20th_ August_ Python_Data_Types

September 14, 2023

[1]:
```
#TOPIC: String Based Assignment Problem

#Question 1: Write a program to reverse a string.
def reverse_string(input_string):
    reversed_string = input_string[::-1]
    return reversed_string

user_input = input("Enter a string: ")
reversed_result = reverse_string(user_input)
print("Reversed string:", reversed_result)
```

```
Enter a string:  Python is a unique programming language which is extensively
used in data science and artificial intelligence
```

```
Reversed string: ecnegilletni laicifitra dna ecneics atad ni desu ylevisnetxe si
hcihw egaugnal gnimmargorp euqinu a si nohtyP
```

[2]:
```
#Question 2: Check if a string is a palindrome.
def is_palindrome(input_string):
    cleaned_string = input_string.lower().replace(" ", "")  # Remove spaces and␣
 ↪make lowercase
    reversed_string = cleaned_string[::-1]
    return cleaned_string == reversed_string

user_input = input("Enter a string: ")
if is_palindrome(user_input):
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

```
Enter a string:  Malayalam
```

```
The string is a palindrome.
```

[3]:
```
#Question 3: Convert a string to uppercase.

original_string = "Hello, World!"
uppercase_string = original_string.upper()
print(uppercase_string)  # Output: HELLO, WORLD!
```

```
HELLO, WORLD!
```

[5]:
```python
#Question 4 Convert a string to lowercase.

original_string = "All programming languages have their own syntax!"
lowercase_string = original_string.lower()
print(lowercase_string)
```

```
all programming languages have their own syntax!
```

[7]:
```python
#Question 5. Count the number of vowels in a string.

def counting_vowels(input_string):
    vowels = "aeiouAEIOU"  # List of vowel characters
    vowel_count = 0

    for char in input_string:
        if char in vowels:
            vowel_count += 1

    return vowel_count

user_input = input("Enter a string: ")
vowel_count = counting_vowels(user_input)
print("Number of vowels:", vowel_count)
```

```
Enter a string:  English is the official language of the United States of
America

Number of vowels: 24
```

[8]:
```python
#Question 6. Count the number of consonants in a string.

def count_consonants(input_string):
    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"  # List of␣
 ↪consonant characters
    consonant_count = 0

    for char in input_string:
        if char in consonants:
            consonant_count += 1

    return consonant_count

user_input = input("Enter a string: ")
consonant_count = count_consonants(user_input)
print("Number of consonants:", consonant_count)
```

```
Enter a string:  The earth has in total 7 continents which are surrouned by
water bodies

Number of consonants: 36
```

[9]:
```python
#Question 7 .Remove all whitespaces from a string.

original_string = "Python is a very popular programming language."
no_whitespace_string = original_string.replace(" ", "").replace("\t", "").
 ↪replace("\n", "")
print(no_whitespace_string)
```

```
Pythonisaverypopularprogramminglanguage.
```

[10]:
```python
#Question 8 : Find the length of a string without using the `len()` function.

def find_string_length(input_string):
    length = 0
    for _ in input_string:
        length += 1
    return length

user_input = input("Enter a string: ")
string_length = find_string_length(user_input)
print("Length of the string:", string_length)
```

```
Enter a string:  Python is very useful in data science and machine learning

Length of the string: 58
```

[11]:
```python
#QUestion 9 :  Check if a string contains a specific word.

def contains_word(input_string, target_word):
    return target_word in input_string

user_input = input("Enter a string: ")
search_word = input("Enter the word to search for: ")

if contains_word(user_input, search_word):
    print(f"The string contains the word '{search_word}'.")
else:
    print(f"The string does not contain the word '{search_word}'.")
```

```
Enter a string:  Python is different from other languages
Enter the word to search for:  r

The string contains the word 'r'.
```

[12]:
```python
#Question 10. Replace a word in a string with another word.

def replace_word(input_string, old_word, new_word):
    return input_string.replace(old_word, new_word)

user_input = input("Enter a string: ")
old_word = input("Enter the word to replace: ")
new_word = input("Enter the replacement word: ")

new_string = replace_word(user_input, old_word, new_word)
print("Modified string:", new_string)
```

```
Enter a string:  Artificial Intelligence
Enter the word to replace:  l
Enter the replacement word:  el
```

```
Modified string: Artificiael Inteeleligence
```

[15]:
```python
#Question 11. Count the occurrences of a word in a string.

def count_word_occurrences(string, target_word):
    words = string.split()
    count = 0
    for word in words:
        if word == target_word:
            count += 1
    return count

user_input = input("Enter a string: ")
search_word = input("Enter the word to count: ")

word_count = count_word_occurrences(user_input, search_word)
print(f"The word '{search_word}' appears {word_count} times in the string.")
```

```
Enter a string:  The entropy of the universe is always increasing according to
the second law of thermodynamics
Enter the word to count:  the
```

```
The word 'the' appears 2 times in the string.
```

[16]:
```python
#Questio 12 : Find the first occurrence of a word in a string.

def find_first_occurrence(input_string, target_word):
    index = input_string.find(target_word)
    return index

user_input = input("Enter a string: ")
search_word = input("Enter the word to search for: ")
```

```
index = find_first_occurrence(user_input, search_word)

if index != -1:
    print(f"The word '{search_word}' first appears at index {index}.")
else:
    print(f"The word '{search_word}' is not found in the string.")
```

```
Enter a string:  It is very important to understand the basics of statistics for
data science
Enter the word to search for:  is

The word 'is' first appears at index 3.
```

[20]:
```
#Question 13. Find the last occurrence of a word in a string.

def find_last_occurrence(input_string, target_word):
    index = input_string.rfind(target_word)
    return index

user_input = input("Enter a string: ")
search_word = input("Enter the word to search for: ")

index = find_last_occurrence(user_input, search_word)

if index != -1:
    print(f"The word '{search_word}' last appears at index {index}.")
else:
    print(f"The word '{search_word}' is not found in the string.")
```

```
Enter a string:  A string contains a sequence of words
Enter the word to search for:  word

The word 'word' last appears at index 32.
```

[21]:
```
#Question 14: 14. Split a string into a list of words.

def split_string(input_string):
    words_list = input_string.split()
    return words_list

user_input = input("Enter a string: ")
split_words = split_string(user_input)
print("List of words:", split_words)
```

```
Enter a string:  Both plants and animals reqwire nutrients to survive.

