

TOPIC: String Based Assignment

1. Write a program to reverse a string.

Python code:-

```
def reverse_string(input_string):  
    reversed_string = input_string[::-1]  
    return reversed_string  
  
user_input = input("Enter a string: ")  
result = reverse_string(user_input)  
print("Original String:", user_input)  
print("Reversed String:", result)
```

output:-

```
Enter a string: 56  
  
Original String: 56  
  
Reversed String: 65
```

2. Check if a string is a palindrome.

python code:-

```
def is_palindrome(input_string):  
    cleaned_string = ''.join(input_string.split()).lower()  
    return cleaned_string == cleaned_string[::-1]  
  
user_input = input("Enter a string: ")  
result = is_palindrome(user_input)  
  
if result:  
    print("The string is a palindrome.")  
else:
```

```
print("The string is not a palindrome.")
```

output:-

Enter a string: 2

The string is a palindrome.

3. Convert a string to uppercase.

Python code:-

```
def convert_to_uppercase(input_string):  
    uppercase_string = input_string.upper()  
    return uppercase_string  
  
user_input = input("Enter a string: ")  
result = convert_to_uppercase(user_input)  
print("Original String:", user_input)  
print("Uppercase String:", result)
```

Output:-

Enter a string: dilkhush

Original String: dilkhush

Uppercase String: DILKHUSH

4. Convert a string to lowercase.

Python code:-

```
input_string = "Hello, World!"  
  
lowercase_string = input_string.lower()  
  
print(lowercase_string)
```

Output:-

hello, world!

5. Count the number of vowels in a string.

Python code:-

```
def count_vowels(input_string):  
  
    vowels = set("aeiouAEIOU")  
  
    vowel_count = 0  
  
    for char in input_string:  
  
        if char in vowels:  
  
            vowel_count += 1  
  
    return vowel_count  
  
input_string = "Hello, World!"  
  
print("Number of vowels:", count_vowels(input_string))
```

Output:-

Number of vowels: 3

6. Count the number of consonants in a string.

Python code:-

```
def count_consonants(input_string):  
  
    vowels = set("aeiouAEIOU")  
  
    consonant_count = 0  
  
    for char in input_string:  
  
        if char.isalpha() and char not in vowels:  
  
            consonant_count += 1  
  
    return consonant_count  
  
input_string = "Hello, World!"  
  
print("Number of consonants:", count_consonants(input_string))
```

Output:-

Number of consonants: 7

7. Remove all whitespaces from a string.

Python code:-

```
def remove_whitespace(input_string):  
    return input_string.replace(" ", "")  
  
input_string = " Hello, World! "  
  
print("Original string:", input_string)  
  
print("String with whitespaces removed:", remove_whitespace(input_string))
```

Output:-

Original string: Hello, World!

String with whitespaces removed:
Hello,World!

8. Find the length of a string without using the `len()` function.

Python code:-

```
def string_length(input_string):  
    length = 0  
  
    for char in input_string:  
        length += 1  
  
    return length  
  
input_string = "Hello, World!"  
  
print("Length of the string:", string_length(input_string))
```

Output:-

Length of the string: 13

9. Check if a string contains a specific word.

Python code:-

```
def contains_word(input_string, word):  
    position = input_string.find(word)  
    return position != -1  
  
input_string = "Hello, World! This is a test string."  
word_to_check = "test"  
  
print("Does the string contain the word '{}'?".format(word_to_check), contains_word(input_string, word_to_check))
```

Output:-

Does the string contain the word 'test'?
True

10. Replace a word in a string with another word.

Python code:-

```
def replace_word(input_string, old_word, new_word):  
    return input_string.replace(old_word, new_word)  
  
input_string = "Hello, World! This is a test string."  
old_word = "test"  
new_word = "example"  
  
print("Original string:", input_string)  
  
updated_string = replace_word(input_string, old_word, new_word)  
  
print("Updated string:", updated_string)
```

Output:-

Original string: Hello, World! This is a test string.

Updated string: Hello, World! This is a example string.

11. Count the occurrences of a word in a string.

Python code:-

```
def count_word_occurrences(input_string, word):
```

```
    words = input_string.split()
```

```
    count = 0
```

```
    for w in words:
```

```
        if w == word:
```

```
            count += 1
```

```
    return count
```

```
input_string = "This is a test string. This string is just a test."
```

```
word_to_count = "test"
```

```
print("Number of occurrences of the word '{}':".format(word_to_count), count_word_occurrences(input_string, word_to_count))
```

Output:-

Number of occurrences of the word 'test': 1

12. Find the first occurrence of a word in a string.

Python code:-

```
def find_first_occurrence(input_string, word):
```

```
    words = input_string.split()
```

```

for index, w in enumerate(words):

    if w == word:

        return index

return -1

input_string = "This is a test string. This string is just a test."

word_to_find = "test"

print("Index of the first occurrence of the word '{}':".format(word_to_find), find_first_occurrence(input_string,
word_to_find))

```

Output:-

Index of the first occurrence of the word 'test': 3

13. Find the last occurrence of a word in a string.

Python code:-

```

def find_last_occurrence(input_string, word):

    return input_string.rfind(word)

input_string = "This is a test string. This string is just a test."

word_to_find = "test"

print("Index of the last occurrence of the word '{}':".format(word_to_find), find_last_occurrence(input_string,
word_to_find))

```

Output:-

Index of the last occurrence of the word 'test': 45

14. Split a string into a list of words.

Python code:-

```
def split_into_words(input_string):  
    words = input_string.split()  
    return words  
  
input_string = "This is a test string. This string is just a test."  
  
words_list = split_into_words(input_string)  
  
print("List of words:", words_list)
```

Output:-

```
List of words: ['This', 'is', 'a', 'test', 'string.',  
'This', 'string', 'is', 'just', 'a', 'test.']
```

15. Join a list of words into a string.

Python code:-

```
def join_words(words_list):  
    joined_string = ' '.join(words_list)  
    return joined_string  
  
words_list = ['This', 'is', 'a', 'test', 'string.', 'This', 'string', 'is', 'just', 'a', 'test.']  
  
joined_string = join_words(words_list)  
  
print("Joined string:", joined_string)
```

Output:-

```
Joined string: This is a test string. This string is just a test.
```

16. Convert a string where words are separated by spaces to one where words are separated by underscores.

Python code:-

```
def convert_spaces_to_underscores(input_string):  
    converted_string = input_string.replace(" ", "_")
```



```
    return converted_string

input_string = "This is a test string."

converted_string = convert_spaces_to_underscores(input_string)

print("Converted string:", converted_string)
```

Output:-

Converted string: This_is_a_test_string.

17. Check if a string starts with a specific word or phrase.

Python code:-

```
def starts_with(input_string, prefix):

    return input_string.startswith(prefix)

input_string = "Hello, World! This is a test string."

prefix_to_check = "Hello"

print("Does the string start with the word or phrase '{}'?".format(prefix_to_check), starts_with(input_string,
prefix_to_check))
```

Output:-

Does the string start with the word or phrase 'Hello'? True

18. Check if a string ends with a specific word or phrase.

Python code:-

```
def ends_with(input_string, suffix):

    return input_string.endswith(suffix)

input_string = "Hello, World! This is a test string."

suffix_to_check = "string."

print("Does the string end with the word or phrase '{}'?".format(suffix_to_check), ends_with(input_string,
suffix_to_check))
```

Output:-

Does the string end with the word or phrase 'string.'? True

19. Convert a string to title case (e.g., "hello world" to "Hello World").

Python code:-

```
def convert_to_title_case(input_string):  
    title_case_string = input_string.title()  
    return title_case_string  
  
input_string = "hello world"  
  
title_case_string = convert_to_title_case(input_string)  
  
print("Original string:", input_string)  
  
print("String in title case:", title_case_string)
```

Output:-

Original string: hello world

String in title case: Hello World

20. Find the longest word in a string.

Python code:-

```
def longest_word(s):  
    words = s.split()  
    longest = ""  
    max_length = 0  
    for word in words:  
        if len(word) > max_length:  
            max_length = len(word)  
            longest = word
```

```
    return longest

input_string = "This is a sample string with some words."

print("Longest word:", longest_word(input_string))
```

