assigmnet2

May 28, 2024

[ ]:
```
#Topic 1
```

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

input_string = input("enter a string to reverse:")
reversed_strin= reversed_string(input_string)
print("reversed string :",reversed_strin)
```

```
enter a string to reverse: 14643
reversed string : 34641
```

[2]:
```
# 2. Check if a string is a palindrome.
def is_palindrome(input_string):
    return input_string==input_string[::-1]
input_string="PoojaajooP"
if is_palindrome(input_string):
    print(input_string,"is a palindrome")
else:
    print(input_string,"is not a palindeom")
```

```
PoojaajooP is a palindrome
```

[3]:
```
# 3. Convert a string to uppercase.
input_string="pooja, najardhane"
uppercase_string=input_string.upper()
print("original string : ",input_string)
print("Uppercase string : ",uppercase_string)
```

```
original string :  pooja, najardhane
Uppercase string :  POOJA, NAJARDHANE
```

[4]:
```
# 4.  Convert a string to lowercase.
input_string="POOJA NAJARDHANE"
lowercase_string=input_string.lower()
print("original string : ",input_string)
```

```
print("Uppercase string : ",lowercase_string)
```

```
original string :  POOJA NAJARDHANE
Uppercase string :  pooja najardhane
```

[5]:
```
# 5. Count the number of vowels in a string.
def count_vowels(input_string):
    vowels="aeiouAEIOU"
    count=0
    for char in input_string:
        if char in vowels:
            count+=1
    return count

input_string="Pooja najardhane"
vowel_count= count_vowels(input_string)
print("number of vowels in the string : ", vowel_count)
```

```
number of vowels in the string :  7
```

[9]:
```
 #6. Count the number of consonants in a string.
def count_consonants(string):
    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"
    count = 0
    for char in string:
        if char in consonants:
            count += 1
    return count

# Example usage:
string = "Hello World"
print("Number of consonants:", count_consonants(string))
```

```
Number of consonants: 7
```

[10]:
```
# 7. Remove all whitespaces from a string.
def remove_whitespace(input_string):
    return input_string.replace("  ","")
input_string="Pooja      Najrdhane"
whitespace_string=remove_whitespace(input_string)
print("original string : ", input_string)
print("string without whitespace : ", whitespace_string)
```

```
original string :  Pooja      Najrdhane
string without whitespace :  PoojaNajrdhane
```

[11]:
```python
# 8. Find the length of a string without using the `len()` function.
def string_length(input_string):
    count=0
    for char in input_string:
        count+=1
    return count
input_string="hello"
length = string_length(input_string)
print("lenght of the string : ", length)
```

```
lenght of the string :  5
```

[12]:
```python
# 9. Check if a string contains a specific word.
def contains_word(input_string,word):
    return word in input_string
input_string="pooja najardhane"
search_word="Pooja"
if contains_word(input_string,search_word):
    print("the string contains the word ", search_word)
else:
    print("the string does not contain the word", search_word)
```

```
the string does not contain the word Pooja
```

[13]:
```python
# 10. Replace a word in a string with another word.
def contains_word(input_string, old_word, new_word):
    return input_string.replace(old_word,new_word)
input_str="Pooja ramesh"
old_word="ramesh"
new_word="najardhane"
new_string=contains_word(input_str,old_word,new_word)
print("original string:", input_str)
print("modified string:",new_string)
```

```
original string: Pooja ramesh
modified string: Pooja najardhane
```

[14]:
```python
# 11. Count the occurrences of a word in a string.
def count_word_occurrences(input_string, target_word):
    words = input_string.split()
    count = 0
    for word in words:
        if word == target_word:
            count += 1
    return count
input_str = "pooja, ranesh, pooja"
target_word = "pooja"
```

```
occurrences = count_word_occurrences(input_str, target_word)
print("Number of occurrences of '{}' in the string: {}".format(target_word,␣
 ↪occurrences))
```

```
Number of occurrences of 'pooja' in the string: 1
```

[15]:
```
# 12. Find the first occurrence of a word in a string.
def find_first_occurrence(input_string, target_word):
    first_index = input_string.find(target_word)
    return first_index

input_str = "pooja, ranesh, pooja"
target_word = "pooja"

first_occurrence_index = find_first_occurrence(input_str, target_word)

print("The first occurrence of '{}' is at index: {}".format(target_word,␣
 ↪first_occurrence_index))
```

```
The first occurrence of 'pooja' is at index: 0
```

[16]:
```
# 13. Find the last occurrence of a word in a string.
def find_last_occurrence(input_string, target_word):

    last_index = input_string.rfind(target_word)
    return last_index


input_str = "pooja, ranesh, pooja"
target_word = "pooja"


last_occurrence_index = find_last_occurrence(input_str, target_word)


print("The last occurrence of '{}' is at index: {}".format(target_word,␣
 ↪last_occurrence_index))
```

```
The last occurrence of 'pooja' is at index: 15
```

[17]:
```
# 14. Split a string into a list of words.

input_str = "My name pooja ramesh najardhnae."

words_list = input_str.split()

print(words_list)
```

```
['My', 'name', 'pooja', 'ramesh', 'najardhnae.']
```

[18]:
```
#  15. Join a list of words into a string.

words_list = ['my', 'name', 'is', 'pooja', 'najardhane']

joined_string = ' '.join(words_list)


print(joined_string)
```

```
my name is pooja najardhane
```

[19]:
```
#  16. Convert a string where words are separated by spaces to one where words␣
 ↪are separated by underscores.

input_str = "This is an example string"

words_list = input_str.split()

converted_str = '_'.join(words_list)

print(converted_str)
```

```
This_is_an_example_string
```

[20]:
```
# 17. Check if a string starts with a specific word or phrase.

input_str = "Hello world!"
start_word = "Hello"
starts_with_word = input_str.startswith(start_word)
print("The string starts with '{}': {}".format(start_word, starts_with_word))
```

```
The string starts with 'Hello': True
```

[21]:
```
#  18. Check if a string ends with a specific word or phrase.
input_str = "Hello world!"

end_word = "world!"
ends_with_word = input_str.endswith(end_word)

print("The string ends with '{}': {}".format(end_word, ends_with_word))
```

```
The string ends with 'world!': True
```

[22]:
```
# 19. Convert a string to title case (e.g., "hello world" to "Hello World").

input_str = "hello world"
```

```
title_case_str = input_str.title()

print(title_case_str)
```

```
Hello World
```

[23]:
```
# 20. Find the longest word in a string.
def find_longest_word(input_string):

    words = input_string.split()

    longest_word = ""
    max_length = 0

    for word in words:
        if len(word) > max_length:
            longest_word = word
            max_length = len(word)

    return longest_word

input_str = "This is an example sentence to find the longest word"

longest_word = find_longest_word(input_str)

# Printing the result
print("The longest word in the string is:", longest_word)
```

```
The longest word in the string is: sentence
```

[1]:
```
# 21. Find the shortest word in a string.
def find_shortest_word(s):
    word = s.split()
    shartest_word=min(word,key=len)
    return shartest_word
string="this is a sample string with some short word like 'cat' and 'dog'."
print("shortest word:", find_shortest_word(string))
```

```
shortest word: a
```

[2]:
```
# 22. Reverse the order of words in a string.
def reverse_words(s):
    words = s.split()
    reversed_words = words[::-1]
    reversed_string = ' '.join(reversed_words)
```

```
    return reversed_string

string = "This is a sample string."
print("Reversed:", reverse_words(string))
```