List of words: ['Both', 'plants', 'and', 'animals', 'reqwire', 'nutrients',
'to', 'survive.']
```

[26]:
```
#Question 15. Join a list of words into a string.

def join_words(word_list):
    joined_string = ' '.join(word_list)  # Using space as separator
    return joined_string

word_list = ['Statistics', 'is_', 'a', 'branch', 'of', 'Mathematics']
joined_string = join_words(word_list)
print("Joined string:", joined_string)
```

```
Joined string: Statistics is_ a branch of Mathematics
```

[27]:
```
#Question 16. Convert a string where words are separated by spaces to one where␣
 ↪words
#are separated by underscores.

def convert_to_underscore(input_string):
    underscore_string = input_string.replace(' ', '_')
    return underscore_string

user_input = input("Enter a string with spaces: ")
underscored_string = convert_to_underscore(user_input)
print("String with underscores:", underscored_string)
```

```
Enter a string with spaces:  London is the capital city of England
String with underscores: London_is_the_capital_city_of_England
```

[3]:
```
#Question 17. Check if a string starts with a specific word or phrase.
def starts_with_phrase(input_string, phrase):
    return input_string.startswith(phrase)

test_string = "Data Science helps make data driven decisions"
phrase_to_check = "Data"

if starts_with_phrase(test_string, phrase_to_check):
    print(f"The string starts with '{phrase_to_check}'.")
else:
    print(f"The string does not start with '{phrase_to_check}'.")
```

```
The string starts with 'Data'.
```

[5]:
```
#Question 18. Check if a string ends with a specific word or phrase.

def ends_with_phrase(input_string, phrase):
    return input_string.endswith(phrase)

test_string = "Artificial Intelligence?"
```

```
phrase_to_check = "Ending word?"

if ends_with_phrase(test_string, phrase_to_check):
    print(f"The string ends with '{phrase_to_check}'.")
else:
    print(f"The string does not end with '{phrase_to_check}'.")
```

```
The string does not end with 'Ending word?'.
```

[6]:
```
#Question. 19. Convert a string to title case (e.g., "hello world" to "Hello␣
 ↪World").

original_string = "Data Science "
title_case_string = original_string.title()

print(title_case_string)
```

```
Data Science
```

[8]:
```
#Question 20. Find the longest word in a string.


def find_longest_word(input_string):
    words = input_string.split()
    longest_word = max(words, key=len)
    return longest_word

test_string = "The longest word in English is␣
 ↪pneumonoultramicroscopicsilicovolcanoconiosis"
longest_word = find_longest_word(test_string)

print(f"The longest word is: {longest_word}")
```

```
The longest word is: pneumonoultramicroscopicsilicovolcanoconiosis
```

[5]:
```
#Question 21. Find the shortest word in a string.

def find_shortest_word(input_string):
    words = input_string.split()
    shortest_word = words[0]
    shortest_length = len(shortest_word)

    for word in words:
        if len(word) < shortest_length:
            shortest_word = word
            shortest_length = len(word)
```

```
    return shortest_word
input_string = "There are a number of species of organisms with similar␣
 ↪features."
shortest_word = find_shortest_word(input_string)
print("Shortest word:", shortest_word)
```

```
Shortest word: a
```

[6]:
```
#Question 22. Reverse the order of words in a string.

def reverse_words(input_string):
    words = input_string.split()  # Step 1
    reversed_words = words[::-1]  # Step 2
    reversed_string = ' '.join(reversed_words)  # Step 3
    return reversed_string

input_string = "The biggest problem in the entire world is poverty"
reversed_string = reverse_words(input_string)
print("Reversed string:", reversed_string)
```

```
Reversed string: poverty is world entire the in problem biggest The
```

[8]:
```
#Question 23. Check if a string is alphanumeric.

def is_alphanumeric(input_string):
    return input_string.isalnum()

# Example usage
test_string = "This is the year 2023"
result = is_alphanumeric(test_string)
if result:
    print("The string is alphanumeric.")
else:
    print("The string is not alphanumeric.")
```

```
The string is not alphanumeric.
```

[10]:
```
#Question 24 Extract all digits from a string.

import re

def extract_digits(input_string):
    digits = re.findall(r'\d', input_string)
    return ''.join(digits)

test_string = "The previous year was 2022"
result = extract_digits(test_string)
```

```
print("Extracted digits:", result)
```

```
Extracted digits: 2022
```

[12]:
```
#Question 25. Extract all alphabets from a string.

import re

def extract_alphabets(input_string):
    alphabets = re.findall(r'[a-zA-Z]', input_string)
    return ''.join(alphabets)

test_string = "The year 2020 was the year of covid"
result = extract_alphabets(test_string)
print("Extracted alphabets:", result)
```

```
Extracted alphabets: Theyearwastheyearofcovid
```

[1]:
```
#26 Count the number of uppercase letters in a string:

def count_uppercase_letters(string):
    count = 0
    for char in string:
        if char.isupper():
            count += 1
    return count

# Example usage:
text = "Hello World"
uppercase_count = count_uppercase_letters(text)
print("Uppercase letters count:", uppercase_count)
```

```
Uppercase letters count: 2
```

[2]:
```
#27 Count the number of lowercase letters in a string:

def count_lowercase_letters(string):
    count = 0
    for char in string:
        if char.islower():
            count += 1
    return count

# Example usage:
text = "Hello World"
lowercase_count = count_lowercase_letters(text)
print("Lowercase letters count:", lowercase_count)
```

```
Lowercase letters count: 8
```

[3]:
```python
#28 Swap the case of each character in a string:

def swap_case(string):
    return string.swapcase()

# Example usage:
text = "Hello World"
swapped_text = swap_case(text)
print("Swapped case:", swapped_text)
```

```
Swapped case: hELLO wORLD
```

[4]:
```python
#28 Remove a specific word from a string:

def remove_word(string, word):
    return string.replace(word, "")

# Example usage:
text = "This is a sample sentence."
word_to_remove = "sample"
new_text = remove_word(text, word_to_remove)
print("Modified text:", new_text)
```

```
Modified text: This is a  sentence.
```

[5]:
```python
#30 Check if a string is a valid email address (basic check):


import re

def is_valid_email(email):
    # This is a basic regex pattern for email validation.
    pattern = r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"
    return re.match(pattern, email) is not None

# Example usage:
email = "example@email.com"
valid = is_valid_email(email)
print("Is valid email:", valid)
```

```
Is valid email: True
```

[6]:
```python
#31 Extract the username from an email address string:

def extract_username(email):
    return email.split('@')[0]
```

```
# Example usage:
email = "example@email.com"
username = extract_username(email)
print("Username:", username)
```

```
Username: example
```

[7]:
```
#32 Extract the domain name from an email address string:

def extract_domain(email):
    return email.split('@')[1]

# Example usage:
email = "example@email.com"
domain = extract_domain(email)
print("Domain:", domain)
```

```
Domain: email.com
```

[8]:
```
#33 Replace multiple spaces in a string with a single space:

def replace_multiple_spaces(string):
    return ' '.join(string.split())

# Example usage:
text = "This   is   a   string   with   multiple   spaces."
cleaned_text = replace_multiple_spaces(text)
print("Cleaned text:", cleaned_text)
```

```
Cleaned text: This is a string with multiple spaces.
```

[9]:
```
#34 Check if a string is a valid URL:

import re

def is_valid_url(url):
    # This is a basic regex pattern for URL validation.
    pattern = r"^(http|https)://[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}/?.*$"
    return re.match(pattern, url) is not None

# Example usage:
url = "https://www.example.com"
valid = is_valid_url(url)
print("Is valid URL:", valid)
```

```
Is valid URL: True
```

[10]:
```python
#35 Extract the protocol (http or https) from a URL string:

def extract_protocol(url):
    return url.split('://')[0]

# Example usage:
url = "https://www.example.com"
protocol = extract_protocol(url)
print("Protocol:", protocol)
```

```
Protocol: https
```

[11]:
```python
#36 Find the frequency of each character in a string:

from collections import Counter

def character_frequency(string):
    return Counter(string)

# Example usage:
text = "Hello, World!"
frequency = character_frequency(text)
print("Character frequency:", frequency)
```