Output:-
Longest word: sample

21. Find the shortest word in a string.

Python code:-

```
def shortest_word(s):

    words = s.split()

    shortest = None

    shortest_length = float('inf') # Initialize to positive infinity

    for word in words:

        if len(word) < shortest_length:

            shortest = word

            shortest_length = len(word)

    return shortest

input_string = "Find the shortest word in this string"

print("Shortest word:", shortest_word(input_string))
```

Output:-
Shortest word: in

22. Reverse the order of words in a string.

Python code:-

```
def reverse_words(s):

    words = s.split()

    words.reverse()
```

```
reversed_string = ''.join(words)

return reversed_string

input_string = "Reverse the order of words in this string"

print("Reversed string:", reverse_words(input_string))
```

Output:-

Reversed string: string this in words of order the Reverse

23. Check if a string is alphanumeric.

Python code:-

```
def is_alphanumeric(s):

    return s.isalnum()

test_string = "Hello123"

print("Is the string alphanumeric?", is_alphanumeric(test_string))
```

Output:-

Is the string alphanumeric? True

24. Extract all digits from a string.

Python code:-

```
def extract_digits(s):

    digits = ""

    for char in s:

        if char.isdigit():

            digits += char

    return digits

input_string = "abc123def456ghi"
```

```
print("Extracted digits:", extract_digits(input_string))
```

Output:-

Extracted digits: 123456

25. Extract all alphabets from a string.

Python code:-

```
def extract_alphabets(s):  
    alphabets = ""  
    for char in s:  
        if char.isalpha():  
            alphabets += char  
    return alphabets  
  
input_string = "abc123def456ghi"  
print("Extracted alphabets:", extract_alphabets(input_string))
```

Output:-

Extracted alphabets: abcdefghi

26. Count the number of uppercase letters in a string.

Python code:-

```
def count_uppercase_letters(s):  
    count = 0  
    for char in s:  
        if char.isupper():  
            count += 1  
    return count  
  
input_string = "Hello World! How Are You?"  
print("Number of uppercase letters:", count_uppercase_letters(input_string))
```

Output:-

Number of uppercase letters: 5

27. Count the number of lowercase letters in a string.

Python code:-

```
def count_lowercase_letters(s):  
    count = 0  
    for char in s:  
        if char.islower():  
            count += 1  
    return count  
  
input_string = "Hello World! How Are You?"  
  
print("Number of lowercase letters:", count_lowercase_letters(input_string))
```

Output:-

Number of lowercase letters: 14

28. Swap the case of each character in a string.

Python code:-

```
def swap_case(s):  
    return s.swapcase()  
  
input_string = "Hello World! How Are You?"  
  
print("Swapped case string:", swap_case(input_string))
```

Output:-

Swapped case string: hELLO wORLD! hOW aRE yOU?

29. Remove a specific word from a string.

Python code:-

```
def remove_word(string, word):  
    return string.replace(word, "")  
  
input_string = "This is a sample sentence with a specific word."  
  
word_to_remove = "specific"  
  
print("Original string:", input_string)  
  
print("String with '{}' removed:".format(word_to_remove), remove_word(input_string, word_to_remove))
```

Output:-

Original string: This is a sample sentence with a specific word.

String with 'specific' removed: This is a sample sentence with a word.

30. Check if a string is a valid email address.

Python code:-

```
import re  
  
def is_valid_email(email):  
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'  
    if re.match(pattern, email):  
        return True  
    else:  
        return False  
  
email_address = "dk1322268@gmail.com"  
  
print("Is '{}' a valid email address?".format(email_address), is_valid_email(email_address))
```

Output:-

Is 'dk1322268@gmail.com' a valid email address? True

31. Extract the username from an email address string.

Python code:-

```
def extract_username(email):  
    parts = email.split('@')  
    username = parts[0]  
    return username  
  
email_address = "example@email.com"  
  
print("Username extracted from '{}' is:".format(email_address), extract_username(email_address))
```

Output:-

Username extracted from 'example@email.com' is: example

32. Extract the domain name from an email address string.

Python code:-

```
def extract_domain(email):  
    parts = email.split('@')  
    domain = parts[1]  
    return domain  
  
email_address = "example@email.com"  
  
print("Domain extracted from '{}' is:".format(email_address), extract_domain(email_address))
```

Output:-

Domain extracted from 'example@email.com' is: email.com

33. Replace multiple spaces in a string with a single space.

Python code:-

```
import re

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

input_string = "This is a string with multiple spaces."

print("String with multiple spaces replaced:", replace_multiple_spaces(input_string))
```

Output:-

String with multiple spaces replaced: This is a string with multiple spaces.

34. Check if a string is a valid URL.

Python code:-

```
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.example.com"

print("Is '{}' a valid URL?".format(url), is_valid_url(url))
```

Output:-

Is 'https://www.example.com' a valid URL? True

35. Extract the protocol (http or https) from a URL string.

Python code:-

```
from urllib.parse import urlparse

def extract_protocol(url):

    result = urlparse(url)

    return result.scheme if result.scheme in ["http", "https"] else None

url = "https://www.example.com/path/to/resource"

print("Protocol extracted from '{}':".format(url), extract_protocol(url))
```

Output:-

```
Protocol extracted from
'https://www.example.com/path/to/resource': https
```

36. Find the frequency of each character in a string.

Python code:-

```
def character_frequency(string):

    frequency = {}

    for char in string:

        frequency[char] = frequency.get(char, 0) + 1

    return frequency

input_string = "hello world"

print("Character frequency:", character_frequency(input_string))
```

Output:-

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

37. Remove all punctuation from a string.

Python code:-

```
import string

def remove_punctuation(string):

    translator = str.maketrans("", "", string.punctuation)

    return string.translate(translator)

input_string = "Hello, world! This is a test string with punctuation."

print("String with punctuation removed:", remove_punctuation(input_string))
```

Output:-

Traceback (most recent call last):

File "c:\Users\klp\Desktop\python\pw37.py", line 14, in <module>

print("String with punctuation removed:", remove_punctuation(input_string))

38. Check if a string contains only digits.

Python code:-

```
def contains_only_digits(s):

    return s.isdigit()

test_string = "12345"

print("Does '{}' contain only digits?".format(test_string), contains_only_digits(test_string))

test_string = "123abc"

print("Does '{}' contain only digits?".format(test_string), contains_only_digits(test_string))
```

Output:-

Does '12345' contain only digits? True

Does '123abc' contain only digits? False

39. Check if a string contains only alphabets.

Python code:-

```
def contains_only_alphabets(s):  
    return s.isalpha()  
  
test_string = "abcdef"  
  
print("Does '{}' contain only alphabetic characters?".format(test_string), contains_only_alphabets(test_string))  
  
test_string = "123abc"  
  
print("Does '{}' contain only alphabetic characters?".format(test_string), contains_only_alphabets(test_string))
```

Output:-

```
Does 'abcdef' contain only alphabetic characters? True  
Does '123abc' contain only alphabetic characters? False
```

40. Convert a string to a list of characters.

Python code:-

```
def string_to_list(string):  
    return list(string)  
  
input_string = "Hello"  
  
print("String converted to a list of characters:", string_to_list(input_string))
```

Output:-

```
String converted to a list of characters:  
['H', 'e', 'l', 'l', 'o']
```

41. Check if two strings are anagrams.

Python code:-

```
def are_anagrams(str1, str2):  
    str1 = str1.lower().replace(" ", "")  
    str2 = str2.lower().replace(" ", "")
```

```

return sorted(str1) == sorted(str2)

string1 = "listen"

string2 = "silent"

print("Are '{}' and '{}' anagrams?".format(string1, string2), are_anagrams(string1, string2))


string1 = "hello"

string2 = "world"

print("Are '{}' and '{}' anagrams?".format(string1, string2), are_anagrams(string1, string2))

```

Output:-

```

Are 'listen' and 'silent' anagrams? True

Are 'hello' and 'world' anagrams? False

```

42. Encode a string using a Caesar cipher.

Python code:-

```

def caesar_cipher_encrypt(text, shift):

    encrypted_text = ""

    for char in text:

        if char.isalpha():

            shifted_char = chr((ord(char) - 65 + shift) % 26 + 65) if char.isupper() else chr((ord(char) - 97 + shift) % 26 + 97)

            encrypted_text += shifted_char

        else:

            encrypted_text += char

    return encrypted_text

plaintext = "Hello, World!"

shift = 3

encrypted_text = caesar_cipher_encrypt(plaintext, shift)

print("Original:", plaintext)

print("Encrypted with Caesar cipher (shift={}):".format(shift), encrypted_text)

```

Output:-

Original: Hello, World!