```
Reversed: string. sample a is This
```

[3]:
```
# 23. Check if a string is alphanumeric.
def is_alphanumeric(s):
    return s.isalnum()

string = "Hello123"
print("Is alphanumeric:", is_alphanumeric(string))
```

```
Is alphanumeric: True
```

[34]:
```
# 24. Extract all digits from a string.
import re
def extract_digits(s):
    return re.findall(r'\d',s)
string="pooja123"
print("digits extracted from the string :", extract_digits(string))
```

```
digits extracted from the string : ['1', '2', '3']
```

[5]:
```
# 25. Extract all alphabets from a string.
import re

def extract_alphabets(s):
    return re.findall(r'[a-zA-Z]', s)

string = "pooja123"
print("Alphabets extracted from the string:", extract_alphabets(string))
```

```
Alphabets extracted from the string: ['p', 'o', 'o', 'j', 'a']
```

[6]:
```
# 26. Count the number of uppercase letters in a string.
def count_uppercase_letters(s):
    return sum(1 for char in s if char.isupper())

string = "Hello World"
print("Number of uppercase letters:", count_uppercase_letters(string))
```

```
Number of uppercase letters: 2
```

[7]:
```
 #27. Count the number of lowercase letters in a string.
def count_lowercase_letters(s):
```

```
    return sum(1 for char in s if char.islower())

string = "Hello World"
print("Number of lowercase letters:", count_lowercase_letters(string))
```

```
Number of lowercase letters: 8
```

[8]:
```
#  28. Swap the case of each character in a string.
def swap_case(s):
    return s.swapcase()

string = "Hello World"
print("Original string:", string)
print("String with swapped case:", swap_case(string))
```

```
Original string: Hello World
String with swapped case: hELLO wORLD
```

[9]:
```
# 29. Remove a specific word from a string.
def remove_word(sentence, word):
    return sentence.replace(word, '')

sentence = "This is a sample sentence with a specific word."
word_to_remove = "specific"
print("Original sentence:", sentence)
print("Sentence with word removed:", remove_word(sentence, word_to_remove))
```

```
Original sentence: This is a sample sentence with a specific word.
Sentence with word removed: This is a sample sentence with a  word.
```

[10]:
```
# 30. Check if a string is a valid email address.
import re

def is_valid_email(email):
    pattern = r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'
    return re.match(pattern, email) is not None

email = "example@example.com"
if is_valid_email(email):
    print("The email is valid.")
else:
    print("The email is not valid.")
```

```
The email is valid.
```

[11]:
```
# 31. Extract the username from an email address string.
def extract_username(email):
    return email.split('@')[0]
```

```
email = "pooja@email.com"
print("Username extracted from the email address:", extract_username(email))
```

```
Username extracted from the email address: pooja
```

[12]:
```
# 32. Extract the domain name from an email address string.
import re

def extract_domain(email):
    match = re.search(r"@(\w+(\.\w+)*)", email)
    if match:
        return match.group(1)
    else:
        return None

email = "example.user@example.com"
domain = extract_domain(email)
if domain:
    print("Domain:", domain)
else:
    print("Invalid email address")
```

```
Domain: example.com
```

[13]:
```
# 33. Replace multiple spaces in a string with a single space.
import re

def replace_multiple_spaces(string):
    return re.sub(r'\s+', ' ', string)

string = "Hello     world    how     are  you?"
cleaned_string = replace_multiple_spaces(string)
print("Cleaned string:", cleaned_string)
```

```
Cleaned string: Hello world how are you?
```

[14]:
```
# 34. Check if a string is a valid URL.
from urllib.parse import urlparse

def is_valid_url(url):
    try:
        result = urlparse(url)
        return all([result.scheme, result.netloc])
    except ValueError:
        return False
```

```
url = "https://www.pooja.com"
print("Is the URL valid?", is_valid_url(url))
```

```
Is the URL valid? True
```

[15]:
```
# 35. Extract the protocol (http or https) from a URL string.
from urllib.parse import urlparse

def extract_protocol(url):
    parsed_url = urlparse(url)
    if parsed_url.scheme in ['http', 'https']:
        return parsed_url.scheme
    else:
        return None

url = "https://www.pooja.com"
protocol = extract_protocol(url)
if protocol:
    print("Protocol:", protocol)
else:
    print("Invalid or unsupported protocol")
```

```
Protocol: https
```

[16]:
```
# 36. Find the frequency of each character in a string.
def count_characters(string):
    frequency = {}
    for char in string:
        if char in frequency:
            frequency[char] += 1
        else:
            frequency[char] = 1
    return frequency

string = "hello world"
character_frequency = count_characters(string)
print("Character frequency:", character_frequency)
```

```
Character frequency: {'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'w': 1, 'r': 1,
'd': 1}
```

[17]:
```
# 37. Remove all punctuation from a string.
import string

def remove_punctuation(text):
    translator = str.maketrans('', '', string.punctuation)
    return text.translate(translator)
```

```
text = "Hello, world! How are you?"
cleaned_text = remove_punctuation(text)
print("Cleaned text:", cleaned_text)
```

```
Cleaned text: Hello world How are you
```

[18]:
```
# 38. Check if a string contains only digits.
def contains_only_digits(text):
    return text.isdigit()

text = "12345"
if contains_only_digits(text):
    print("The string contains only digits.")
else:
    print("The string contains non-digit characters.")
```

```
The string contains only digits.
```

[19]:
```
# 39. Check if a string contains only alphabets.
def contains_only_alphabets(text):
    return text.isalpha()

text = "Hello"
if contains_only_alphabets(text):
    print("The string contains only alphabetic characters.")
else:
    print("The string contains non-alphabetic characters.")
```

```
The string contains only alphabetic characters.
```

[20]:
```
# 40. Convert a string to a list of characters.
string = "Hello, World!"
characters = [char for char in string]
print(characters)


characters = list(string)
print(characters)
```

```
['H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!']
['H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!']
```

[21]:
```
# 41. Check if two strings are anagrams.
def are_anagrams(str1, str2):

    str1_clean = ''.join(char.lower() for char in str1 if char.isalnum())
    str2_clean = ''.join(char.lower() for char in str2 if char.isalnum())
```

```
    sorted_str1 = sorted(str1_clean)
    sorted_str2 = sorted(str2_clean)


    return sorted_str1 == sorted_str2


string1 = "Listen"
string2 = "Silent"
if are_anagrams(string1, string2):
    print(f"{string1} and {string2} are anagrams.")
else:
    print(f"{string1} and {string2} are not anagrams.")
```

```
Listen and Silent are anagrams.
```

[22]:
```
#  42. Encode a string using a Caesar cipher.
def caesar_cipher_encrypt(text, shift):
    encrypted_text = ""
    for char in text:
        if char.isalpha():

            ascii_code = ord(char)

            is_upper = char.isupper()

            shifted_ascii = (ascii_code - 65 + shift if is_upper else␣
 ↪ascii_code - 97 + shift) % 26

            encrypted_char = chr(shifted_ascii + 65 if is_upper else␣
 ↪shifted_ascii + 97)
            encrypted_text += encrypted_char
        else:

            encrypted_text += char
    return encrypted_text


text = "Hello, World!"
shift = 3
encrypted_text = caesar_cipher_encrypt(text, shift)
print("Original:", text)
print("Encrypted:", encrypted_text)
```

```
Original: Hello, World!
```

```
Encrypted: Khoor, Zruog!
```

[23]:
```python
# 43. Decode a Caesar cipher encoded string.
def caesar_cipher_decrypt(text, shift):
    decrypted_text = ""
    for char in text:
        if char.isalpha():

            ascii_code = ord(char)

            is_upper = char.isupper()

            shifted_ascii = (ascii_code - 65 - shift if is_upper else␣
 ↪ascii_code - 97 - shift) % 26

            decrypted_char = chr(shifted_ascii + 65 if is_upper else␣
 ↪shifted_ascii + 97)
            decrypted_text += decrypted_char
        else:

            decrypted_text += char
    return decrypted_text


encrypted_text = "Khoor, Zruog!"
shift = 3
decrypted_text = caesar_cipher_decrypt(encrypted_text, shift)
print("Encrypted:", encrypted_text)
print("Decrypted:", decrypted_text)
```