```
Character frequency: Counter({'l': 3, 'o': 2, 'H': 1, 'e': 1, ',': 1, ' ': 1,
'W': 1, 'r': 1, 'd': 1, '!': 1})
```

[ ]:
```python
#Remove all punctuation from a string:

import string

def remove_punctuation(string):
 return string.translate(str.maketrans('', '', string.punctuation))

# Example usage:
text = "Hello, World!"
cleaned_text = remove_punctuation(text)
print("Cleaned text:", cleaned_text)
```

[17]:
```python
#38 Check if a string contains only digits:

def contains_only_digits(string):
    return string.isdigit()

# Example usage:
text = "12345"
is_digits = contains_only_digits(text)
```

```
print("Contains only digits:", is_digits)
```

```
Contains only digits: True
```

[18]:
```
#39. Check if a string contains only alphabets.

def contains_only_alphabets(string):
    return string.isalpha()

# Example usage:
text = "Hello"
is_alphabets = contains_only_alphabets(text)
print("Contains only alphabets:", is_alphabets)
```

```
Contains only alphabets: True
```

[19]:
```
#40 Convert a string to a list of characters:

def string_to_list(string):
    return list(string)

# Example usage:
text = "Hello"
char_list = string_to_list(text)
print("List of characters:", char_list)
```

```
List of characters: ['H', 'e', 'l', 'l', 'o']
```

[20]:
```
#41 Check if two strings are anagrams:

def are_anagrams(str1, str2):
    # Remove spaces and convert to lowercase for case-insensitive comparison
    str1 = str1.replace(" ", "").lower()
    str2 = str2.replace(" ", "").lower()

    return sorted(str1) == sorted(str2)

# Example usage:
string1 = "listen"
string2 = "silent"
are_anagram = are_anagrams(string1, string2)
print("Are anagrams:", are_anagram)

#42 Encode a string using a Caesar cipher:

def caesar_cipher_encode(text, shift):
    encoded_text = ""
```

```
    for char in text:
        if char.isalpha():
            shift_amount = 65 if char.isupper() else 97
            encoded_char = chr((ord(char) - shift_amount + shift) % 26 +␣
 ↪shift_amount)
        else:
            encoded_char = char
        encoded_text += encoded_char
    return encoded_text

# Example usage:
text = "Hello, World!"
shift = 3
encoded_text = caesar_cipher_encode(text, shift)
print("Encoded text:", encoded_text)

#43 Decode a Caesar cipher encoded string:

def caesar_cipher_decode(text, shift):
    return caesar_cipher_encode(text, -shift)

# Example usage:
encoded_text = "Khoor, Zruog!"
shift = 3
decoded_text = caesar_cipher_decode(encoded_text, shift)
print("Decoded text:", decoded_text)

#44Find the most frequent word in a string:

def most_frequent_word(text):
    words = text.split()
    word_counts = {}
    for word in words:
        word_counts[word] = word_counts.get(word, 0) + 1
    most_frequent = max(word_counts, key=word_counts.get)
    return most_frequent

# Example usage:
text = "This is a sample sample sentence with sample words."
most_frequent = most_frequent_word(text)
print("Most frequent word:", most_frequent)

#45 Find all unique words in a string:

def unique_words(text):
    words = text.split()
    return list(set(words))
```

```
# Example usage:
text = "This is a sample sentence with repeated words. This is a sample."
unique = unique_words(text)
print("Unique words:", unique)

#46 Count the number of syllables in a string (a basic approach):

def count_syllables(word):
    vowels = "AEIOUaeiou"
    count = 0
    in_vowel_group = False

    for char in word:
        if char in vowels:
            if not in_vowel_group:
                count += 1
            in_vowel_group = True
        else:
            in_vowel_group = False

    return count

def count_syllables_in_text(text):
    words = text.split()
    total_syllables = sum(count_syllables(word) for word in words)
    return total_syllables

# Example usage:
text = "Hello, this is a simple example."
syllable_count = count_syllables_in_text(text)
print("Syllable count:", syllable_count)
```

```
Are anagrams: True
Encoded text: Khoor, Zruog!
Decoded text: Hello, World!
Most frequent word: sample
Unique words: ['a', 'sample.', 'words.', 'repeated', 'This', 'with', 'is',
'sample', 'sentence']
Syllable count: 10
```

[25]:
```
#47 Check if a string contains any special characters:

import string

def contains_special_characters(text):
    special_characters = set(string.punctuation)
```

```
    return any(char in special_characters for char in text)

# Example usage:
text = "Hello, World!"
contains_special = contains_special_characters(text)
print("Contains special characters:", contains_special)

#48 Remove the nth word from a string:

def remove_nth_word(text, n):
    words = text.split()
    if n >= 1 and n <= len(words):
        del words[n - 1]
    return ' '.join(words)

# Example usage:
text = "This is a sample sentence."
n = 3
modified_text = remove_nth_word(text, n)
print("Modified text:", modified_text)

#49 Insert a word at the nth position in a string:

def insert_word_at_position(text, word, n):
    words = text.split()
    if n >= 0 and n <= len(words):
        words.insert(n, word)
    return ' '.join(words)

# Example usage:
text = "This is a sample sentence."
word_to_insert = "new"
position = 2
modified_text = insert_word_at_position(text, word_to_insert, position)
print("Modified text:", modified_text)

#50 Convert a CSV string to a list of lists:

import csv

def csv_string_to_list(csv_string):
    # Assuming the CSV string has rows separated by newlines and columns␣
 ↪separated by commas.
    # You can adjust the delimiter and newline characters as needed.
    csv_list = list(csv.reader(csv_string.splitlines(), delimiter=','))
    return csv_list
```

```
# Example usage:
csv_string = "John,Doe,30\nAlice,Smith,25\nBob,Johnson,35"
csv_list = csv_string_to_list(csv_string)
print("CSV List:")
for row in csv_list:
    print(row)
```

```
Contains special characters: True
Modified text: This is sample sentence.
Modified text: This is new a sample sentence.
CSV List:
['John', 'Doe', '30']
['Alice', 'Smith', '25']
['Bob', 'Johnson', '35']
```

[27]:
```
#List Based Practice Problem :

#1Create a list with integers from 1 to 10:

my_list = list(range(1, 11))
print(my_list)

#2 Find the length of a list without using the len() function:

def find_length(lst):
    count = 0
    for _ in lst:
        count += 1
    return count

my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
length = find_length(my_list)
print("Length of the list:", length)


#3Append an element to the end of a list:

my_list = [1, 2, 3, 4, 5]
my_list.append(6)
print(my_list)

#4 Insert an element at a specific index in a list:

my_list = [1, 2, 3, 5]
my_list.insert(3, 4)
print(my_list)
```

```python
#5 Remove an element from a list by its value:

my_list = [1, 2, 3, 4, 5]
my_list.remove(3)
print(my_list)

#6 Remove an element from a list by its index:

my_list = [1, 2, 3, 4, 5]
removed_element = my_list.pop(2)
print("Removed element:", removed_element)
print("Updated list:", my_list)

#7 Check if an element exists in a list:

my_list = [1, 2, 3, 4, 5]
element_to_check = 3
if element_to_check in my_list:
    print(f"{element_to_check} exists in the list.")
else:
    print(f"{element_to_check} does not exist in the list.")