Encrypted with Caesar cipher (shift=3): Koor, Zruog!

43. Decode a Caesar cipher encoded string.

Python code:-

```
def caesar_cipher_decrypt(text, shift):  
    decrypted_text = ""  
    for char in text:  
        if char.isalpha():  
            shifted_char = chr((ord(char) - 65 - shift) % 26 + 65) if char.isupper() else chr((ord(char) - 97 - shift) % 26 + 97)  
            decrypted_text += shifted_char  
        else:  
            decrypted_text += char  
    return decrypted_text  
  
encrypted_text = "Koor, Zruog!"  
  
shift = 3  
  
decrypted_text = caesar_cipher_decrypt(encrypted_text, shift)  
print("Original (encoded):", encrypted_text)  
print("Decrypted with Caesar cipher (shift={}):".format(shift), decrypted_text)
```

Output:-

Original (encoded): Koor, Zruog!

Decrypted with Caesar cipher (shift=3): Hello, World!

44. Find the most frequent word in a string.

Python code:-

```
def most_frequent_word(s):  
    words = s.lower().split()
```

```

word_freq = {}

for word in words:

    word_freq[word] = word_freq.get(word, 0) + 1

max_freq_word = max(word_freq, key=word_freq.get)

return max_freq_word

input_string = "This is a test string. This string contains several words, but some words appear more than once."

print("Most frequent word:", most_frequent_word(input_string))

```

Output:-

Most frequent word: this

45. Find all unique words in a string.

Python code:-

```

def unique_words(s):

    words = s.lower().split()

    unique_word_set = set(words)

    return unique_word_set

input_string = "This is a test string. This string contains several words, but some words appear more than once."

print("Unique words:", unique_words(input_string))

```

Output:-

Unique words: {'contains', 'words,', 'but', 'string', 'test', 'this', 'some', 'appear', 'a', 'more', 'several', 'than', 'once.', 'is', 'words', 'string.'}

46. Count the number of syllables in a string.

Python code:-

```

def count_syllables(word):

    word = word.lower()

```

```

vowels = "aeiouy"

syllables = 0

for i in range(len(word)):

    if word[i] in vowels and (i == 0 or word[i - 1] not in vowels):

        syllables += 1

if word.endswith("e"):

    syllables -= 1

if syllables == 0:

    syllables = 1

return syllables

word = "banana"

print("Number of syllables in '{}' is:".format(word), count_syllables(word))


word = "programming"

print("Number of syllables in '{}' is:".format(word), count_syllables(word))

```

Output:-

Number of syllables in 'banana' is: 3

Number of syllables in 'programming' is: 3

47. Check if a string contains any special characters.

Python code:-

```

import re

def contains_special_characters(s):

    pattern = r'^a-zA-Z0-9\s]'

    match = re.search(pattern, s)

    return bool(match)

input_string = "Hello World! This string contains special characters like @ and #."

print("Does '{}' contain special characters?".format(input_string), contains_special_characters(input_string))

```


Output:-

Does 'Hello World! This string contains special characters like @ and #.' contain special characters? True

48. Remove the nth word from a string.

Python code:-

```
def remove_nth_word(s, n):  
    words = s.split()  
  
    if n >= 0 and n < len(words):  
        del words[n]  
  
        return ' '.join(words)  
  
    else:  
        return "Invalid index or string doesn't have enough words."  
  
input_string = "This is a sample string with some words."  
  
index_to_remove = 2 # Index of the word to be removed  
  
print("String with nth word removed:", remove_nth_word(input_string, index_to_remove))
```

Output:-

String with nth word removed: This is sample string with some words.

49. Insert a word at the nth position in a string.

Python code:-

```
def insert_word_at_nth_position(s, word, n):  
    words = s.split()  
  
    if n >= 0 and n <= len(words):  
        words.insert(n, word)  
  
        return ' '.join(words)  
  
    else:  
        return "Invalid index."
```

```
input_string = "This is a sample string with some words."

word_to_insert = "new"

index_to_insert = 3

print("String with word inserted at nth position:", insert_word_at_nth_position(input_string, word_to_insert,
index_to_insert))
```

Output:-

String with word inserted at nth position: This is a new sample string with some words.

50. Convert a CSV string to a list of lists.

Python code:-

```
def csv_string_to_list_of_lists(csv_string):

    lines = csv_string.strip().split('\n')

    result = []

    for line in lines:

        values = line.split(',')

        result.append(values)

    return result

csv_string = """1,John,Doe
2,Jane,Smith
3,Alice,Johnson"""

csv_list = csv_string_to_list_of_lists(csv_string)

print("CSV string converted to list of lists:", csv_list)
```

Output:-

CSV string converted to list of lists: [['1', 'John', 'Doe'], ['2', 'Jane', 'Smith'], ['3', 'Alice', 'Johnson']]

List Based Practice Problem :

1. Create a list with integers from 1 to 10.

Python code:-

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(numbers)
```

Output:-

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

2. Find the length of a list without using the `len()` function.

Python code:-

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
length = 0
for _ in numbers:
    length += 1
print("Length of the list:", length)
```

Output:-

```
Length of the list: 10
```

3. Append an element to the end of a list.

Python code:-

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

Output:-

```
[1, 2, 3, 4, 5, 6]
```

4. Insert an element at a specific index in a list.

Python code:-

```
numbers = [1, 2, 3, 4, 5]
numbers.insert(2, 10)
print(numbers)
```

Output:-

```
[1, 2, 10, 3, 4, 5]
```

5. Remove an element from a list by its value.

Python code:-

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

Output:-

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

6. Remove an element from a list by its index.

Python code:-

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

Output:-

Removed element: 3

Updated list: [1, 2, 4, 5]

7. Check if an element exists in a list.

Python code:-

```
numbers = [1, 2, 3, 4, 5]
if 3 in numbers:
    print("3 exists in the list.")
```

else:

```
print("3 does not exist in the list.")
```

Output:-

3 exists in the list.

8. Find the index of the first occurrence of an element in list..

Python code:-

```
numbers = [4, 7, 2, 9, 4, 5, 6]
```

```
first_index = numbers.index(4)
```

```
print("Index of the first occurrence of 4:", first_index)
```

Output:-

Index of the first occurrence of 4: 0

9.Count the occurrences of an element in a list.

Python code:-

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

```
count = numbers.count(2)
```

```
print("Number of occurrences of 2:", count)
```

Output:-

Number of occurrences of 2: 4

10.Reverse the order of elements in a list.