```
Encrypted: Khoor, Zruog!
Decrypted: Hello, World!
```

[24]:
```python
# 44. Find the most frequent word in a string.
def most_frequent_word(text):

    words = text.split()


    word_freq = {}
    for word in words:

        word = word.strip('.,!?').lower()
        if word in word_freq:
            word_freq[word] += 1
        else:
            word_freq[word] = 1
```

```
    most_freq_word = max(word_freq, key=word_freq.get)
    return most_freq_word, word_freq[most_freq_word]


text = "This is a sample sentence. Another sample sentence with a word repeated␣
 ↪multiple times."
most_freq_word, frequency = most_frequent_word(text)
print(f"The most frequent word is '{most_freq_word}' with a frequency of␣
 ↪{frequency}.")
```

```
The most frequent word is 'a' with a frequency of 2.
```

[25]:
```
# 45. Find all unique words in a sting.
def unique_words(text):

    words = text.split()


    cleaned_words = [word.strip('.,!?').lower() for word in words]


    unique_word_set = set(cleaned_words)

    return unique_word_set


text = "This is a sample sentence. Another sample sentence with a word repeated␣
 ↪multiple times."
unique_words_set = unique_words(text)
print("Unique words:", unique_words_set)
```

```
Unique words: {'this', 'times', 'sentence', 'word', 'a', 'another', 'with',
'is', 'repeated', 'multiple', 'sample'}
```

[26]:
```
# 46. Count the number of syllables in a string.
import re

def count_syllables(word):

    word = re.sub(r'[^a-zA-Z]', '', word.lower())

    pattern = re.compile(r'[aeiouy]+')

    syllables = len(re.findall(pattern, word))
    return syllables
```

```
word = "hello"
print("Number of syllables in '", word, "':", count_syllables(word))
```

```
Number of syllables in ' hello ': 2
```

[27]:
```
# 47. Check if a string contains any special characters.
import re

def contains_special_characters(string):

    pattern = re.compile(r'[^a-zA-Z0-9\s]')

    match = re.search(pattern, string)
    if match:
        return True
    else:
        return False

text = "Hello, World!"
if contains_special_characters(text):
    print("The string contains special characters.")
else:
    print("The string does not contain any special characters.")
```

```
The string contains special characters.
```

[28]:
```
# 48. Remove the nth word from a string.
def remove_nth_word(text, n):
    words = text.split()

    if 0 <= n < len(words):
        del words[n]
        updated_text = ' '.join(words)
        return updated_text
    else:
        return "Invalid index. Word not removed."

text = "This is a sample sentence with some words."
n = 3
updated_text = remove_nth_word(text, n)
print("Original:", text)
print("Updated:", updated_text)
```

```
Original: This is a sample sentence with some words.
Updated: This is a sentence with some words.
```

[29]:
```python
# 49. Insert a word at the nth position in a string.
def insert_word_at_nth_position(text, word, n):
    words = text.split()

    if 0 <= n <= len(words):
        words.insert(n, word)
        updated_text = ' '.join(words)
        return updated_text
    else:
        return "Invalid index. Word not inserted."

text = "This is a sample sentence with some words."
word_to_insert = "new"
n = 3
updated_text = insert_word_at_nth_position(text, word_to_insert, n)
print("Original:", text)
print("Updated:", updated_text)
```

```
Original: This is a sample sentence with some words.
Updated: This is a new sample sentence with some words.
```

[30]:
```python
# 50. Convert a CSV string to a list of lists.
import csv
from io import StringIO

def csv_string_to_list(csv_string):
    csv_file = StringIO(csv_string)

    csv_reader = csv.reader(csv_file)

    csv_data = list(csv_reader)

    return csv_data

csv_string = "1,John,Doe\n2,Jane,Smith\n3,Bob,Johnson"
csv_data = csv_string_to_list(csv_string)
print("CSV Data:")
for row in csv_data:
    print(row)
```

```
CSV Data:
['1', 'John', 'Doe']
['2', 'Jane', 'Smith']
['3', 'Bob', 'Johnson']
```

[ ]:
```python
# Topic2
# List Based Practice Problem :
```

[31]:
```
# 1. Create a list with integers from 1 to 10.
numbers = list(range(1, 10))
print(numbers)
```

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

[32]:
```
# 2. Find the length of a list without using the `len()` function.
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
count = 0
for number in numbers:
    count += 1
print(count)
```

```
10
```

[33]:
```
# 3. Append an element to the end of a list.
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
numbers.append(11)
print(numbers)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

[35]:
```
# 4. Insert an element at a specific index in a list.
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
numbers.insert(5, 15)
print(numbers)
```

```
[1, 2, 3, 4, 5, 15, 6, 7, 8, 9, 10]
```

[36]:
```
# 5. Remove an element from a list by its value.
numbers = [1, 2, 3, 4, 5, 15, 6, 7, 8, 9, 10]
numbers.remove(5)
print(numbers)
```

```
[1, 2, 3, 4, 15, 6, 7, 8, 9, 10]
```

[37]:
```
# 6. Remove an element from a list by its index.
my_list=[1,2,3,4,5]
index_to_remove=2
removed_element=my_list.pop(index_to_remove)
print("remove element:", removed_element)
print("update list:",my_list)
```

```
remove element: 3
update list: [1, 2, 4, 5]
```

[38]:
```
# 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("Element exists in the list.")
else:
    print("Element does not exist in the list")
```

```
Element exists in the list.
```

[39]:
```
# 8. Find the index of the first occurrence of an element in a list.
my_list = [1,2,3,4,5,3]
element_to_find=4
index=my_list.index(element_to_find)
print("Index of the first occurrence of the element:",index)
```

```
Index of the first occurrence of the element: 3
```

[40]:
```
# 9. Count the occurrences of an element in a list.
my_list=[1,2,3,4,3,5,3,3]
element_to_count=3
count=my_list.count(element_to_count)
print("number of occurrences of the element:", count)
```

```
number of occurrences of the element: 4
```

[41]:
```
# 10. Reverse the order of elements in a list.
my_list=[1,2,3,4,5]
my_list.reverse()
print("reverse list :", my_list)
```

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

[42]:
```
#11. Sort a list in ascending order.
my_list=[4,6,7,2,3]
my_list.sort()
print("sorted list in ascending order:", my_list)
```

```
sorted list in ascending order: [2, 3, 4, 6, 7]
```

[43]:
```
#12. Sort a list in descending order.
my_list=[5,6,2,9,4]
my_list.sort(reverse = True)
print("sorted list in descending order:", my_list)
```

```
sorted list in descending order: [9, 6, 5, 4, 2]
```

[44]:
```
# 13. Create a list of even numbers from 1 to 20.
even_number = [x for x in range(1,21)if x%2==0]
print("list of even number from 1 to 21:", even_number)
```

```
list of even number from 1 to 21: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

[45]:
```
# 14. Create a list of odd numbers from 1 to 20.
odd_number = [x for x in range(1,20)if x%2!=0]
print("list of even number from 1 to 20:", odd_number)
```

```
list of even number from 1 to 20: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

[46]:
```
# 15. Find the sum of all elements in a list.
my_list=[1,2,3,4,5]
sum_of_element=sum(my_list)
print("sum of all element in the list:", sum_of_element)
```

```
sum of all element in the list: 15
```