#8 Find the index of the first occurrence of an element in a list:

my_list = [1, 2, 3, 4, 5, 3, 6]
element_to_find = 3
if element_to_find in my_list:
    index = my_list.index(element_to_find)
    print(f"The first occurrence of {element_to_find} is at index {index}.")
else:
    print(f"{element_to_find} is not in the list.")

#9 Count the occurrences of an element in a list:

my_list = [1, 2, 3, 4, 5, 3, 6]
element_to_count = 3
count = my_list.count(element_to_count)
print(f"{element_to_count} occurs {count} times in the list.")

#10 Reverse the order of elements in a list:

my_list = [1, 2, 3, 4, 5]
my_list.reverse()
print(my_list)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
Length of the list: 10
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5]
[1, 2, 4, 5]
Removed element: 3
Updated list: [1, 2, 4, 5]
3 exists in the list.
The first occurrence of 3 is at index 2.
3 occurs 2 times in the list.
[5, 4, 3, 2, 1]
```

[28]:
```python
#11Sort a list in ascending order:

my_list = [5, 2, 9, 1, 8, 3]
my_list.sort()
print(my_list)

#12Sort a list in descending order:

my_list = [5, 2, 9, 1, 8, 3]
my_list.sort(reverse=True)
print(my_list)

#13 Create a list of even numbers from 1 to 20:

even_numbers = list(range(2, 21, 2))
print(even_numbers)

#14 Create a list of odd numbers from 1 to 20:

odd_numbers = list(range(1, 21, 2))
print(odd_numbers)

#15 Find the sum of all elements in a list:

my_list = [1, 2, 3, 4, 5]
total = sum(my_list)
print("Sum:", total)

#16 Find the maximum value in a list:

my_list = [5, 2, 9, 1, 8, 3]
max_value = max(my_list)
print("Maximum value:", max_value)

#17 Find the minimum value in a list:
```

```
my_list = [5, 2, 9, 1, 8, 3]
min_value = min(my_list)
print("Minimum value:", min_value)

#18 Create a list of squares of numbers from 1 to 10:

squares = [x**2 for x in range(1, 11)]
print(squares)

#19 Create a list of random numbers:

import random

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

#20 Remove duplicates from a list:

my_list = [1, 2, 2, 3, 4, 4, 5]
unique_list = list(set(my_list))
print(unique_list)
```

```
[1, 2, 3, 5, 8, 9]
[9, 8, 5, 3, 2, 1]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
Sum: 15
Maximum value: 9
Minimum value: 1
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
[5, 95, 28, 72, 67, 92, 74, 69, 66, 55]
[1, 2, 3, 4, 5]
```

[29]:
```
#21 Find the common elements between two lists:

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

#22 Find the difference between two lists:

list1 = [1, 2, 3, 4, 5]
list2 = [3, 4, 5, 6, 7]
difference = list(set(list1) - set(list2))
print("Difference:", difference)
```

```python
#23 Merge two lists:

list1 = [1, 2, 3]
list2 = [4, 5, 6]
merged_list = list1 + list2
print("Merged list:", merged_list)

#24 Multiply all elements in a list by 2:

my_list = [1, 2, 3, 4, 5]
doubled_list = [x * 2 for x in my_list]
print("Doubled list:", doubled_list)

#25 Filter out all even numbers from a list:

my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
filtered_list = [x for x in my_list if x % 2 != 0]
print("Filtered list (odd numbers):", filtered_list)
```

```
Common elements: [3, 4, 5]
Difference: [1, 2]
Merged list: [1, 2, 3, 4, 5, 6]
Doubled list: [2, 4, 6, 8, 10]
Filtered list (odd numbers): [1, 3, 5, 7, 9]
```

[30]:
```python
#26 Convert a list of strings to a list of integers:

string_list = ["1", "2", "3", "4", "5"]
integer_list = list(map(int, string_list))
print("List of integers:", integer_list)

#27 Convert a list of integers to a list of strings:


integer_list = [1, 2, 3, 4, 5]
string_list = list(map(str, integer_list))
print("List of strings:", string_list)

#28 Flatten a nested list:


nested_list = [[1, 2, 3], [4, 5], [6, 7, 8]]
flattened_list = [item for sublist in nested_list for item in sublist]
print("Flattened list:", flattened_list)

#29 Create a list of the first 10 Fibonacci numbers:
```

```python
def generate_fibonacci(n):
    fibonacci_list = [0, 1]
    while len(fibonacci_list) < n:
        next_num = fibonacci_list[-1] + fibonacci_list[-2]
        fibonacci_list.append(next_num)
    return fibonacci_list

fibonacci_sequence = generate_fibonacci(10)
print("Fibonacci sequence:", fibonacci_sequence)

#30 Check if a list is sorted:

def is_sorted(lst):
    return all(lst[i] <= lst[i + 1] for i in range(len(lst) - 1))

my_list = [1, 2, 3, 4, 5]
sorted_check = is_sorted(my_list)
print("Is the list sorted:", sorted_check)


#31 Rotate a list to the left by n positions:

def rotate_left(lst, n):
    n = n % len(lst)  # Ensure n is within the list length
    return lst[n:] + lst[:n]

my_list = [1, 2, 3, 4, 5]
rotated_list = rotate_left(my_list, 2)
print("Rotated list to the left:", rotated_list)

#32 Rotate a list to the right by n positions:

def rotate_right(lst, n):
    n = n % len(lst)  # Ensure n is within the list length
    return lst[-n:] + lst[:-n]

my_list = [1, 2, 3, 4, 5]
rotated_list = rotate_right(my_list, 2)
print("Rotated list to the right:", rotated_list)

#33 Create a list of prime numbers up to 50:

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

prime_numbers = [num for num in range(2, 51) if is_prime(num)]
print("Prime numbers up to 50:", prime_numbers)

#34 Split a list into chunks of size n:

def chunk_list(lst, n):
    return [lst[i:i + n] for i in range(0, len(lst), n)]

my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
chunked_list = chunk_list(my_list, 3)
print("Chunked list:", chunked_list)

#35 Find the second largest number in a list:

def second_largest(lst):
    unique_sorted = sorted(set(lst), reverse=True)
    if len(unique_sorted) < 2:
        return None
    return unique_sorted[1]

my_list = [5, 2, 9, 1, 8, 3]
second_largest_num = second_largest(my_list)
print("Second largest number:", second_largest_num)
```

```
List of integers: [1, 2, 3, 4, 5]
List of strings: ['1', '2', '3', '4', '5']
Flattened list: [1, 2, 3, 4, 5, 6, 7, 8]
Fibonacci sequence: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
Is the list sorted: True
Rotated list to the left: [3, 4, 5, 1, 2]
Rotated list to the right: [4, 5, 1, 2, 3]
Prime numbers up to 50: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
Chunked list: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
Second largest number: 8
```

[33]:
```
#36 Replace every element in a list with its square:

my_list = [1, 2, 3, 4, 5]
squared_list = [x**2 for x in my_list]
print("Squared list:", squared_list)

#37 Convert a list to a dictionary where list elements become keys and their␣
 ↪indices become values:
```

```python
my_list = ["apple", "banana", "cherry"]
dict_from_list = {element: index for index, element in enumerate(my_list)}
print("Dictionary from list:", dict_from_list)

#38 Shuffle the elements of a list randomly:

import random

my_list = [1, 2, 3, 4, 5]
random.shuffle(my_list)
print("Shuffled list:", my_list)