Python code:-

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

```
numbers.reverse()
```

```
print("Reversed list:", numbers)
```

Output:-

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

11. Sort a list in ascending order.

Python code:-

```
numbers = [5, 2, 8, 1, 3]

numbers.sort()

print("Sorted list in ascending order:", numbers)
```

Output:-

Sorted list in ascending order: [1, 2, 3, 5, 8]

12. Sort a list in descending order.

Python code:-

```
numbers = [5, 2, 8, 1, 3]

numbers.sort(reverse=True)

print("Sorted list in descending order:", numbers)
```

Output:-

Sorted list in descending order: [8, 5, 3, 2, 1]

13. Create a list of even numbers from 1 to 20.

Python code:-

```
even_numbers = [num for num in range(1, 21) if num % 2 == 0]

print(even_numbers)
```

output:-

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

14. Create a list of odd numbers from 1 to 20.

Python code:-

```
odd_numbers = [num for num in range(1, 21) if num % 2 != 0]

print(odd_numbers)
```

Output:-

[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

15. Find the sum of all elements in a list.

Python code:-

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

total_sum = sum(numbers)

print("Sum of all elements:", total_sum)
```

Output:-

Sum of all elements: 15

16. Find the maximum value in a list.

Python code:-

```
numbers = [1, 3, 7, 5, 9, 2, 6]

max_value = max(numbers)

print("Maximum value in the list:", max_value)
```

Output:-

Maximum value in the list: 9

17. Find the minimum value in a list.

Python code:-

```
numbers = [1, 3, 7, 5, 9, 2, 6]

min_value = min(numbers)

print("Minimum value in the list:", min_value)
```

Output:-

Minimum value in the list: 1

18. Create a list of squares of numbers from 1 to 10.

Python code:-

```
squares = [num**2 for num in range(1, 11)]

print(squares)
```

Output:-

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

19. Create a list of random numbers.

Python code:-

```
import random

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

print(random_numbers)
```

Output:-

```
[11, 27, 15, 32, 84, 29, 35, 55, 74, 76]
```

20. Remove duplicates from a list.

Python code:-

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

unique_list = []

for item in original_list:

    if item not in unique_list:

        unique_list.append(item)

print(unique_list)
```

Output:-

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

21. Find the common elements between two lists.

Python code:-

```
list1 = [1, 2, 3, 4, 5]

list2 = [3, 4, 5, 6, 7]

common_elements = list(set(list1) & set(list2))
```



```
print("Common elements:", common_elements)
```

Output:-

Common elements: [3, 4, 5]

22. Find the difference between two lists.

Python code:-

```
list1 = [1, 2, 3, 4, 5]
```

```
list2 = [3, 4, 5, 6, 7]
```

```
difference = list(set(list1) - set(list2))
```

```
print("Difference between list1 and list2:", difference)
```

Output:-

Difference between list1 and list2: [1, 2]

23. Merge two lists.

Python code:-

```
list1 = [1, 2, 3]
```

```
list2 = [4, 5, 6]
```

```
merged_list = list1 + list2
```

```
print("Merged list:", merged_list)
```

Output:-

Merged list: [1, 2, 3, 4, 5, 6]

24. Multiply all elements in a list by 2.

Python code:-

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

```
modified_list = [x * 2 for x in original_list]
```

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

Output:-

Modified list: [2, 4, 6, 8, 10]

25. Filter out all even numbers from a list.

Python code:-

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

```
filtered_list = [x for x in original_list if x % 2 != 0]
```

```
print("Filtered list:", filtered_list)
```

Output:-

Filtered list: [1, 3, 5, 7, 9]

26. Convert a list of strings to a list of integers.

Python code:-

```
string_list = ["1", "2", "3", "4", "5"]
```

```
integer_list = [int(x) for x in string_list]
```

```
print("List of integers:", integer_list)
```

Output:-

List of integers: [1, 2, 3, 4, 5]

27. Convert a list of integers to a list of strings.

Python code:-

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

```
string_list = [str(x) for x in integer_list]
```

```
print("List of strings:", string_list)
```

Output:-

List of strings: ['1', '2', '3', '4', '5']

28. Flatten a nested list.

Python code:-

```
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)
```

Output:-

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

29. Create a list of the first 10 Fibonacci numbers.

Python code:-

```
def fibonacci(n):
```

```
    fib_sequence = [0, 1]
```

```
    for i in range(2, n):
```

```
        fib_sequence.append(fib_sequence[-1] + fib_sequence[-2])
```

```
    return fib_sequence[:n]
```

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

```
print("First 10 Fibonacci numbers:", first_10_fibonacci)
```

Output:-

First 10 Fibonacci numbers: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

30. Check if a list is sorted.

Python code:-

```
def is_sorted(arr):  
    for i in range(len(arr) - 1):  
        if arr[i] > arr[i + 1]:  
            return False  
    return True  
  
sorted_list = [1, 2, 3, 4, 5]  
  
unsorted_list = [5, 2, 8, 1, 3]  
  
print("Is sorted_list sorted?", is_sorted(sorted_list))  
  
print("Is unsorted_list sorted?", is_sorted(unsorted_list))
```

Output:-

Is sorted_list sorted? True

Is unsorted_list sorted? False

31. Rotate a list to the left by `n` positions.

Python code:-

```
def rotate_left(arr, n):  
    return arr[n:] + arr[:n]  
  
original_list = [1, 2, 3, 4, 5]  
  
n = 2  
  
rotated_list = rotate_left(original_list, n)  
  
print("Original list:", original_list)  
  
print("Rotated list:", rotated_list)
```

Output:-

Original list: [1, 2, 3, 4, 5]

Rotated list: [3, 4, 5, 1, 2]

32. Rotate a list to the right by `n` positions.

Python code:-

```
def rotate_right(arr, n):
```

```
    return arr[-n:] + arr[:-n]
```

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

```
n = 2
```

```
rotated_list = rotate_right(original_list, n)
```

```
print("Original list:", original_list)
```

```
print("Rotated list:", rotated_list)
```

Output:-

Original list: [1, 2, 3, 4, 5]

Rotated list: [4, 5, 1, 2, 3]

33. Create a list of prime numbers up to 50.

Python code:-

```
def is_prime(n):
```

```
    if n <= 1:
```

```
        return False
```

```
    elif n <= 3:
```

```
        return True
```

```
    elif n % 2 == 0 or n % 3 == 0:
```

```
        return False
```

```
i = 5
```

```
while i * i <= n:
```

```
    if n % i == 0 or n % (i + 2) == 0:
```

```
        return False
```

```
    i += 6
```

```
return True
```

```
prime_numbers = [x for x in range(2, 51) if is_prime(x)]
```

```
print("Prime numbers up to 50:", prime_numbers)
```

Output:-

```
Prime numbers up to 50: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

34. Split a list into chunks of size `n`.

Python code:-

```
def split_into_chunks(lst, n):
```

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

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

```
chunk_size = 3
```

```
chunks = split_into_chunks(original_list, chunk_size)
```

```
print("Original list:", original_list)
```

```
print("Chunks of size", chunk_size, ":", chunks)
```

Output:-

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

```
Chunks of size 3 : [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10]]
```

35. Find the second largest number in a list.

Python code:-

```
def second_largest(arr):  
    sorted_arr = sorted(arr, reverse=True)  
    if len(sorted_arr) < 2:  
        return "List doesn't have a second largest element"  
    return sorted_arr[1]  
  
numbers = [10, 5, 20, 8, 15]  
  
print("Second largest number:", second_largest(numbers))
```

Output:-

Second largest number: 15

36. Replace every element in a list with its square.

Python code:-

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

Output:-

Original list: [1, 2, 3, 4, 5]

Squared list: [1, 4, 9, 16, 25]

37. Convert a list to a dictionary where list elements become keys and their indices become values.

Python code:-

```
original_list = ['a', 'b', 'c', 'd', 'e']

result_dict = {value: index for index, value in enumerate(original_list)}

print("Original list:", original_list)

print("Dictionary:", result_dict)
```

output:-

Original list: ['a', 'b', 'c', 'd', 'e']

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

38. Shuffle the elements of a list randomly.

Python code:-

```
import random

original_list = [1, 2, 3, 4, 5]

random.shuffle(original_list)

print("Shuffled list:" original_list)
```

Output:-

Shuffled list: [5, 4, 3, 2, 1]

39. Create a list of the first 10 factorial numbers.

Python code:-

```
def factorial(n):
```

```
    if n == 0:
```

```
        return 1
```



```
result = 1
```

```
for i in range(1, n + 1):
```

```
    result *= i
```

```
return result
```

```
factorial_numbers = [factorial(n) for n in range(10)]
```

```
print("First 10 factorial numbers:", factorial_numbers)
```

Output:-

First 10 factorial numbers: [1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880]

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

Python code:-