[47]:
```
# 16. Find the maximum value in a list.
my_list=[10,5,20,15,30]
max_value=max(my_list)
print("Maximum value in the list :", max_value)
```

```
Maximum value in the list : 30
```

[48]:
```
# 17. Find the minimum value in a list.
my_list=[10,5,20,15,30]
min_value=min(my_list)
print("Maximum value in the list :", min_value)
```

```
Maximum value in the list : 5
```

[49]:
```
# 18. Create a list of squares of numbers from 1 to 10.
squares=[x**2 for x in range(1,11)]
print("list of squares of numbers from 1 to 10:",squares)
```

```
list of squares of numbers from 1 to 10: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

[50]:
```
# 19. Create a list of random numbers.
import random
random_number = [random.randint(1, 100) for _ in range(10)]
print("list of random  numbers :", random_number)
```

```
list of random  numbers : [48, 32, 60, 57, 56, 34, 23, 59, 16, 64]
```

[51]:
```
# 20. Remove duplicates from a list.
my_list=[1,2,3,4,2,6,3]
unique_list=list(set(my_list))
print("list after removing duplicated:", unique_list)
```

```
list after removing duplicated: [1, 2, 3, 4, 6]
```

[52]:
```
# 21. Find the common elements between two lists.
list1=[1,2,3,4,5]
list2=[4,5,6,7,8]
common_element=set(list1).intersection(set(list2))
print("common element between the two lists:", common_element)
```

```
common element between the two lists: {4, 5}
```

[53]:
```
# 22. Find the difference between two lists.
list1=[1,2,3,4,5,6]
list2=[4,5,6,7,8,9]
difference_list1_list2=list(set(list1).difference(set(list2)))
difference_list2_list1=list(set(list2).difference(set(list1)))
print("Difference from list1 to list2:", difference_list1_list2)
print("Difference from list2 to list1:", difference_list2_list1)
```

```
Difference from list1 to list2: [1, 2, 3]
Difference from list2 to list1: [8, 9, 7]
```

[54]:
```
 #23. Merge two lists.#
list1=[1,2,3,4]
list2=[5,6,7,8]
merged_list=list1+list2
print("merged list:", merged_list)
```

```
merged list: [1, 2, 3, 4, 5, 6, 7, 8]
```

[55]:
```
# 24. Multiply all elements in a list by 2.
my_list=[1,2,3,4,5]
multiplied_list=[x*2 for x in my_list]
print("list after multiplying all elements by 2:", multiplied_list)
```

```
list after multiplying all elements by 2: [2, 4, 6, 8, 10]
```

[56]:
```
# 25. Filter out all even numbers from a list.
my_list=[1,2,3,4,5,6,7,8,9,10]
filtered_list=[x for x in my_list if x%2!=0]
print("list after filtering out even numbers:", filtered_list)
```

```
list after filtering out even numbers: [1, 3, 5, 7, 9]
```

[57]:
```
# 26. Convert a list of strings to a list of integers.
string_list=["1","2","3","4","5"]
integer_list=[int (x) for x in string_list]
print("list of integers:",integer_list)
```

```
list of integers: [1, 2, 3, 4, 5]
```

[58]:
```
# 27. Convert a list of integers to a list of strings.
string_list=["1","2","3","4","5"]
integer_list=[int (x) for x in string_list]
print("list of integers:",integer_list)
```

```
list of integers: [1, 2, 3, 4, 5]
```

[59]:
```
# 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)
```

```
flattened list: [1, 2, 3, 4, 5, 6, 7, 8]
```

[60]:
```
# 29. Create a list of the first 10 Fibonacci numbers.
def fibonacci(n):
    fib_sequence = [0, 1]
    while len(fib_sequence) < n:
        fib_sequence.append(fib_sequence[-1] + fib_sequence[-2])
    return fib_sequence

first_10_fibonacci = fibonacci(10)
print(first_10_fibonacci)
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

[61]:
```
# 30. Check if a list is sorted.
my_list=[1,2,3,4,5]
is_sorted = my_list == sorted(my_list)
if is_sorted:
    print("this list is sorted.")
else:
    print("this list is not sorted")
```

```
this list is sorted.
```

[62]:
```
# 31. Rotate a list to the left by `n` positions.
def rotate_left(lst,n):
    return lst[n:]+lst[:n]
my_list=[1,2,3,4,5]
n=2
rotate_left=rotate_left(my_list,n)
print("list after rotating left by ",n , "positions:", rotate_left)
```

```
list after rotating left by  2 positions: [3, 4, 5, 1, 2]
```

[63]:
```
# 32. Rotate a list to the right by `n` positions.
def rotate_right(lst,n):
```

```
    return lst[-n:]+lst[:-n]
my_list=[1,2,3,4,5]
n=2
rotate_list=rotate_right(my_list,n)
print("list after rotating right by ", n, "position:", rotate_list)
```

```
list after rotating right by  2 position: [4, 5, 1, 2, 3]
```

[64]:
```
# 33. Create a list of prime numbers up to 50.
def is_prime(num):
    if num<2:
        return False
    for i in range(2,int(num**0.5)+1):
        if num% i == 0:
            return True
prime_numbers=[num for num in range(2,51)if is_prime (num)]
print("list of prime numbers up to 50:", prime_numbers)
```

```
list of prime numbers up to 50: [4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22,
24, 25, 26, 27, 28, 30, 32, 33, 34, 35, 36, 38, 39, 40, 42, 44, 45, 46, 48, 49,
50]
```

[65]:
```
# 34. Split a list into chunks of size `n`.
def split_into_chunks(lst,n):
    return[lst[i:i+n] for i in range(0,len(lst),n) ]
my_list=[1,2,3,4,5,6,7,8,9,10]
chunk_size=3
chunk=split_into_chunks(my_list,chunk_size)
print("list split into chunks of size", chunk_size, ":", chunk)
```

```
list split into chunks of size 3 : [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10]]
```

[66]:
```
# 35. Find the second largest number in a list.
def second_largest(lst):
    sorted_list=sorted(lst)
    return sorted_list[-2]
my_list=[10,5,20,15,30]
second_largest_num=second_largest(my_list)
print("second largest number in the list:", second_largest_num)
```

```
second largest number in the list: 20
```

[67]:
```
# 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("list with every element replaced by its square:", squared_list)
```

```
list with every element replaced by its square: [1, 4, 9, 16, 25]
```

[68]:
```
# 37. Convert a list to a dictionary where list elements become keys and their indices become values.
my_list=['a','b','c','d','e']
result_dict={element: index for index, element in enumerate(my_list)}
print("dictionary :",result_dict)
```

```
dictionary : {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4}
```

[69]:
```
# 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)
```

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

[70]:
```
# 39. Create a list of the first 10 factorial numbers.
def factorial(n):
    if n==0:
        return 1
    else:
        return n*factorial (n-1)
factorial_number=[factorial(i) for i in range(10)]
print("list of the first 10 factorial number:", factorial_number)
```

```
list of the first 10 factorial number: [1, 1, 2, 6, 24, 120, 720, 5040, 40320,
362880]
```

[71]:
```
# 40. Check if two lists have at least one element in common.
list1=[1,2,3,4,5]
list2=[6,7,8,9,10]
has_common_element=any(element in list2 for element in list1)
if has_common_element:
    print("this two list have at last one common element.")
else:
    print("this two list do not have at last one common element.")
```

```
this two list do not have at last one common element.
```

[72]:
```
# 41. Remove all elements from a list.
my_list=[1,2,3,4,5]
my_list[:]=[]
print("list after removing all element :", my_list)
```

```
list after removing all element : []
```

