt', 'w') as dest:
    for line in source:
        dest.write(line)
print("File copied successfully.")
```

```
File copied successfully.
```

In [67]:
```
#40.Program to open a file, search for a specific word, and print the lines where it appears

search_word = "file"
found = False  # Flag to track if any match is found

with open('sample.txt', 'r') as file:
    for line_num, line in enumerate(file, 1):
        if search_word in line:
            print(f"Line {line_num}: {line.strip()}")
            found = True

if not found:
    print(f"No lines found containing the word '{search_word}'.")
```

```
Line 1: Hello, this is a sample text file.
Line 4: This is useful for reading file content line by line.
```

In [64]:
```
#40.Program to open a file, search for a specific word, and print the lines where it appears
#Enhanced Code.

# Take inputs from user
file_path = input("Enter the file name or path (e.g., sample.txt): ")
search_word = input("Enter the word you want to search: ")
case_sensitive = input("Should the search be case-sensitive? (yes/no): ").strip().lower() == "yes"

match_count = 0  # Counter for matched lines
```

```python
try:
    with open(file_path, 'r') as file:
        for line_num, line in enumerate(file, 1):
            content = line if case_sensitive else line.lower()
            word = search_word if case_sensitive else search_word.lower()

            if word in content:
                print(f"Line {line_num}: {line.strip()}")
                match_count += 1

    if match_count == 0:
        print("No matches found.")
    else:
        print(f"\nTotal matches found: {match_count}")

except FileNotFoundError:
    print("Error: The specified file does not exist.")
except Exception as e:
    print(f"An error occurred: {e}")
```

```
Enter the file name or path (e.g., sample.txt): sample.txt
Enter the word you want to search: file
Should the search be case-sensitive? (yes/no): no
Line 1: Hello, this is a sample text file.
Line 4: This is useful for reading file content line by line.

Total matches found: 2
```