```
def have_common_element(list1, list2):
```

```
    intersection = set(list1) & set(list2)
```

```
    return len(intersection) > 0
```

```
list1 = [1, 2, 3, 4, 5]
```

```
list2 = [4, 5, 6, 7, 8]
```

```
if have_common_element(list1, list2):
```

```
    print("The lists have at least one common element.")
```

```
else:
```

```
    print("The lists do not have any common elements.")
```

Output:-

The lists have at least one common element.

41. Remove all elements from a list.

Python code:-

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

```
original_list = []
```

```
print("List after removal:", original_list)
```

output:-

List after removal: []

42. Replace negative numbers in a list with 0.

Python code:-

```
original_list = [1, -2, 3, -4, 5, -6]
```

```
modified_list = [x if x >= 0 else 0 for x in original_list]
```

```
print("Original list:", original_list)
```

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

Output:-

Original list: [1, -2, 3, -4, 5, -6]

Modified list: [1, 0, 3, 0, 5, 0]

43. Convert a string into a list of words.

Python code:-

```
input_string = "Convert this string into a list of words"
```

```
word_list = input_string.split()
```

```
print("List of words:", word_list)
```

Output:-

List of words: ['Convert', 'this', 'string', 'into', 'a', 'list', 'of', 'words']

44. Convert a list of words into a string.

Python code:-

```
word_list = ['Convert', 'this', 'list', 'of', 'words', 'into', 'a', 'string']  
  
output_string = ''.join(word_list)  
  
print("Output string:", output_string)
```

Output:-

Output string: Convert this list of words into a string

45. Create a list of the first `n` powers of 2.

Python code:-

```
n = 5  
  
powers_of_2 = [2**i for i in range(n)]  
  
print("First", n, "powers of 2:", powers_of_2)
```

Output:-

First 5 powers of 2: [1, 2, 4, 8, 16]

46. Find the longest string in a list of strings.

Python code:-

```
string_list = ['apple', 'banana', 'kiwi', 'orange', 'strawberry']  
  
longest_string = max(string_list, key=len)  
  
print("Longest string:", longest_string)
```

Output:-

Longest string: strawberry

47. Find the shortest string in a list of strings.

Python code:-

```
string_list = ['apple', 'banana', 'kiwi', 'orange', 'strawberry']
```

```
shortest_string = min(string_list, key=len)
```

```
print("Shortest string:", shortest_string)
```

Output:-

Shortest string: kiwi

48. Create a list of the first `n` triangular numbers.

Python code:-

```
def triangular_number(n):
```

```
    return (n * (n + 1)) // 2
```

```
n = 5
```

```
triangular_numbers = [triangular_number(i) for i in range(1, n + 1)]
```

```
print("First", n, "triangular numbers:", triangular_numbers)
```

Output:-

First 5 triangular numbers: [1, 3, 6, 10, 15]

49. Check if a list contains another list as a subsequence.

Python code:-

```
def is_subsequence(lst, subsequence):
```

```
    i, j = 0, 0
```

```
    while i < len(lst) and j < len(subsequence):
```

```
        if lst[i] == subsequence[j]:
```

```
            j += 1
```

```

    i += 1

    return j == len(subsequence)

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

subsequence = [3, 5, 7]

if is_subsequence(main_list, subsequence):

    print("The subsequence", subsequence, "is present in the main list.")

else:

    print("The subsequence", subsequence, "is not present in the main list.")

```

Output:-

The subsequence [3, 5, 7] is present in the main list.

50. Swap two elements in a list by their indices.

Python code:-

```

def swap_elements(arr, index1, index2):

    arr[index1], arr[index2] = arr[index2], arr[index1]

my_list = [1, 2, 3, 4, 5]

index1 = 1

index2 = 3

print("Original list:", my_list)

swap_elements(my_list, index1, index2)

print("List after swapping elements at indices", index1, "and", index2, ":", my_list)

```

Output:-

Original list: [1, 2, 3, 4, 5]

List after swapping elements at indices 1 and 3 : [1, 4, 3, 2, 5]

Tuple Based Practice Problem :-

1. Create a tuple with integers from 1 to 5.

Python code:-

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

```
print(my_tuple)
```

Output:-

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

2. Access the third element of a tuple.

Python code:-

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

```
third_element = my_tuple[2]
```

```
print(third_element)
```

Output:-

```
3
```

3. Find the length of a tuple without using the `len()` function.

Python code:-

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

```
length = 0
```

```
for _ in my_tuple:
```

```
    length += 1
```

```
print("Length of the tuple:", length)
```

Output:-

Length of the tuple: 5

4.Count the occurrences of an element in a tuple.

Python code:-

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

```
element_to_count = 2
```

```
occurrences = my_tuple.count(element_to_count)
```

```
print("Number of occurrences of", element_to_count, "in the tuple:", occurrences)
```

Output-

Number of occurrences of 2 in the tuple: 3

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

Python code:-

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

```
element_to_find = 2
```

```
first_index = my_tuple.index(element_to_find)
```

```
print("Index of the first occurrence of", element_to_find, "in the tuple:", first_index)
```

Output:-

Index of the first occurrence of 2 in the tuple: 1

6. Check if an element exists in a tuple.

Python code:-

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

element_to_check = 3

if element_to_check in my_tuple:

    print(element_to_check, "exists in the tuple.")

else:

    print(element_to_check, "does not exist in the tuple.")
```

Output:-

3 exists in the tuple.

7. Convert a tuple to a list.

Python code:-

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

my_list = list(my_tuple)

print("Tuple converted to list:", my_list)
```

Output:-

Tuple converted to list: [1, 2, 3, 4, 5]

8. Convert a list to a tuple.

Python code:-

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

my_tuple = tuple(my_list)

print("List converted to tuple:", my_tuple)
```

Output:-

List converted to tuple: (1, 2, 3, 4, 5)

9. Unpack the elements of a tuple into variables.

Python code:-

```
my_tuple = (1, 2, 3)

a, b, c = my_tuple

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

Output:-

Unpacked variables: 1 2 3

10. Create a tuple of even numbers from 1 to 10.

Python code:-

```
even_numbers_tuple = tuple(i for i in range(1, 11) if i % 2 == 0)

print("Tuple of even numbers from 1 to 10:", even_numbers_tuple)
```

Output:-

Tuple of even numbers from 1 to 10: (2, 4, 6, 8, 10)

11. Create a tuple of odd numbers from 1 to 10.

Python code:-

```
odd_numbers_tuple = tuple(i for i in range(1, 11) if i % 2 != 0)

print("Tuple of odd numbers from 1 to 10:", odd_numbers_tuple)
```

Output:-

Tuple of odd numbers from 1 to 10: (1, 3, 5, 7, 9)

12. Concatenate two tuples.

Python code:-

```
tuple1 = (1, 2, 3)

tuple2 = (4, 5, 6)

concatenated_tuple = tuple1 + tuple2

print("Concatenated tuple:", concatenated_tuple)
```

Output:-

Concatenated tuple: (1, 2, 3, 4, 5, 6)

13. Repeat a tuple three times.

Python code:-

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

repeated_tuple = original_tuple * 3

print("Repeated tuple three times:", repeated_tuple)
```

Output:-

Repeated tuple three times: (1, 2, 3, 1, 2, 3, 1, 2, 3)

14. Check if a tuple is empty.

Python code:-

```
empty_tuple = ()

if empty_tuple:

    print("Tuple is not empty.")

else:

    print("Tuple is empty.")
```

Output:-

Tuple is empty.