[73]:
```
# 42. Replace negative numbers in a list with 0.
my_list=[1,-2,3,-4,5,-6,7,8,-9]
modified_list=[x if x>=0 else 0 for x in my_list]
print("list after replacing negative number with 0:", modified_list)
```

```
list after replacing negative number with 0: [1, 0, 3, 0, 5, 0, 7, 8, 0]
```

[74]:
```
# 43. Convert a string into a list of words.
my_string="Hello world, this is a string"
word_list=my_string.split()
print("list of words:",word_list)
```

```
list of words: ['Hello', 'world,', 'this', 'is', 'a', 'string']
```

[75]:
```
# 44. Convert a list of words into a string.
word_list=['hello','world','this','is','a','string']
my_string=' '.join(word_list)
print("string :", my_string)
```

```
string : hello world this is a string
```

[76]:
```
# 45. Create a list of the first `n` powers of 2.
n=5
powers_of_2=[2** i for i in range(n)]
print("list of the first", n, "powers of 2:", powers_of_2)
```

```
list of the first 5 powers of 2: [1, 2, 4, 8, 16]
```

[77]:
```
# 46. Find the longest string in a list of strings.
string_list=["apple","banana","orange","strawbery"]
longest_string=max(string_list,key=len)
print("longest string:", longest_string)
```

```
longest string: strawbery
```

[78]:
```
# 47. Find the shortest string in a list of strings.
string_list=["apple","banana","orange","strawbery"]
shortest_string=min(string_list,key=len)
print("longest string:", shortest_string)
```

```
longest string: apple
```

[79]:
```
# 48. Create a list of the first `n` triangular numbers.
n=5
triangular_number=[(i*(i+1))//2 for i in range (1, n+1)]
print("list of first", n, "triangular number:", triangular_number)
```

```
list of first 5 triangular number: [1, 3, 6, 10, 15]
```

[80]:
```
# 49. Check if a list contains another list as a subsequence.
def is_subsequence(subseq, seq):
    i,j=0,0
    while i<len(subseq)and j < len(seq):
        if subseq[i] == seq[j]:
            i +=1
        j +=1
    return i == len(seq)
list1 = [1,2,3,4,5,6]
list2 = [2,4,6]
result = is_subsequence(list2,list1)
if result:
    print("list2 is a subsequence of list1.")
else:
    print("list2 is not a subsequence of list1.")
```

```
list2 is not a subsequence of list1.
```

[81]:
```
# 50. Swap two elements in a list by their indices.
def swap_element(lst,index1,index2):
    lst[index1],lst[index2]=lst[index2],lst[index1]
my_list=[1,2,3,4,5]
index1=1
index2=3
swap_element(my_list,index1,index2)
print("list after swapping element:", my_list)
```

```
list after swapping element: [1, 4, 3, 2, 5]
```

[ ]:
```
#topic 3
#Tuple Based Practice Problem
```

[83]:
```
# 1. Create a tuple with integers from 1 to 5.
my_tuple=(1,2,3,4,5)
print(my_tuple)
```

```
(1, 2, 3, 4, 5)
```

[84]:
```
# 2. Access the third element of a tuple.
my_tuple=(1,2,3,4,5)
third_element=my_tuple[2]
print(third_element)
```

```
3
```

[85]:
```
# 3. Find the length of a tuple without using the `len()` function.
my_tuple=(1,2,3,4,5)
length=0
```

```
for _ in my_tuple:
    length +=1
print(length)
```

```
5
```

[86]:
```
# 4. Count the occurrences of an element in a tuple.
my_tuple=(1,2,3,4,5,3,3)
element=3
occurrences= my_tuple.count(element)
print(" this element ", element , "occurs", occurrences,"this in teh tuple.")
```

```
 this element  3 occurs 3 this in teh tuple.
```

[87]:
```
# 5. Find the index of the first occurrence of an element in a tuple.
my_tuple=(1,2,3,4,5,4,4)
element=4
first_index=my_tuple.index(element)
print("this index of the first occurrenc3 of", element , "is:", first_index)
```

```
this index of the first occurrenc3 of 4 is: 3
```

[88]:
```
# 6. Check if an element exists in a tuple.
my_tuple=(1,2,3,4,5)
element_to_check=4
if element_to_check in my_tuple:
    print("element exists in the tuple")
else:
    print("element does not exists in the tuple")
```

```
element exists in the tuple
```

[89]:
```
# 7. Convert a tuple to a list.
my_tuple=(1,2,3,4,5)
my_list=list(my_tuple)
print(my_list)
```

```
[1, 2, 3, 4, 5]
```

[90]:
```
# 8. Convert a list to a tuple.
my_list=(1,2,3,4,5)
my_tuple=tuple(my_list)
print(my_tuple)
```

```
(1, 2, 3, 4, 5)
```

[91]:
```
# 9. Unpack the elements of a tuple into variables.
my_tuple=(1,2,3)
```

```
a,b,c=my_tuple
print("a=",a)
print("b=",b)
print("c=",c)
```

```
a= 1
b= 2
c= 3
```

[92]:
```
# 10. Create a tuple of even numbers from 1 to 10.
even_number=tuple(x for x in range (2,11,2))
print(even_number)
```

```
(2, 4, 6, 8, 10)
```

[93]:
```
# 11. Create a tuple of odd numbers from 1 to 10.
odd_number=tuple(x for x in range(1,11)if x % 2 !=0)
print(odd_number)
```

```
(1, 3, 5, 7, 9)
```

[94]:
```
# 12. Concatenate two tuples.
tuple1=(1,2,3,4)
tuple2=(5,6,7,8)
concatenated_tuple=tuple1+tuple2
print(concatenated_tuple)
```

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

[95]:
```
# 13. Repeat a tuple three times.
original_tuple=(1,2,3)
repeated_tuple=original_tuple*3
print(repeated_tuple)
```

```
(1, 2, 3, 1, 2, 3, 1, 2, 3)
```

[96]:
```
# 14. Check if a tuple is empty.
my_tuple=()
if not my_tuple:
    print("tuple is empty")
else:
    print("tuple is not empty")
```

```
tuple is empty
```

[97]:
```
# 15. Create a nested tuple.
nested_tuple=((1,2),(3,4),(5,6))
print(nested_tuple)
```

```
((1, 2), (3, 4), (5, 6))
```

[98]:
```
#16. Access the first element of a nested tuple.
nested_tuple=((1,2),(3,4),(5,6))
first_element=nested_tuple[0]
print(first_element)
```

```
(1, 2)
```

[99]:
```
# 17. Create a tuple with a single element.
single_element_tuple=(6)
print(single_element_tuple)
```

```
6
```

[100]:
```
# 18. Compare two tuples.
tuple1=(1,2,3)
tuple2=(1,2,3)
if tuple1==tuple2:
    print("tuple are equal")
else:
    print("tuple are not equal")
```

```
tuple are equal
```

[138]:
```
# 19. Delete a tuple.
my_tuple = (1, 2, 3)
del my_tuple

try:
    print(my_tuple)
except NameError as e:
    print(e)
```

```
name 'my_tuple' is not defined
```

[107]:
```
# 20. Slice a tuple.
my_tuple=(1,2,3,4,5)
sliced_typle=my_tuple[1:4]
print(sliced_typle)
```

```
(2, 3, 4)
```

[103]:
```
# 21. Find the maximum value in a tuple.
my_tuple=(5,6,2,8,3,0)
max_value=max(my_tuple)
print("maximum value in the tuple:", max_value)
```

```
maximum value in the tuple: 8
```

[104]:
```
# 22. Find the minimum value in a tuple.
my_tuple=(5,3,6,7,9,0)
min_value=min(my_tuple)
print(" minimum value in the tuple:", min_value)
```

```
 minimum value in the tuple: 0
```