#39 Create a list of the first 10 factorial numbers:

import math

factorial_numbers = [math.factorial(n) for n in range(10)]
print("Factorial numbers:", factorial_numbers)

#40 Check if two lists have at least one element in common:

list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]
have_common_element = any(item in list1 for item in list2)
print("Do the lists have a common element:", have_common_element)

#41 Remove all elements from a list:

my_list = [1, 2, 3, 4, 5]
my_list.clear()
print("Empty list:", my_list)

#42 Replace negative numbers in a list with 0:

my_list = [1, -2, 3, -4, 5]
non_negative_list = [max(0, x) for x in my_list]
print("Non-negative list:", non_negative_list)

#43 Convert a string into a list of words:

text = "This is a sample sentence."
word_list = text.split()
print("List of words:", word_list)

#44 Convert a list of words into a string:

word_list = ["This", "is", "a", "list", "of", "words"]
```

```
text = ' '.join(word_list)
print("String:", text)
```

```
Squared list: [1, 4, 9, 16, 25]
Dictionary from list: {'apple': 0, 'banana': 1, 'cherry': 2}
Shuffled list: [1, 3, 2, 4, 5]
Factorial numbers: [1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880]
Do the lists have a common element: True
Empty list: []
Non-negative list: [1, 0, 3, 0, 5]
List of words: ['This', 'is', 'a', 'sample', 'sentence.']
String: This is a list of words
```

[34]:
```
#45 Create a list of the first n powers of 2:

def powers_of_2(n):
    return [2 ** i for i in range(n)]

n = 5
power_list = powers_of_2(n)
print("Powers of 2:", power_list)

#46 Find the longest string in a list of strings:

string_list = ["apple", "banana", "cherry", "date"]
longest_string = max(string_list, key=len)
print("Longest string:", longest_string)

#47 Find the shortest string in a list of strings:

string_list = ["apple", "banana", "cherry", "date"]
shortest_string = min(string_list, key=len)
print("Shortest string:", shortest_string)

#48 Create a list of the first n triangular numbers:

def triangular_numbers(n):
    return [i * (i + 1) // 2 for i in range(n)]

n = 5
triangular_list = triangular_numbers(n)
print("Triangular numbers:", triangular_list)

#49 Check if a list contains another list as a subsequence:

def is_subsequence(subsequence, sequence):
    i = 0
```

```
    for item in sequence:
        if item == subsequence[i]:
            i += 1
            if i == len(subsequence):
                return True
    return False

list1 = [1, 2, 3, 4, 5, 6]
list2 = [2, 4, 6]
is_subseq = is_subsequence(list2, list1)
print("Is list2 a subsequence of list1:", is_subseq)

#50 Swap two elements in a list by their indices:

def swap_elements(lst, index1, index2):
    lst[index1], lst[index2] = lst[index2], lst[index1]

my_list = [1, 2, 3, 4, 5]
swap_elements(my_list, 1, 3)
print("List after swapping elements:", my_list)
```

```
Powers of 2: [1, 2, 4, 8, 16]
Longest string: banana
Shortest string: date
Triangular numbers: [0, 1, 3, 6, 10]
Is list2 a subsequence of list1: True
List after swapping elements: [1, 4, 3, 2, 5]
```

[35]:
```
#Tuple Based Practice Problem :

#1 Create a tuple with integers from 1 to 5:

my_tuple = (1, 2, 3, 4, 5)
print(my_tuple)

#2 Access the third element of a tuple:

my_tuple = (1, 2, 3, 4, 5)
third_element = my_tuple[2]
print("Third element:", third_element)

#3 Find the length of a tuple without using the len() function:

def find_length(my_tuple):
    count = 0
    for _ in my_tuple:
        count += 1
```

```
    return count

my_tuple = (1, 2, 3, 4, 5)
length = find_length(my_tuple)
print("Length of the tuple:", length)

#4 Count the occurrences of an element in a tuple:

my_tuple = (1, 2, 2, 3, 2, 4, 5, 2)
element_to_count = 2
count = my_tuple.count(element_to_count)
print(f"{element_to_count} occurs {count} times in the tuple.")

#5 Find the index of the first occurrence of an element in a tuple:

my_tuple = (1, 2, 3, 4, 5, 3, 6)
element_to_find = 3
if element_to_find in my_tuple:
    index = my_tuple.index(element_to_find)
    print(f"The first occurrence of {element_to_find} is at index {index}.")
else:
    print(f"{element_to_find} is not in the tuple.")

#6 Check if an element exists in a tuple:

my_tuple = (1, 2, 3, 4, 5)
element_to_check = 3
if element_to_check in my_tuple:
    print(f"{element_to_check} exists in the tuple.")
else:
    print(f"{element_to_check} does not exist in the tuple.")

#7 Convert a tuple to a list:

my_tuple = (1, 2, 3, 4, 5)
my_list = list(my_tuple)
print("Tuple converted to a list:", my_list)

#8 Convert a list to a tuple:

my_list = [1, 2, 3, 4, 5]
my_tuple = tuple(my_list)
print("List converted to a tuple:", my_tuple)
#9 Unpack the elements of a tuple into variables:

my_tuple = (1, 2, 3)
a, b, c = my_tuple
```

```
print("Unpacked variables:", a, b, c)

#10 Create a tuple of even numbers from 1 to 10:

even_numbers_tuple = tuple(range(2, 11, 2))
print("Tuple of even numbers:", even_numbers_tuple)
```

```
(1, 2, 3, 4, 5)
Third element: 3
Length of the tuple: 5
2 occurs 4 times in the tuple.
The first occurrence of 3 is at index 2.
3 exists in the tuple.
Tuple converted to a list: [1, 2, 3, 4, 5]
List converted to a tuple: (1, 2, 3, 4, 5)
Unpacked variables: 1 2 3
Tuple of even numbers: (2, 4, 6, 8, 10)
```

[36]:
```
#11 Create a tuple of odd numbers from 1 to 10:

odd_numbers_tuple = tuple(range(1, 11, 2))
print("Tuple of odd numbers:", odd_numbers_tuple)

#12 Concatenate two tuples:

tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
concatenated_tuple = tuple1 + tuple2
print("Concatenated tuple:", concatenated_tuple)

#13 Repeat a tuple three times:

my_tuple = (1, 2, 3)
repeated_tuple = my_tuple * 3
print("Repeated tuple:", repeated_tuple)

#14 Check if a tuple is empty:

empty_tuple = ()
if not empty_tuple:
    print("The tuple is empty.")
else:
    print("The tuple is not empty.")

#15 Create a nested tuple:

nested_tuple = ((1, 2, 3), (4, 5, 6), (7, 8, 9))
```

```
print("Nested tuple:", nested_tuple)

#16 Access the first element of a nested tuple:

nested_tuple = ((1, 2, 3), (4, 5, 6), (7, 8, 9))
first_element = nested_tuple[0]
print("First element of the nested tuple:", first_element)

#17 Create a tuple with a single element:

single_element_tuple = (42,)
print("Single-element tuple:", single_element_tuple)

#18 Compare two tuples:

tuple1 = (1, 2, 3)
tuple2 = (1, 2, 3)
if tuple1 == tuple2:
    print("The two tuples are equal.")
else:
    print("The two tuples are not equal.")

    #19 Delete a tuple (tuples are immutable, so they can't be modified or␣
 ↪deleted, but you can delete the reference to it):

my_tuple = (1, 2, 3)
del my_tuple
# Attempting to access my_tuple here would result in an error.