15. Create a nested tuple.

Python code:-

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

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

Output:-

Nested tuple: ((1, 2), (3, 4), (5, 6))

16. Access the first element of a nested tuple.

Python code:-

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

first_element = nested_tuple[0][0]

print("First element of the nested tuple:", first_element)
```

Output:-

First element of the nested tuple: 1

17. Create a tuple with a single element.

Python code:-

```
single_element_tuple = (1,)
print(single_element_tuple)
```

Output:-

(1,)

18. Compare two tuples.

Python code:-

```
tuple1 = (1, 2, 3)
tuple2 = (1, 2, 4)

if tuple1 == tuple2:
    print("The tuples are equal.")
else:
    print("The tuples are not equal.")

if tuple1 != tuple2:
    print("The tuples are not equal.")
else:
    print("The tuples are equal.")

if tuple1 > tuple2:
    print("tuple1 is greater than tuple2.")
else:
    print("tuple1 is not greater than tuple2.")

if tuple1 < tuple2:
    print("tuple1 is less than tuple2.")
else:
```

```
print("tuple1 is not less than tuple2.")
```

Output:-

The tuples are not equal.

The tuples are not equal.

tuple1 is not greater than tuple2.

tuple1 is less than tuple2.

19. Delete a tuple.

Python code:-

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

```
del my_tuple
```

Output:-

```
PS C:\Users\klp\Desktop\python>
```

20. Slice a tuple.

Python code:-

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

```
sliced_tuple = my_tuple[1:4]
```

```
print("Sliced tuple:", sliced_tuple)
```

Output:-

```
Sliced tuple: (2, 3, 4)
```

21. Find the maximum value in a tuple.

Python code:-

```
my_tuple = (5, 2, 8, 1, 9)
```

```
max_value = max(my_tuple)
```

```
print("Maximum value in the tuple:", max_value)
```

Output:-

Maximum value in the tuple: 9

22. Find the minimum value in a tuple.

Python code:-

```
my_tuple = (5, 2, 8, 1, 9)

min_value = min(my_tuple)

print("Minimum value in the tuple:", min_value)
```

Output:-

Minimum value in the tuple: 1

23. Convert a string to a tuple of characters.

Python code:-

```
my_string = "hello"

tuple_of_characters = tuple(my_string)

print("Tuple of characters:", tuple_of_characters)
```

Output:-

Tuple of characters: ('h', 'e', 'l', 'l', 'o')

24. Convert a tuple of characters to a string.

Python code:-

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

string_from_tuple = ''.join(tuple_of_characters)

print("String from tuple of characters:", string_from_tuple)
```

Output:-

String from tuple of characters: hello

25. Create a tuple from multiple data types.

Python code:-

```
mixed_tuple = (1, "hello", 3.14, True)

print("Mixed tuple:", mixed_tuple)
```

Output:-

```
Mixed tuple: (1, 'hello', 3.14, True)
```

26. Check if two tuples are identical.

Python code:-

```
tuple1 = (1, 2, 3)

tuple2 = (1, 2, 3)

if tuple1 == tuple2:

    print("The tuples are identical.")

else:

    print("The tuples are not identical.")
```

Output:-

```
The tuples are identical.
```

27. Sort the elements of a tuple.

Python code:-

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

sorted_tuple = tuple(sorted(my_tuple))

print("Sorted tuple:", sorted_tuple)
```

Output:-

```
Sorted tuple: (1, 2, 3, 4, 5)
```

28. Convert a tuple of integers to a tuple of strings.

Python code:-

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

tuple_of_strings = tuple(str(x) for x in tuple_of_integers)

print("Tuple of strings:", tuple_of_strings)
```

Output:-

Tuple of strings: ('1', '2', '3', '4', '5')

29. Convert a tuple of strings to a tuple of integers.

Python code:-

```
tuple_of_strings = ('1', '2', '3', '4', '5')  
  
tuple_of_integers = tuple(int(x) for x in tuple_of_strings)  
  
print("Tuple of integers:", tuple_of_integers)
```

Output:-

Tuple of integers: (1, 2, 3, 4, 5)

30. Merge two tuples.

Python code:-

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

Output:-

Merged tuple: (1, 2, 3, 4, 5, 6)

31. Flatten a nested tuple.

Python code:-

```
def flatten_tuple(nested_tuple):  
  
    flattened_list = []  
  
    for item in nested_tuple:  
  
        if isinstance(item, tuple):  
  
            flattened_list.extend(flatten_tuple(item))  
  
        else:  
  
            flattened_list.append(item)
```

```
return tuple(flattened_list)
```

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

```
flattened_tuple = flatten_tuple(nested_tuple)
```

```
print("Flattened tuple:", flattened_tuple)
```

Output:-

Flattened tuple: (1, 2, 3, 4, 5, 6, 7)

32. Create a tuple of the first 5 prime numbers.

Python code:-

```
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 = []
```

```
number = 2 # Start with the first prime number
```

```
while len(prime_numbers) < 5:
```

```
    if is_prime(number):
```

```
        prime_numbers.append(number)
```

```
    number += 1
```

```
prime_numbers_tuple = tuple(prime_numbers)
```

```
print("Tuple of the first 5 prime numbers:", prime_numbers_tuple)
```

Output:-

Tuple of the first 5 prime numbers: (2, 3, 5, 7, 11)

33. Check if a tuple is a palindrome.

Python code:-

```
def is_palindrome(my_tuple):  
    return my_tuple == my_tuple[::-1]  
  
tuple1 = (1, 2, 3, 2, 1)  
  
tuple2 = (1, 2, 3, 4, 5)  
  
print("Is tuple1 a palindrome?", is_palindrome(tuple1))  
  
print("Is tuple2 a palindrome?", is_palindrome(tuple2))
```

Output:-

Is tuple1 a palindrome? True

Is tuple2 a palindrome? False

34. Create a tuple of squares of numbers from 1 to 5.

Python code:-

```
squares_tuple = tuple(i ** 2 for i in range(1, 6))  
  
print("Tuple of squares of numbers from 1 to 5:", squares_tuple)
```

Output:-

Tuple of squares of numbers from 1 to 5: (1, 4, 9, 16, 25)

35. Filter out all even numbers from a tuple.

Python code:-

```
my_tuple = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
  
filtered_tuple = tuple(x for x in my_tuple if x % 2 != 0)  
  
print("Tuple with even numbers filtered out:", filtered_tuple)
```

Output:-

Tuple with even numbers filtered out: (1,
3, 5, 7, 9)

36. Multiply all elements in a tuple by 2.

Python code:-

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

multiplied_tuple = tuple(x * 2 for x in original_tuple)

print("Tuple with all elements multiplied by 2:", multiplied_tuple)
```

Output:-

Tuple with all elements multiplied by 2: (2, 4, 6, 8, 10)

37. Create a tuple of random numbers.

Python code:-

```
import random

start_range = 1

end_range = 100

tuple_size = 5

random_tuple = tuple(random.randint(start_range, end_range) for _ in range(tuple_size))

print("Tuple of random numbers:", random_tuple)
```

Output:-

Tuple of random numbers: (73, 98, 94, 41, 93)

38. Check if a tuple is sorted.

Python code:-

```
def is_sorted(my_tuple):

    return my_tuple == tuple(sorted(my_tuple))

tuple1 = (1, 2, 3, 4, 5)

tuple2 = (5, 4, 3, 2, 1)

print("Is tuple1 sorted?", is_sorted(tuple1))

print("Is tuple2 sorted?", is_sorted(tuple2))
```

Output:-

Is tuple1 sorted? True

Is tuple2 sorted? False

39. Rotate a tuple to the left by `n` positions.

Python code:-

```
def rotate_left(my_tuple, n):  
    return my_tuple[n:] + my_tuple[:n]  
  
original_tuple = (1, 2, 3, 4, 5)  
  
n = 2  
  
rotated_tuple = rotate_left(original_tuple, n)  
  
print("Original tuple:", original_tuple)  
  
print("Tuple rotated to the left by", n, "positions:", rotated_tuple)
```

Output:-

Original tuple: (1, 2, 3, 4, 5)

Tuple rotated to the left by 2 positions: (3, 4, 5, 1, 2)

40. Rotate a tuple to the right by `n` positions.

Python code:-