[109]:
```
# 23. Convert a string to a tuple of characters.
my_string="hello"
char_tuple=tuple(my_string)
print(char_tuple)
```

```
('h', 'e', 'l', 'l', 'o')
```

[110]:
```
# 24. Convert a tuple of characters to a string.
char_tuple=('h','e','l','l','o')
my_string=''.join(char_tuple)
print(my_string)
```

```
hello
```

[111]:
```
# 25. Create a tuple from multiple data types.
mixed_tuple=(1,"apple","true",3,14)
print(mixed_tuple)
```

```
(1, 'apple', 'true', 3, 14)
```

[112]:
```
# 26. Check if two tuples are identical.
tuple1=(1,2,3)
tuple2=(1,2,3)
if tuple1==tuple2:
    print("tuples are identical")
else:
    print("tuples are not identical")
```

```
tuples are identical
```

[113]:
```
# 27. Sort the elements of a tuple.
my_tuple=(3,2,5,6,7,8,5)
sorted_list=sorted(my_tuple)
sorted_tuple=tuple(sorted_list)
print(sorted_tuple)
```

```
(2, 3, 5, 5, 6, 7, 8)
```

[114]:
```
# 28. Convert a tuple of integers to a tuple of strings.
tuple_of_integers=(1,2,3,4,5)
tuple_of_string=tuple(str(x)for x in tuple_of_integers)
```

```
print(tuple_of_string)
```

```
('1', '2', '3', '4', '5')
```

[115]:
```
# 29. Convert a tuple of strings to a tuple of integers.
tuple_of_integers=('1','2','3','4','5')
tuple_of_integers=tuple(int(x)for x in tuple_of_integers)
print(tuple_of_integers)
```

```
(1, 2, 3, 4, 5)
```

[116]:
```
#30. Merge two tuples
tuple_of_integers=('1','2','3','4','5')
tuple_of_integers=tuple(int(x)for x in tuple_of_integers)
print(tuple_of_integers)
```

```
(1, 2, 3, 4, 5)
```

[117]:
```
# 31. Flatten a nested tuple.
def flatten_tuple(nested_tuple):
    flatten_list = []
    for item in nested_tuple:
        if isinstance(item, tuple):
            flatten_list.extend(flatten_tuple(item))
        else:
            flatten_list.append(item)
    return tuple(flatten_list)

nested_tuple = ((1, 2), (3, (4, 5)), 6)
flattened_tuple = flatten_tuple(nested_tuple)
print(flattened_tuple)
```

```
(1, 2, 3, 4, 5, 6)
```

[118]:
```
# 32. Create a tuple of the first 5 prime numbers.
prime_number=(2,3,5,7,11)
print(prime_number)
```

```
(2, 3, 5, 7, 11)
```

[119]:
```
# 33. Check if a tuple is a palindrome.
def is_palindrome_tuple(input_tuple):
    input_list=list(input_tuple)
    reversed_list=input_list[::-1]
    return input_list==reversed_list
tuple1=(1,2,3,4,1,2)
tuple2=(1,2,3,4,5)
print(is_palindrome_tuple(tuple1))
```

```
print(is_palindrome_tuple(tuple2))
```

```
False
False
```

[120]:
```
# 34. Create a tuple of squares of numbers from 1 to 5.
squares_tuple=tuple(x**2 for x in range (1,6))
print(squares_tuple)
```

```
(1, 4, 9, 16, 25)
```

[121]:
```
#35. Filter out all even numbers from a tuple.
original_tuple=(1,2,3,4,5,6,7,8,9,10)
filtred_tuple=tuple(x for x in original_tuple if x % 2!=0)
print(filtred_tuple)
```

```
(1, 3, 5, 7, 9)
```

[122]:
```
# 36. Multiply all elements in a tuple by 2.
original_tuple=(1,2,3,4,5,6,7,8)
multiplied_tuple=tuple(x* 2 for x in original_tuple)
print(multiplied_tuple)
```

```
(2, 4, 6, 8, 10, 12, 14, 16)
```

[123]:
```
# 37. Create a tuple of random numbers.
import random
random_tuple=tuple(random.randint(1,100) for _ in range (5))
print(random_tuple)
```

```
(44, 86, 75, 34, 51)
```

[124]:
```
# 38. Check if a tuple is sorted.
def is_sorted_tuple(input_tuple):
    return input_tuple==tuple(sorted(input_tuple))
tuple1=(1,2,3,4,5)
tuple2=(5,4,3,2,1)
print(is_sorted_tuple(tuple1))
print(is_sorted_tuple(tuple2))
```

```
True
False
```

[125]:
```
# 39. Rotate a tuple to the left by `n` positions.
def rotate_left_tuple(input_tuple,n):
    n=n% len(input_tuple)
    return input_tuple[n:]+input_tuple[:n]
```

```
original_tuple=(1,2,3,4,5)
rotate_left=rotate_left_tuple(original_tuple,2)
print(original_tuple)
```

```
(1, 2, 3, 4, 5)
```

[126]:
```
# 40. Rotate a tuple to the right by `n` positions.
def rotate_right_tuple(input_tuple,n):
    a=n% len(input_tuple)
    return input_tuple[-n:]+input_tuple[:-n]
original_tuple=(1,2,3,4,5)
rotate_tuple=rotate_right_tuple(original_tuple,2)
print(rotate_tuple)
```

```
(4, 5, 1, 2, 3)
```

[127]:
```
# 41. Create a tuple of the first 5 Fibonacci numbers.
def generate_fibonacci(n):
    fibonacci = [0, 1]
    for i in range(2, n):
        fibonacci.append(fibonacci[i-1] + fibonacci[i-2])
    return tuple(fibonacci)

fibonacci_tuple = generate_fibonacci(5)
print(fibonacci_tuple)
```

```
(0, 1, 1, 2, 3)
```

[129]:
```
# 42. Create a tuple from user input.
input_str=input("enter elements separated by space/comma:")
user_tuple=tuple(input_str.split())
print(user_tuple)
```

```
enter elements separated by space/comma: My Name is Pooja
('My', 'Name', 'is', 'Pooja')
```

[130]:
```
#  43. Swap two elements in a tuple.
def swapped_elements(input_tuple, index1, index2):
    list_version = list(input_tuple)
    list_version[index1], list_version[index2] = list_version[index2],␣
 ↪list_version[index1]  # Correct swap
    return tuple(list_version)
my_tuple = (1, 2, 3, 4, 5)
swapped_tuple = swapped_elements(my_tuple, 1, 3)
print(swapped_tuple)
```

```
(1, 4, 3, 2, 5)
```

[131]:
```
# 44. Reverse the elements of a tuple.
def reverse_tuple(input_tuple):
    return input_tuple[::-1]

my_tuple = (1, 2, 3, 4, 5)
reversed_tuple = reverse_tuple(my_tuple)
print(reversed_tuple)
```

```
(5, 4, 3, 2, 1)
```

[132]:
```
# 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
powers_tuple = powers_of_2(n)
print(powers_tuple)
```

```
(1, 2, 4, 8, 16)
```

[133]:
```
# 46. Find the longest string in a tuple of strings.
def longest_string(strings):
    if not strings:
        return None
    return max(strings, key=len)


string_tuple = ("apple", "banana", "cherry", "date")
longest = longest_string(string_tuple)
print(longest)
```

```
banana
```

[134]:
```
# 47. Find the shortest string in a tuple of strings.
strings = ("apple", "banana", "cherry", "date", "elderberry", "fig", "grape")


shortest_string = min(strings, key=len)


print(f"The shortest string is: {shortest_string}")
```

```
The shortest string is: fig
```