#20 Slice a tuple:

my_tuple = (1, 2, 3, 4, 5)
sliced_tuple = my_tuple[1:4]
print("Sliced tuple:", sliced_tuple)
```

```
Tuple of odd numbers: (1, 3, 5, 7, 9)
Concatenated tuple: (1, 2, 3, 4, 5, 6)
Repeated tuple: (1, 2, 3, 1, 2, 3, 1, 2, 3)
The tuple is empty.
Nested tuple: ((1, 2, 3), (4, 5, 6), (7, 8, 9))
First element of the nested tuple: (1, 2, 3)
Single-element tuple: (42,)
The two tuples are equal.
Sliced tuple: (2, 3, 4)
```

[37]:
```
#21 Find the maximum value in a tuple:
```

```python
my_tuple = (7, 12, 3, 9, 5)
max_value = max(my_tuple)
print("Maximum value:", max_value)

#22 Find the minimum value in a tuple:

my_tuple = (7, 12, 3, 9, 5)
min_value = min(my_tuple)
print("Minimum value:", min_value)

#23 Convert a string to a tuple of characters:

text = "Hello"
char_tuple = tuple(text)
print("Tuple of characters:", char_tuple)

#24 Convert a tuple of characters to a string:

char_tuple = ('H', 'e', 'l', 'l', 'o')
text = ''.join(char_tuple)
print("String:", text)

#25 Create a tuple from multiple data types:

mixed_tuple = (1, "apple", 3.14, True)
print("Mixed tuple:", mixed_tuple)

#26 Check if two tuples are identical:

tuple1 = (1, 2, 3)
tuple2 = (1, 2, 3)
if tuple1 == tuple2:
    print("The two tuples are identical.")
else:
    print("The two tuples are not identical.")

#27 Sort the elements of a tuple:

my_tuple = (7, 12, 3, 9, 5)
sorted_tuple = tuple(sorted(my_tuple))
print("Sorted tuple:", sorted_tuple)

#28 Convert a tuple of integers to a tuple of strings:

int_tuple = (1, 2, 3, 4, 5)
str_tuple = tuple(map(str, int_tuple))
print("Tuple of strings:", str_tuple)
```

```
#29 Convert a tuple of strings to a tuple of integers:

str_tuple = ("1", "2", "3", "4", "5")
int_tuple = tuple(map(int, str_tuple))
print("Tuple of integers:", int_tuple)

#30 Merge two tuples:

tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
merged_tuple = tuple1 + tuple2
print("Merged tuple:", merged_tuple)

#31 Flatten a nested tuple:

nested_tuple = ((1, 2, 3), (4, 5, 6), (7, 8, 9))
flattened_tuple = tuple(item for sublist in nested_tuple for item in sublist)
print("Flattened tuple:", flattened_tuple)
```

```
Maximum value: 12
Minimum value: 3
Tuple of characters: ('H', 'e', 'l', 'l', 'o')
String: Hello
Mixed tuple: (1, 'apple', 3.14, True)
The two tuples are identical.
Sorted tuple: (3, 5, 7, 9, 12)
Tuple of strings: ('1', '2', '3', '4', '5')
Tuple of integers: (1, 2, 3, 4, 5)
Merged tuple: (1, 2, 3, 4, 5, 6)
Flattened tuple: (1, 2, 3, 4, 5, 6, 7, 8, 9)
```

[38]:
```
#32 Create a tuple of the first 5 prime numbers:

prime_numbers = (2, 3, 5, 7, 11)
print("Tuple of prime numbers:", prime_numbers)

#33 Check if a tuple is a palindrome:

def is_palindrome(my_tuple):
    return my_tuple == my_tuple[::-1]

my_tuple = (1, 2, 3, 2, 1)
is_palindrome_check = is_palindrome(my_tuple)
print("Is the tuple a palindrome:", is_palindrome_check)

#34 Create a tuple of squares of numbers from 1 to 5:
```

```python
squares_tuple = tuple(x**2 for x in range(1, 6))
print("Tuple of squares:", squares_tuple)

#35 Filter out all even numbers from a tuple:

my_tuple = (1, 2, 3, 4, 5, 6, 7, 8, 9)
filtered_tuple = tuple(x for x in my_tuple if x % 2 != 0)
print("Filtered tuple (odd numbers):", filtered_tuple)

#36 Multiply all elements in a tuple by 2:

my_tuple = (1, 2, 3, 4, 5)
doubled_tuple = tuple(x * 2 for x in my_tuple)
print("Doubled tuple:", doubled_tuple)

#37 Create a tuple of random numbers:

import random

random_tuple = tuple(random.randint(1, 100) for _ in range(5))
print("Tuple of random numbers:", random_tuple)

#38 Check if a tuple is sorted:

def is_sorted(my_tuple):
    return all(my_tuple[i] <= my_tuple[i + 1] for i in range(len(my_tuple) - 1))

sorted_tuple = (1, 2, 3, 4, 5)
unsorted_tuple = (3, 1, 4, 2, 5)
is_sorted_check = is_sorted(sorted_tuple)
is_sorted_check_unsorted = is_sorted(unsorted_tuple)
print("Is the sorted tuple sorted:", is_sorted_check)
print("Is the unsorted tuple sorted:", is_sorted_check_unsorted)

#39 Rotate a tuple to the left by n positions:

def rotate_left(my_tuple, n):
    n = n % len(my_tuple)
    return my_tuple[n:] + my_tuple[:n]

my_tuple = (1, 2, 3, 4, 5)
rotated_left_tuple = rotate_left(my_tuple, 2)
print("Rotated tuple to the left:", rotated_left_tuple)

#40 Rotate a tuple to the right by n positions:
```

```
def rotate_right(my_tuple, n):
    n = n % len(my_tuple)
    return my_tuple[-n:] + my_tuple[:-n]

my_tuple = (1, 2, 3, 4, 5)
rotated_right_tuple = rotate_right(my_tuple, 2)
print("Rotated tuple to the right:", rotated_right_tuple)

#41 Create a tuple of the first 5 Fibonacci numbers:

def generate_fibonacci(n):
    fibonacci_list = [0, 1]
    while len(fibonacci_list) < n:
        next_num = fibonacci_list[-1] + fibonacci_list[-2]
        fibonacci_list.append(next_num)
    return tuple(fibonacci_list)

fibonacci_sequence = generate_fibonacci(5)
print("Tuple of Fibonacci numbers:", fibonacci_sequence)
```

```
Tuple of prime numbers: (2, 3, 5, 7, 11)
Is the tuple a palindrome: True
Tuple of squares: (1, 4, 9, 16, 25)
Filtered tuple (odd numbers): (1, 3, 5, 7, 9)
Doubled tuple: (2, 4, 6, 8, 10)
Tuple of random numbers: (65, 90, 75, 44, 79)
Is the sorted tuple sorted: True
Is the unsorted tuple sorted: False
Rotated tuple to the left: (3, 4, 5, 1, 2)
Rotated tuple to the right: (4, 5, 1, 2, 3)
Tuple of Fibonacci numbers: (0, 1, 1, 2, 3)
```

[39]:
```
#42 Create a tuple from user input:

user_input = input("Enter a comma-separated list of values: ")
user_tuple = tuple(user_input.split(","))
print("User-created tuple:", user_tuple)