```
def rotate_right(my_tuple, n):  
    return my_tuple[-n:] + my_tuple[:-n]  
  
original_tuple = (1, 2, 3, 4, 5)  
  
n = 2  
  
rotated_tuple = rotate_right(original_tuple, n)  
  
print("Original tuple:", original_tuple)  
  
print("Tuple rotated to the right by", n, "positions:", rotated_tuple)
```

Output:-

Original tuple: (1, 2, 3, 4, 5)

Tuple rotated to the right by 2 positions: (4, 5, 1, 2, 3)

41. Create a tuple of the first 5 Fibonacci numbers.

Python code:-

```
def fibonacci(n):  
    fib = [0, 1]  
    for i in range(2, n):  
        fib.append(fib[i-1] + fib[i-2])  
    return tuple(fib)  
  
fibonacci_tuple = fibonacci(5)  
  
print("Tuple of the first 5 Fibonacci numbers:", fibonacci_tuple)
```

Output:-

Tuple rotated to the right by 2 positions: (4, 5, 1, 2, 3)

42. Create a tuple from user input.

Python code:-

```
user_input_list = input("Enter elements separated by spaces: ").split()  
  
user_tuple = tuple(user_input_list)
```

Output:-

Enter elements separated by spaces: 5 6 1 2 3 7 9

Tuple from user input: ('5', '6', '1', '2', '3', '7', '9')

43. Swap two elements in a tuple.

Python code:-

```
def swap_elements(my_tuple, index1, index2):  
    temp_list = list(my_tuple)  
    temp_list[index1], temp_list[index2] = temp_list[index2], temp_list[index1]  
    return tuple(temp_list)  
  
original_tuple = (1, 2, 3, 4, 5)  
  
index1 = 1  
  
index2 = 3  
  
swapped_tuple = swap_elements(original_tuple, index1, index2)
```

```
print("Original tuple:", original_tuple)
```

```
print("Tuple after swapping elements at indices", index1, "and", index2, ":", swapped_tuple)
```

Output:-

Original tuple: (1, 2, 3, 4, 5)

Tuple after swapping elements at indices 1
and 3 : (1, 4, 3, 2, 5)

44. Reverse the elements of a tuple.

Python code:-

```
def reverse_tuple(my_tuple):
```

```
    return my_tuple[::-1]
```

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

```
reversed_tuple = reverse_tuple(original_tuple)
```

```
print("Original tuple:", original_tuple)
```

```
print("Reversed tuple:", reversed_tuple)
```

Output:-

Original tuple: (1, 2, 3, 4, 5)

Reversed tuple: (5, 4, 3, 2, 1)

45. Create a tuple of the first `n` powers of 2.

Python code:-

```
def powers_of_2(n):
```

```
    return tuple(2 ** i for i in range(n))
```

```
n = 5
```

```
powers_tuple = powers_of_2(n)
```

```
print("Tuple of the first", n, "powers of 2:", powers_tuple)
```

Output;-

Tuple of the first 5 powers of 2: (1, 2, 4, 8, 16)

46. Find the longest string in a tuple of strings.

Python code:-

```
def longest_string_in_tuple(my_tuple):  
    longest = ""  
  
    for string in my_tuple:  
        if len(string) > len(longest):  
            longest = string  
  
    return longest  
  
my_tuple = ("apple", "banana", "orange", "kiwi", "strawberry")  
  
longest_string = longest_string_in_tuple(my_tuple)  
  
print("Longest string in the tuple:", longest_string)
```

Output:-

Longest string in the tuple: strawberry

47. Find the shortest string in a tuple of strings.

Python code:-

```
def shortest_string(strings):  
    return min(strings, key=len)  
  
strings_tuple = ("apple", "banana", "orange", "kiwi")  
  
shortest = shortest_string(strings_tuple)  
  
print("Shortest string:", shortest)
```

Output:-

Shortest string: kiwi

48. Create a tuple of the first `n` triangular numbers.

Python code:-

```
def triangular_numbers(n):  
    return tuple((i * (i + 1)) // 2 for i in range(1, n + 1))  
  
n = 5  
  
triangular_tuple = triangular_numbers(n)
```

```
print("Tuple of the first", n, "triangular numbers:", triangular_tuple)
```

Output:-

Tuple of the first 5 triangular numbers: (1, 3, 6, 10, 15)

49. Check if a tuple contains another tuple as a subsequence.

Python code:-

```
def contains_subsequence(main_tuple, subsequence_tuple):  
    main_length = len(main_tuple)  
    sub_length = len(subsequence_tuple)  
    for i in range(main_length - sub_length + 1):  
        if main_tuple[i:i + sub_length] == subsequence_tuple:  
            return True  
    return False  
  
main_tuple = (1, 2, 3, 4, 5, 6, 7)  
subsequence_tuple = (3, 4, 5)  
  
print("Does the main tuple contain the subsequence tuple?", contains_subsequence(main_tuple,  
subsequence_tuple))
```

Output:-

Does the main tuple contain the subsequence tuple? True

50. Create a tuple of alternating 1s and 0s of length `n`.

Python code:-

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

```
n = 7
```

```
alternating_tuple = alternating_ones_zeros(n)
```

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

Output:-

Tuple of alternating 1s and 0s of length 7 :
(1, 0, 1, 0, 1, 0, 1)

Set Based Practice Problem :-

1. Create a set with integers from 1 to 5.

Python code:-

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

```
print(my_set)
```

Output:-

{1, 2, 3, 4, 5}

2. Add an element to a set.

Python code:-

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

```
my_set.add(6)
```

```
print(my_set)
```

Output:-

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

3. Remove an element from a set.

Python code:-

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

```
my_set.remove(3)
```



```
print(my_set)
```

Output:-

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

4. Check if an element exists in a set.

Python code:-

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

```
if 3 in my_set:
```

```
    print("3 exists in the set")
```

```
else:
```

```
    print("3 does not exist in the set")
```

Output:-

```
3 exists in the set
```

5. Find the length of a set without using the `len()` function.

Python code:-

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

```
count = 0
```

```
for _ in my_set:
```

```
    count += 1
```

```
print("Length of the set:", count)
```

Output:-

```
Length of the set: 5
```

6. Clear all elements from a set.

Python code:-

```
my_set = {1, 2, 3, 4, 5}
my_set.clear()
print(my_set)
```

Output:-

```
set()
```

7. Create a set of even numbers from 1 to 10.

Python code:-

```
even_numbers = {x for x in range(1, 11) if x % 2 == 0}
print(even_numbers)
```

Output:-

```
{2, 4, 6, 8, 10}
```

8. Create a set of odd numbers from 1 to 10.

Python code:-

```
odd_numbers = {x for x in range(1, 11) if x % 2 != 0}
print(odd_numbers)
```

Output:-

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

9. Find the union of two sets.

Python code:-

```
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
union_set = set1.union(set2)
```

```
print(union_set)
```

Output:-

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

10. Find the intersection of two sets.

Python code:-

```
set1 = {1, 2, 3, 4, 5}
```

```
set2 = {4, 5, 6, 7, 8}
```

```
intersection_set = set1.intersection(set2)
```

```
print(intersection_set)
```

Output:-

```
{4, 5}
```

11. Find the difference between two sets.

Python code:-

```
set1 = {1, 2, 3, 4, 5}
```

```
set2 = {4, 5, 6, 7, 8}
```

```
difference_set = set1.difference(set2)
```

```
print(difference_set)
```

Output:-

```
{1, 2, 3}
```

12. Check if a set is a subset of another set.

Python code:-

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

```
set2 = {1, 2, 3, 4, 5}

is_subset = set1.issubset(set2)

print(is_subset)
```

Output:-

True

13. Check if a set is a superset of another set.

Python code:-

```
set1 = {1, 2, 3, 4, 5}

set2 = {1, 2, 3}

is_superset = set1.issuperset(set2)

print(is_superset)
```

Output:-

True

14. Create a set from a list.

Python code:-

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

my_set = set(my_list)

print(my_set)
```

Output:-

{1, 2, 3, 4, 5}

15. Convert a set to a list.

Python code:-

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

```
my_list = list(my_set)
```

```
print(my_list)
```

Output:-

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

16. Remove a random element from a set.

Python code:-

```
import random
```

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

```
random_element = my_set.pop()
```

```
print("Removed element:", random_element)
```

```
print("Set after removal:", my_set)
```