[135]:
```
# 48. Create a tuple of the first `n` triangular numbers.
def triangular_number(n):
   return tuple(i * (i + 1) // 2 for i in range(1, n + 1))
n = 5
```

```
triangular_tuple = triangular_number(n)
print(triangular_tuple)
```

```
(1, 3, 6, 10, 15)
```

[136]:
```
# 49. Check if a tuple contains another tuple as a subsequence.
def contains_subsequence(main_tuple, sub_tuple):
    n, m = len(main_tuple), len(sub_tuple)
    if m > n:
        return False

    for i in range(n - m + 1):
        if main_tuple[i:i + m] == sub_tuple:
            return True

    return False

main_tuple = (1, 2, 3, 4, 5, 6)
sub_tuple = (3, 4, 5)

result = contains_subsequence(main_tuple, sub_tuple)
print(result)
```

```
True
```

[137]:
```
#  50. Create a tuple of alternating 1s and 0s of length `n`.
def alternating_tuple(n):
    return tuple(1 if i % 2==0 else 0 for i in range(n))
n=6
alternating=alternating_tuple(n)
print(alternating)
```

```
(1, 0, 1, 0, 1, 0)
```

[ ]:
```
#Topic 4
#Set Based Practice Problem
```

[1]:
```
# 1. Create a set with integers from 1 to 5.
my_set={1,2,3,4,5}
print(my_set)
```

```
{1, 2, 3, 4, 5}
```

[2]:
```
# 2. Add an element to a set.
my_set={1,2,3,4,5}
my_set.add(6)
print(my_set)
```

```
{1, 2, 3, 4, 5, 6}
```

[3]:
```python
# 3. Remove an element from a set.
my_set={1,2,3,4,5}
my_set.remove(3)
print(my_set)
```

```
{1, 2, 4, 5}
```

[4]:
```python
# 4. Check if an element exists in a set.
my_set={1,2,3,4,5}
element=3
if element in my_set:
    print("element", element,"exists in the set.")
else:
    print("element", element, "does not exists in the set")
```

```
element 3 exists in the set.
```

[5]:
```python
# 5. Find the length of a set without using the `len()` function.
my_set={1,2,3,4,}
length=0
for _ in my_set:
    length=1
    print("length of the set", length)
```

```
length of the set 1
length of the set 1
length of the set 1
length of the set 1
```

[6]:
```python
# 6. Clear all elements from a set.
my_set={1,2,3,4,5}
my_set.clear()
print("set after clearing all element : ", my_set)
```

```
set after clearing all element :  set()
```

[7]:
```python
# 7. Create a set of even numbers from 1 to 10.
even_number={x for x in range(1,11)if x % 2==0}
print("set of even number from 1 to 10: ", even_number)
```

```
set of even number from 1 to 10:  {2, 4, 6, 8, 10}
```

[8]:
```python
# 8. Create a set of odd numbers from 1 to 10.
odd_number={x for x in range (1,11)if x % 2 ==0}
print(" set of odd number from 1 to 10:", odd_number)
```

```
set of odd number from 1 to 10: {2, 4, 6, 8, 10}
```

[9]:
```
# 9. Find the union of two sets.
set1={1,2,3}
set2={3,4,5}
union_set_method=set1.union(set2)
print("union using union()method :", union_set_method)
union_set_operator=set1|set2
print("union using| operator:", union_set_operator)
```

```
union using union()method : {1, 2, 3, 4, 5}
union using| operator: {1, 2, 3, 4, 5}
```

[10]:
```
# 10. Find the intersection of two sets.
set1={1,2,3,4,5}
set2={4,5,6,7,8}
intersection_set_method=set1.intersection(set2)
print("intersection using intersection()method:", intersection_set_method)
intersection_set_operator=set1&set2
print("intersection using & operator:", intersection_set_method)
```

```
intersection using intersection()method: {4, 5}
intersection using & operator: {4, 5}
```

[11]:
```
# 11. Find the difference between two sets.
set1={1,2,3}
set2={3,4,5}
difference_set_method=set1.difference(set2)
print("difference using difference() method:", difference_set_method)
difference_set_operator=set1-set2
print("difference using operator:", difference_set_operator)
```

```
difference using difference() method: {1, 2}
difference using operator: {1, 2}
```

[12]:
```
# 12. Check if a set is a subset of another set.
set1={1,2,3}
set2={1,2,3,4,5}
is_subset_method=set1.issubset(set2)
print("is set1 a subset of set2 using issubset() method", is_subset_method)
is_subset_operator=set1<=set2
print("is set1 a subset of set2 using <= operator:", is_subset_operator)
```

```
is set1 a subset of set2 using issubset() method True
is set1 a subset of set2 using <= operator: True
```

[13]:
```
# 13. Check if a set is a superset of another set.
asset1 = {1, 2, 3, 4, 5}
```

```
set2 = {3, 4}

if asset1.issuperset(set2):
    print("set1 is a superset of set2")
else:
    print("set1 is not a superset of set2")
```

```
set1 is a superset of set2
```

[14]:
```
# 14. Create a set from a list.
my_list = [1, 2, 3, 4, 5]
my_set = set(my_list)
print(my_set)
```

```
{1, 2, 3, 4, 5}
```

[15]:
```
# 15. Convert a set to a list.
my_set = {1, 2, 3, 4, 5}
my_list = list(my_set)
print(my_list)
```

```
[1, 2, 3, 4, 5]
```

[16]:
```
# 16. Remove a random element from a set.
my_set = {1, 2, 3, 4, 5}
removed_element = my_set.pop()
print("Removed element:", removed_element)
print("Updated set:", my_set)
```

```
Removed element: 1
Updated set: {2, 3, 4, 5}
```

[17]:
```
# 17. Pop an element from a set.
my_set = {1, 2, 3, 4, 5}
element_to_pop = 3

if element_to_pop in my_set:
    my_set.remove(element_to_pop)
    print("Element", element_to_pop, "was popped from the set.")
else:
    print("Element", element_to_pop, "is not in the set.")

print("Updated set:", my_set)
```

```
Element 3 was popped from the set.
Updated set: {1, 2, 4, 5}
```

[18]:
```python
# 18. Check if two sets have no elements in common.
set1 = {1, 2, 3}
set2 = {4, 5, 6}

if set1.isdisjoint(set2):
    print("The sets have no elements in common.")
else:
    print("The sets have at least one element in common.")
```

```
The sets have no elements in common.
```

[19]:
```python
# 19. Find the symmetric difference between two sets.
set1 = {1, 2, 3}
set2 = {3, 4, 5}

symmetric_diff = set1.symmetric_difference(set2)
print("Symmetric difference:", symmetric_diff)
```

```
Symmetric difference: {1, 2, 4, 5}
```

[20]:
```python
# 20. Update a set with elements from another set.
set1 = {1, 2, 3}
set2 = {3, 4, 5}

set1.update(set2)

print("Updated set1:", set1)
```

```
Updated set1: {1, 2, 3, 4, 5}
```

[21]:
```python
# 21. Create a set of the first 5 prime numbers.
def generate_primes(n):
    primes = set()
    num = 2
    while len(primes) < n:
        is_prime = True
        for i in range(2, int(num ** 0.5) + 1):
            if num % i == 0:
                is_prime = False
                break
        if is_prime:
            primes.add(num)
        num += 1
    return primes

prime_numbers = generate_primes(5)
print(prime_numbers)
```

```
{2, 3, 5, 7, 11}
```

[22]:
```python
# 22. Check if two sets are identical.
set1 = {1, 2, 3}
set2 = {3, 2, 1}

if set1 == set2:
    print("The sets are identical.")
else:
    print("The sets are not identical.")
```

```
The sets are identical.
```