#43 Swap two elements in a tuple:

def swap_elements(my_tuple, index1, index2):
    # Convert the tuple to a list to perform the swap, then convert it back to␣
 ↪a tuple.
    temp_list = list(my_tuple)
    temp_list[index1], temp_list[index2] = temp_list[index2], temp_list[index1]
    return tuple(temp_list)
```

```python
my_tuple = (1, 2, 3, 4, 5)
swapped_tuple = swap_elements(my_tuple, 1, 3)
print("Tuple after swapping elements:", swapped_tuple)

#44 Reverse the elements of a tuple:

my_tuple = (1, 2, 3, 4, 5)
reversed_tuple = tuple(reversed(my_tuple))
print("Reversed tuple:", reversed_tuple)

#45 Create a tuple of the first n powers of 2:

def powers_of_2(n):
    return tuple(2 ** i for i in range(n))

n = 5
power_tuple = powers_of_2(n)
print("Tuple of powers of 2:", power_tuple)

#46 Find the longest string in a tuple of strings:

string_tuple = ("apple", "banana", "cherry", "date")
longest_string = max(string_tuple, key=len)
print("Longest string:", longest_string)

#47 Find the shortest string in a tuple of strings:

string_tuple = ("apple", "banana", "cherry", "date")
shortest_string = min(string_tuple, key=len)
print("Shortest string:", shortest_string)

#48 Create a tuple of the first n triangular numbers:

def triangular_numbers(n):
    return tuple(i * (i + 1) // 2 for i in range(n))

n = 5
triangular_tuple = triangular_numbers(n)
print("Tuple of triangular numbers:", triangular_tuple)

#49 Check if a tuple contains another tuple as a subsequence:

def is_subsequence(subsequence, sequence):
    n = len(subsequence)
    for i in range(len(sequence) - n + 1):
        if sequence[i:i + n] == subsequence:
            return True
```

```
    return False

tuple1 = (1, 2, 3, 4, 5, 6)
tuple2 = (2, 3, 4)
is_subseq = is_subsequence(tuple2, tuple1)
print("Is tuple2 a subsequence of tuple1:", is_subseq)

#50 Create a tuple of alternating 1s and 0s of length n:

def alternating_ones_and_zeros(n):
    return tuple(1 if i % 2 == 0 else 0 for i in range(n))

n = 8
alternating_tuple = alternating_ones_and_zeros(n)
print("Tuple of alternating 1s and 0s:", alternating_tuple)
```

```
Enter a comma-separated list of values:  1,2,3,4,5

User-created tuple: ('1', '2', '3', '4', '5')
Tuple after swapping elements: (1, 4, 3, 2, 5)
Reversed tuple: (5, 4, 3, 2, 1)
Tuple of powers of 2: (1, 2, 4, 8, 16)
Longest string: banana
Shortest string: date
Tuple of triangular numbers: (0, 1, 3, 6, 10)
Is tuple2 a subsequence of tuple1: True
Tuple of alternating 1s and 0s: (1, 0, 1, 0, 1, 0, 1, 0)
```

[40]:
```
#Set Based Practice Problem :

#1 Create a set with integers from 1 to 5:

my_set = {1, 2, 3, 4, 5}
print("Set:", my_set)

#2 Add an element to a set:

my_set = {1, 2, 3}
my_set.add(4)
print("Set after adding an element:", my_set)
#3 Remove an element from a set:

my_set = {1, 2, 3, 4}
my_set.remove(3)
print("Set after removing an element:", my_set)

#4 Check if an element exists in a set:
```

```python
my_set = {1, 2, 3, 4, 5}
element_to_check = 3
if element_to_check in my_set:
    print(f"{element_to_check} exists in the set.")
else:
    print(f"{element_to_check} does not exist in the set.")

#5 Find the length of a set without using the len() function:

my_set = {1, 2, 3, 4, 5}
count = 0
for _ in my_set:
    count += 1
print("Length of the set:", count)

#6 Clear all elements from a set:

my_set = {1, 2, 3, 4, 5}
my_set.clear()
print("Empty set:", my_set)

#7 Create a set of even numbers from 1 to 10:

even_numbers_set = {x for x in range(2, 11, 2)}
print("Set of even numbers:", even_numbers_set)

#8 Create a set of odd numbers from 1 to 10:

odd_numbers_set = {x for x in range(1, 11, 2)}
print("Set of odd numbers:", odd_numbers_set)

#9 Find the union of two sets:

set1 = {1, 2, 3}
set2 = {3, 4, 5}
union_set = set1.union(set2)
print("Union of two sets:", union_set)

#10 Find the intersection of two sets:

set1 = {1, 2, 3}
set2 = {3, 4, 5}
intersection_set = set1.intersection(set2)
print("Intersection of two sets:", intersection_set)

#11 Find the difference between two sets:
```

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
difference_set = set1.difference(set2)
print("Difference between two sets:", difference_set)

#12 Check if a set is a subset of another set:

set1 = {1, 2, 3}
set2 = {1, 2, 3, 4, 5}
is_subset = set1.issubset(set2)
print("Is set1 a subset of set2:", is_subset)

#13 Check if a set is a superset of another set:

set1 = {1, 2, 3, 4, 5}
set2 = {1, 2, 3}
is_superset = set1.issuperset(set2)
print("Is set1 a superset of set2:", is_superset)

#14 Create a set from a list:

my_list = [1, 2, 3, 4, 5]
set_from_list = set(my_list)
print("Set from a list:", set_from_list)

#15 Convert a set to a list:

my_set = {1, 2, 3, 4, 5}
list_from_set = list(my_set)
print("List from a set:", list_from_set)
```

```
Set: {1, 2, 3, 4, 5}
Set after adding an element: {1, 2, 3, 4}
Set after removing an element: {1, 2, 4}
3 exists in the set.
Length of the set: 5
Empty set: set()
Set of even numbers: {2, 4, 6, 8, 10}
Set of odd numbers: {1, 3, 5, 7, 9}
Union of two sets: {1, 2, 3, 4, 5}
Intersection of two sets: {3}
Difference between two sets: {1, 2}
Is set1 a subset of set2: True
Is set1 a superset of set2: True
Set from a list: {1, 2, 3, 4, 5}
List from a set: [1, 2, 3, 4, 5]
```