Output:-

```
Removed element: 1
```

```
Set after removal: {2, 3, 4, 5}
```

17. Pop an element from a set.

Python code:-

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

```
popped_element = my_set.pop()
```

```
print("Popped element:", popped_element)
```

```
print("Set after popping:", my_set)
```

Output:-

Popped element: 1

Set after popping: {2, 3, 4, 5}

18. Check if two sets have no elements in common.

Python code:-

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

```
set2 = {4, 5, 6}
```

```
no_common_elements = set1.isdisjoint(set2)
```

```
print(no_common_elements)
```

Output:-

True

19. Find the symmetric difference between two sets.

Python code:-

```
set1 = {1, 2, 3, 4}
```

```
set2 = {3, 4, 5, 6}
```

```
symmetric_difference_set = set1.symmetric_difference(set2)
```

```
print(symmetric_difference_set)
```

Output:-

{1, 2, 5, 6}

20. Update a set with elements from another set.

Python code:-

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

set2 = {3, 4, 5}

set1.update(set2)

print(set1)
```

Output:-

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

21. Create a set of the first 5 prime numbers.

Python code:-

```
def is_prime(n):

    if n <= 1:

        return False

    elif n <= 3:

        return True

    elif n % 2 == 0 or n % 3 == 0:

        return False

    i = 5

    while i * i <= n:

        if n % i == 0 or n % (i + 2) == 0:

            return False

        i += 6

    return True

prime_numbers = {x for x in range(2, 20) if is_prime(x)}

print(prime_numbers)
```

Output:-

```
{2, 3, 5, 7, 11, 13, 17, 19}
```

22. Check if two sets are identical.

Python code:-

```
set1 = {1, 2, 3, 4, 5}
```

```
set2 = {1, 2, 3, 4, 5}
```

```
if set1 == set2:
```

```
    print("The sets are identical.")
```

```
else:
```

```
    print("The sets are not identical.")
```

Output:-

```
The sets are identical.
```

23. Create a frozen set.

Python code:-

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

```
frozen_set = frozenset(my_set)
```

```
print(frozen_set)
```

Output:-

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


24. Check if a set is disjoint with another set.

Python code:-

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

set2 = {4, 5, 6}

if set1.isdisjoint(set2):

    print("The sets are disjoint.")

else:

    print("The sets have elements in common.")
```

Output:-

The sets are disjoint.

25. Create a set of squares of numbers from 1 to 5.

Python code:-

```
squares_set = {x ** 2 for x in range(1, 6)}

print(squares_set)
```

Output:-

{1, 4, 9, 16, 25}

26. Filter out all even numbers from a set.

Python code:-

```
original_set = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

filtered_set = {x for x in original_set if x % 2 != 0}

print(filtered_set)
```

Output:-

{1, 3, 5, 7, 9}

27. Multiply all elements in a set by 2.

Python code:-

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

multiplied_set = {x * 2 for x in original_set}

print(multiplied_set)
```

Output:-
{2, 4, 6, 8, 10}

28. Create a set of random numbers.

Python code:-

```
import random

random_set = set(random.sample(range(1, 101), 5))

print(random_set)
```

Output:-
{35, 40, 15, 82, 20}

29. Check if a set is empty.

Python code:-

```
my_set = set()

if not my_set:

    print("The set is empty.")

else:

    print("The set is not empty.")
```

Output:-

The set is empty.

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

Python code:-

```
nested_set = {frozenset({1, 2}), frozenset({3, 4, 5}), frozenset({6, 7, 8})}
```

```
print(nested_set)
```

Output:-

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

31. Remove an element from a set using the discard method.

Python code:-

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

```
my_set.discard(3)
```

```
print(my_set)
```

Output:-

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

32. Compare two sets.

Python code:-

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

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

```
print("Are the sets equal?", set1 == set2)

print("Are the sets not equal?", set1 != set2)

print("Is set1 a subset of set2?", set1 <= set2)

print("Is set1 a proper subset of set2?", set1 < set2)

print("Is set1 a superset of set2?", set1 >= set2)

print("Is set1 a proper superset of set2?", set1 > set2)
```

Output:-

Are the sets equal? True

Are the sets not equal? False

Is set1 a subset of set2? True

Is set1 a proper subset of set2? False

Is set1 a superset of set2? True

Is set1 a proper superset of set2? False

33. Create a set from a string.

Python code:-

```
my_string = "hello"

my_set = {char for char in my_string}

print(my_set)
```

Output:-

{'e', 'h', 'o', 'l'}

34. Convert a set of strings to a set of integers.

Python code:-

```
set_of_strings = {"1", "2", "3", "4", "5"}

set_of_integers = {int(string) for string in set_of_strings}

print(set_of_integers)
```

Output:-

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

35. Convert a set of integers to a set of strings.

Python code:-

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

```
set_of_strings = {str(integer) for integer in set_of_integers}
```

```
print(set_of_strings)
```

Output:-

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

36. Create a set from a tuple.

Python code:-

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

```
my_set = set(my_tuple)
```

```
print(my_set)
```

Output:-

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

37. Convert a set to a tuple.

Python code:-

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

```
my_tuple = tuple(my_set)
```

```
print(my_tuple)
```

Output:-

(1, 2, 3, 4, 5)

38. Find the maximum value in a set.

Python code:-

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

```
max_value = max(my_set)
```

```
print("Maximum value in the set:", max_value)
```

Output:-

Maximum value in the set: 9

39. Find the minimum value in a set.

Python code:-

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

```
min_value = min(my_set)
```

```
print("Minimum value in the set:", min_value)
```

Output:-

Minimum value in the set: 1

40. Create a set from user input.

Python code:-

```
user_input = input("Enter elements separated by spaces: ")
```

```
elements = user_input.split()
```

```
user_set = set(elements)
```

```
print("Set created from user input:", user_set)
```

Output:-

Enter elements separated by spaces: 1 5 4

2

Set created from user input: {'5', '4', '1',
'2'}

41. Check if the intersection of two sets is empty.

Python code:-

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

```
set2 = {4, 5, 6}
```

```
if set1.isdisjoint(set2):
```

```
    print("The intersection of the sets is empty.")
```

```
else:
```

```
    print("The intersection of the sets is not empty.")
```

Output:-

The intersection of the sets is empty.

42. Create a set of the first 5 Fibonacci numbers.

Python code:-

```
def generate_fibonacci(n):
```

```
    fibonacci_set = set()
```

```
    a, b = 0, 1
```

```
    for _ in range(n):
```

```
        fibonacci_set.add(a)
```

```
        a, b = b, a + b
```

```
    return fibonacci_set
```

```
fibonacci_set = generate_fibonacci(5)

print(fibonacci_set)
```

Output:-
{0, 1, 2, 3}

43. Remove duplicates from a list using sets.

Python code:-

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

unique_list = list(set(original_list))

print(unique_list)
```

Output:-
[1, 2, 3, 4, 5]

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

Python code:-

```
set1 = {1, 2, 3, 4}

set2 = {4, 3, 2, 1}

if set1 == set2:

    print("The sets have the same elements.")

else:

    print("The sets do not have the same elements.")
```

Output:-
The sets have the same elements.

45. Create a set of the first `n` powers of 2.

Python code:-

```
n = 5  
  
powers_of_2_set = {2 ** i for i in range(n)}  
  
print(powers_of_2_set)
```

Output:-

```
{1, 2, 4, 8, 16}
```

46. Find the common elements between a set and a list.

Python code:-

```
my_set = {1, 2, 3, 4, 5}  
  
my_list = [4, 5, 6, 7, 8]  
  
common_elements = my_set.intersection(my_list)  
  
print(common_elements)
```

Output:-

```
{4, 5}
```

47. Create a set of the first `n` triangular numbers.

Python code:-

```
n = 5  
  
triangular_numbers_set = {i * (i + 1) // 2 for i in range(1, n + 1)}  
  
print(triangular_numbers_set)
```

Output:-

```
{1, 3, 6, 10, 15}
```