[23]:
```python
# 23. Create a frozen set.
my_set = {1, 2, 3, 4, 5}
frozen_set = frozenset(my_set)
print(frozen_set)
```

```
frozenset({1, 2, 3, 4, 5})
```

[24]:
```python
# 24. Check if a set is disjoint with another set.
set1 = {1, 2, 3}
set2 = {4, 5, 6}

if set1.isdisjoint(set2):
    print("The sets are disjoint.")
else:
    print("The sets are not disjoint.")
```

```
The sets are disjoint.
```

[25]:
```python
# 25. Create a set of squares of numbers from 1 to 5.
squares = {x ** 2 for x in range(1, 6)}
print(squares)
```

```
{1, 4, 9, 16, 25}
```

[26]:
```python
# 26. Filter out all even numbers from a set.
my_set = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
odd_numbers = {x for x in my_set if x % 2 != 0}
print(odd_numbers)
```

```
{1, 3, 5, 7, 9}
```

[27]:
```python
# 27. Multiply all elements in a set by 2.
original_set={1,2,3,4,5}
doubled_element_set={x*2 for x in original_set}
print("set with element multiplied by 2:", doubled_element_set)
```

```
set with element multiplied by 2: {2, 4, 6, 8, 10}
```

[28]:
```python
# 28. Create a set of random numbers.
import random

random_set = set()
for _ in range(10):  # Change 10 to the desired size of the set
    random_set.add(random.randint(1, 100))  # Generate a random integer between␣
 ↪1 and 100 (inclusive)

print(random_set)
```

```
{97, 36, 68, 39, 43, 29, 22, 55, 56, 61}
```

[29]:
```python
# 29. Check if a set is empty.
my_set = set()

if not my_set:
    print("The set is empty.")
else:
    print("The set is not empty.")
```

```
The set is empty.
```

[30]:
```python
# 30. Create a nested set (hint: use frozenset).
nested_set = {frozenset({1, 2}), frozenset({3, 4, 5})}
print(nested_set)
```

```
{frozenset({3, 4, 5}), frozenset({1, 2})}
```

[31]:
```python
# 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 element:", my_set)
```

```
set after removing element: {1, 2, 4, 5}
```

[32]:
```python
# 32. Compare two sets.
set1={1,2,3,4}
set2={1,2,3,3}
print("equality :", set1==set2)
print("subset:", set1.issubset(set2))
print("superset:", set1.issuperset(set2))
print("proper subset:", set1<set2)
print("proper superset:", set1>set2)
print("disjoint:", set1.isdisjoint(set2))
```

```
equality : False
```

```
subset: False
superset: True
proper subset: False
proper superset: True
disjoint: False
```

[33]:
```python
# 33. Create a set from a string.
my_set="hello"
my_set=set(my_set)
print("set from string:", my_set)
```

```
set from string: {'l', 'e', 'o', 'h'}
```

[34]:
```python
# 34. Convert a set of strings to a set of integers.
set_of_string={"1","2","3","4","5"}
set_of_integers={int(x) for x in set_of_string}
print("set of integers converted from set of string:", set_of_integers)
```

```
set of integers converted from set of string: {1, 2, 3, 4, 5}
```

[35]:
```python
# 35. Convert a set of integers to a set of strings.
set_of_integers={1,2,3,4,5}
set_of_strings={str(x) for x in set_of_integers}
print("set of string converted from set of integers", set_of_strings)
```

```
set of string converted from set of integers {'4', '2', '5', '1', '3'}
```

[36]:
```python
# 36. Create a set from a tuple.
my_tuple=(1,2,3,4,5)
my_set=set(my_tuple)
print("set from tuple:", my_set)
```

```
set from tuple: {1, 2, 3, 4, 5}
```

[37]:
```python
# 37. Convert a set to a tuple.
my_set={1,2,3,4,5}
my_tuple=tuple(my_set)
print("tuple converted from set:", my_tuple)
```

```
tuple converted from set: (1, 2, 3, 4, 5)
```

[38]:
```python
# 38. Find the maximum value in a set.
my_set={1,3,4,7,5}
max_value=max(my_set)
print("maximum value in the set:",max_value)
```

```
maximum value in the set: 7
```

[39]:
```
# 39. Find the minimum value in a set.
my_set={1,2,3,4,6}
min_value=min(my_set)
print("minimum value in the set:", min_value)
```

```
minimum value in the set: 1
```

[42]:
```
# 40. Create a set from user input.
user_input=input("enter element separated by spaces:")
input_itst=user_input.split()
user_set=set(input_itst)
print("set created from user input:", user_set)
```

```
enter element separated by spaces: 3 4 5 7 2 7
set created from user input: {'4', '2', '7', '5', '3'}
```

[43]:
```
# 41. Check if the intersection of two sets is empty.
set1={1,2,3,4}
set2={5,6,7,8}
intersection_empty=set1.isdisjoint(set2)
print("is the intersection of the two sets empty?", intersection_empty)
```

```
is the intersection of the two sets empty? True
```

[44]:
```
# 42. Create a set of the first 5 Fibonacci numbers.
fib_set=set()
a,b=0,1
for _ in range(5):
    fib_set.add(a)
    a,b=b,a+b
print("set of the first 5 fibonacci number:", fib_set)
```

```
set of the first 5 fibonacci number: {0, 1, 2, 3}
```

[45]:
```
# 43. Remove duplicates from a list using sets.
my_list={1,2,3,4,5,4,3,2}
unique_list=list(set(my_list))
print("list with duplicates removed:", unique_list)
```

```
list with duplicates removed: [1, 2, 3, 4, 5]
```

[46]:
```
# 44. Check if two sets have the same elements, regardless of their count.
set1={1,2,3}
set2={2,3,4}
same_element =len(set1)==len(set2)
print("do the set have the same element, regardless, of count?", same_element)
```

```
do the set have the same element, regardless, of count? True
```

[47]:
```
# 45. Create a set of the first `n` powers of 2.
n=5
power_of_2_set={2**i for i in range(n)}
print("set of the first ", n, "powers of 2:", power_of_2_set)
```

```
set of the first  5 powers of 2: {1, 2, 4, 8, 16}
```

[48]:
```
# 46. Find the common elements between a set and a list.
my_set={1,2,3,4,5}
my_list={4,5,6,7,8}
common_element_method=my_set.intersection(my_list)
print("common element using intersection() method:", common_element_method)
common_element_operater=my_set&set(my_list)
print("common element using & operator:", common_element_operater)
```

```
common element using intersection() method: {4, 5}
common element using & operator: {4, 5}
```

[49]:
```
# 47. Create a set of the first `n` triangular numbers.
n=5
triangular_number_set={i*(i+1)//2 for i in range(1,n+1)}
print("set of the first",n,"triangular number:", triangular_number_set)
```

```
set of the first 5 triangular number: {1, 3, 6, 10, 15}
```

[50]:
```
# 48. Check if a set contains another set as a subset.
set1={1,2,3,4,5}
set2={3,2,1}
is_subset_method=set2.issubset(set1)
print("is set2 a subset of set1 using issubset() method?", is_subset_method)
is_subset_operator=set2<=set1
print("is set2 a subset of set1 using<=operator?", is_subset_operator)
```

```
is set2 a subset of set1 using issubset() method? True
is set2 a subset of set1 using<=operator? True
```

[51]:
```
# 49. Create a set of alternating 1s and 0s of length `n`.
n=5
alternating_set={1 if i%2==0 else 0 for i in range(n)}
print("set of alternating 1s and 0s of length", n, ":",alternating_set)
```

```
set of alternating 1s and 0s of length 5 : {0, 1}
```

[52]:
```
# 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 using union() method:", merged_set)
```

```
merged set using union() method: {1, 2, 3, 4, 5, 6, 7}
```