[ ]:
```python
#16 Remove a random element from a set:

import random

my_set = {1, 2, 3, 4, 5}
random_element = random.choice(list(my_set))
my_set.remove(random_element)
print("Set after removing a random element:", my_set)

#17 Pop an element from a set:

my_set = {1, 2, 3, 4, 5}
popped_element = my_set.pop()
print("Popped element:", popped_element)
print("Set after popping an element:", my_set)

#18 Check if two sets have no elements in common:

set1 = {1, 2, 3}
set2 = {4, 5, 6}
no_common_elements = not bool(set1.intersection(set2))
print("Do the sets have no common elements:", no_common_elements)

#19 Find the symmetric difference between two sets:

set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}
symmetric_difference = set1.symmetric_difference(set2)
print("Symmetric difference between two sets:", symmetric_difference)

#20 Update a set with elements from another set:

set1 = {1, 2, 3}
set2 = {3, 4, 5}
set1.update(set2)
print("Updated set:", set1)

#21 Create a set of the first 5 prime numbers:

prime_numbers_set = {2, 3, 5, 7, 11}
print("Set of prime numbers:", prime_numbers_set)

#22 Check if two sets are identical:

set1 = {1, 2, 3}
set2 = {3, 2, 1}
are_identical = set1 == set2
```

```python
print("Are the two sets identical:", are_identical)

#23 Create a frozen set:

my_frozen_set = frozenset({1, 2, 3, 4, 5})
print("Frozen set:", my_frozen_set)

#24 Check if a set is disjoint with another set:

set1 = {1, 2, 3}
set2 = {4, 5, 6}
are_disjoint = set1.isdisjoint(set2)
print("Are the sets disjoint:", are_disjoint)

#25 Create a set of squares of numbers from 1 to 5:

squares_set = {x**2 for x in range(1, 6)}
print("Set of squares:", squares_set)

#26 Filter out all even numbers from a set:

my_set = {1, 2, 3, 4, 5, 6, 7, 8, 9}
filtered_set = {x for x in my_set if x % 2 != 0}
print("Filtered set (odd numbers):", filtered_set)

#27 Multiply all elements in a set by 2:

my_set = {1, 2, 3, 4, 5}
doubled_set = {x * 2 for x in my_set}
print("Doubled set:", doubled_set)

#28 Create a set of random numbers:

import random

random_set = {random.randint(1, 100) for _ in range(5)}
print("Set of random numbers:", random_set)

#29 Check if a set is empty:

my_set = set()
is_empty = not bool(my_set)
print("Is the set empty:", is_empty)

#30 Create a nested set (hint: use frozenset):

nested_set = {frozenset({1, 2}), frozenset({3, 4})}
```

```
print("Nested set:", nested_set)

#31 Remove an element from a set using the discard method:

my_set = {1, 2, 3, 4, 5}
my_set.discard(3)
print("Set after removing an element using discard:", my_set)
```

```
Set after removing a random element: {1, 2, 4, 5}
Popped element: 1
Set after popping an element: {2, 3, 4, 5}
Do the sets have no common elements: True
Symmetric difference between two sets: {1, 2, 5, 6}
Updated set: {1, 2, 3, 4, 5}
Set of prime numbers: {2, 3, 5, 7, 11}
Are the two sets identical: True
Frozen set: frozenset({1, 2, 3, 4, 5})
Are the sets disjoint: True
Set of squares: {1, 4, 9, 16, 25}
Filtered set (odd numbers): {1, 3, 5, 7, 9}
Doubled set: {2, 4, 6, 8, 10}
Set of random numbers: {39, 44, 55, 61, 30}
Is the set empty: True
Nested set: {frozenset({3, 4}), frozenset({1, 2})}
Set after removing an element using discard: {1, 2, 4, 5}
```

[ ]:
```
#32 Compare two sets:

set1 = {1, 2, 3}
set2 = {3, 4, 5}
are_equal = set1 == set2
print("Are the two sets equal:", are_equal)

#33 Create a set from a string:

text = "hello"
char_set = set(text)
print("Set from a string:", char_set)

#34 Convert a set of strings to a set of integers:

str_set = {"1", "2", "3", "4", "5"}
int_set = {int(x) for x in str_set}
print("Set of integers:", int_set)

#35 Convert a set of integers to a set of strings:
```

```python
int_set = {1, 2, 3, 4, 5}
str_set = {str(x) for x in int_set}
print("Set of strings:", str_set)

#36 Create a set from a tuple:

my_tuple = (1, 2, 3, 4, 5)
set_from_tuple = set(my_tuple)
print("Set from a tuple:", set_from_tuple)

#37 Convert a set to a tuple:

my_set = {1, 2, 3, 4, 5}
tuple_from_set = tuple(my_set)
print("Tuple from a set:", tuple_from_set)

#38 Find the maximum value in a set:

my_set = {7, 12, 3, 9, 5}
max_value = max(my_set)
print("Maximum value in the set:", max_value)

#39 Find the minimum value in a set:

my_set = {7, 12, 3, 9, 5}
min_value = min(my_set)
print("Minimum value in the set:", min_value)

#40 Create a set from user input:

user_input = input("Enter a comma-separated list of values: ")
user_set = set(user_input.split(","))
print("User-created set:", user_set)

#41 Check if the intersection of two sets is empty:

set1 = {1, 2, 3}
set2 = {4, 5, 6}
intersection_empty = not bool(set1.intersection(set2))
print("Is the intersection of two sets empty:", intersection_empty)

#42 Create a set of the first 5 Fibonacci numbers:

def generate_fibonacci(n):
    fibonacci_set = {0, 1}
    while len(fibonacci_set) < n:
        next_num = sum(list(fibonacci_set)[-2:])
```

```
        fibonacci_set.add(next_num)
    return fibonacci_set

fibonacci_set = generate_fibonacci(5)
print("Set of Fibonacci numbers:", fibonacci_set)

#43 Remove duplicates from a list using sets:

my_list = [1, 2, 2, 3, 3, 4, 4, 5]
unique_set = set(my_list)
unique_list = list(unique_set)
print("List with duplicates removed:", unique_list)

#44 Check if two sets have the same elements, regardless of their count:

set1 = {1, 2, 3, 4, 5}
set2 = {5, 4, 3, 2, 1, 1, 2}
have_same_elements = set1 == set2
print("Do the sets have the same elements:", have_same_elements)

#45 Create a set of the first n powers of 2:

def powers_of_2(n):
    return {2 ** i for i in range(n)}

n = 5
power_set = powers_of_2(n)
print("Set of powers of 2:", power_set)

#46 Find the common elements between a set and a list:

my_set = {1, 2, 3, 4, 5}
my_list = [3, 4, 5, 6, 7]
common_elements = my_set.intersection(my_list)
print("Common elements between set and list:", common_elements)

#47 Create a set of the first n triangular numbers:

def triangular_numbers(n):
    return {i * (i + 1) // 2 for i in range(n)}

n = 5
triangular_set = triangular_numbers(n)
print("Set of triangular numbers:", triangular_set)

#48 Check if a set contains another set as a subset:
```

```python
set1 = {1, 2, 3, 4, 5}
set2 = {2, 3, 4}
is_subset = set2.issubset(set1)
print("Is set2 a subset of set1:", is_subset)

#49 Create a set of alternating 1s and 0s of length n:

def alternating_ones_and_zeros(n):
    return {1 if i % 2 == 0 else 0 for i in range(n)}

n = 8
alternating_set = alternating_ones_and_zeros(n)
print("Set of alternating 1s and 0s:", alternating_set)

#50 Merge multiple sets into one:

set1 = {1, 2, 3}
set2 = {3, 4, 5}
set3 = {5, 6, 7}
merged_set = set1.union(set2, set3)
print("Merged set:", merged_set)
```

```
Are the two sets equal: False
Set from a string: {'l', 'h', 'e', 'o'}
Set of integers: {1, 2, 3, 4, 5}
Set of strings: {'5', '2', '1', '4', '3'}
Set from a tuple: {1, 2, 3, 4, 5}
Tuple from a set: (1, 2, 3, 4, 5)
Maximum value in the set: 12
Minimum value in the set: 3

Enter a comma-separated list of values:  3,4,5,6,7,8,9

User-created set: {'6', '9', '5', '8', '7', '4', '3'}
Is the intersection of two sets empty: True
```

[ ]: