8. Convert Celsius to Fahrenheit

```
# Prompt user for temperature in Celsius
celsius = float(input("Enter the temperature in Celsius: "))

# Check if input is a valid number
if not math.isnan(celsius):
    # Convert Celsius to Fahrenheit
    fahrenheit = (celsius * 9) / 5 + 32
    print(f"{celsius} degrees Celsius is equal to {fahrenheit} degrees
Fahrenheit.")
else:
    print("Please enter a valid number for the temperature in Celsius.")
```

9. Generate a Random Number

```
import random

# Prompt user for the range
min_range = float(input("Enter the minimum value of the range: "))
max_range = float(input("Enter the maximum value of the range: "))

# Check if input is a valid number
if not (math.isnan(min_range) or math.isnan(max_range) or min_range >=
max_range):
    # Generate a random number within the specified range
    random_number = random.uniform(min_range, max_range)
    print(f"A random number between {min_range} and {max_range} is:
{random_number}")
else:
    print("Please enter valid numbers, ensuring that the minimum value is less
than the maximum value.")
```

10. Check if a number is Positive, Negative, or Zero

```
# Prompt user for a number
number = float(input("Enter a number: "))

# Check if input is a valid number
if not math.isnan(number):
    # Check if the number is positive, negative, or zero
    if number > 0:
        print(f"{number} is a positive number.")
    elif number < 0:
        print(f"{number} is a negative number.")
    else:
        print("The entered number is zero.")
else:
    print("Please enter a valid number.")
```

11. Check if a Number is Odd or Even

```
# Prompt user for a number
number = input("Enter a number: ")

# Check if input is a valid integer
if number.isdigit():
    number = int(number)
    # Check if the number is odd or even
    if number % 2 == 0:
        print(f"{number} is an even number.")
    else:
        print(f"{number} is an odd number.")
else:
    print("Please enter a valid integer.")
```

12. Find the Largest Among Three Numbers

```
# Prompt user for three numbers
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

# Check if inputs are valid numbers
if not (math.isnan(num1) or math.isnan(num2) or math.isnan(num3)):
    # Find the largest among the three numbers
    largest_number = max(num1, num2, num3)
    print(f"The largest number among {num1}, {num2}, and {num3} is:
{largest_number}")
else:
    print("Please enter valid numbers for all three inputs.")
```

13. Check Prime Number

```
# Prompt user for a number
number = int(input("Enter a number: "))

# Check if input is a valid integer greater than 1
if number > 1:
    is_prime = True

    # Check for factors from 2 to the square root of the number
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            is_prime = False
            break

    # Display the result
    if is_prime:
        print(f"{number} is a prime number.")
    else:
        print(f"{number} is not a prime number.")
else:
    print("Please enter a valid integer greater than 1.")
```

14. Print All Prime Numbers in an Interval

```
# Prompt user for the interval
start_number = int(input("Enter the starting number of the interval: "))
end_number = int(input("Enter the ending number of the interval: "))

# Check if inputs are valid integers and the startNumber is less than the
endNumber
if (
    isinstance(start_number, int)
    and isinstance(end_number, int)
    and start_number < end_number
    and start_number > 1
):
    print(f"Prime numbers in the interval [{start_number}, {end_number}]:")

    # Check for prime numbers in the interval
    for i in range(start_number, end_number + 1):
        is_prime = True

        # Check for prime numbers
        for j in range(2, int(i ** 0.5) + 1):
            if i % j == 0:
                is_prime = False
                break

        if is_prime:
            print(i)
else:
    print(
        "Please enter valid integers, ensuring that the starting number is less
than the ending number and greater than 1."
    )
```

15. Find the Factorial of a Number

```
# Prompt user for a non-negative integer
number = int(input("Enter a non-negative integer: "))

# Check if input is a valid non-negative integer
if isinstance(number, int) and number >= 0:
    # Calculate the factorial
    factorial = 1
    for i in range(1, number + 1):
        factorial *= i

    print(f"The factorial of {number} is: {factorial}")
else:
    print("Please enter a valid non-negative integer.")
```

16. Display the Multiplication Table

```
# Prompt user for a number
number = int(input("Enter a number for the multiplication table: "))

# Check if input is a valid integer
if isinstance(number, int):
    # Specify the range for the multiplication table
    range_val = 10

    print(f"Multiplication table for {number} (up to {range_val}):")

    # Display the multiplication table
    for i in range(1, range_val + 1):
        result = number * i
        print(f"{number} x {i} = {result}")
else:
    print("Please enter a valid integer.")
```

17. Print the Fibonacci Sequence

```
# Prompt user for the number of terms in the Fibonacci sequence
num_terms = int(input("Enter the number of terms in the Fibonacci sequence: "))

# Check if input is a valid integer
if isinstance(num_terms, int) and num_terms > 0:
    print(f"Fibonacci sequence of {num_terms} terms:")

    fib_list = [0, 1]

    # Generate the Fibonacci sequence
    for i in range(2, num_terms):
        fib_list.append(fib_list[i - 1] + fib_list[i - 2])

    print(", ".join(map(str, fib_list)))
else:
    print("Please enter a valid positive integer for the number of terms.")
```

18. Check Armstrong Number

```
# Prompt user for a number
number = int(input("Enter a number: "))

# Check if input is a valid positive integer
if isinstance(number, int) and number > 0:
    original_number = number
    number_of_digits = len(str(original_number))
    sum_of_digits = 0

    # Calculate the sum of digits each raised to the power of the number of
digits
    while original_number > 0:
        digit = original_number % 10
        sum_of_digits += digit ** number_of_digits
        original_number //= 10

    # Check if the number is an Armstrong number
    if sum_of_digits == number:
        print(f"{number} is an Armstrong number.")
    else:
        print(f"{number} is not an Armstrong number.")
else:
    print("Please enter a valid positive integer.")
```

19. Find Armstrong Number in an Interval

```
# Prompt user for the interval
start_number = int(input("Enter the starting number of the interval: "))
end_number = int(input("Enter the ending number of the interval: "))

# Check if inputs are valid positive integers and the startNumber is less than
the endNumber
if (
    isinstance(start_number, int)
    and isinstance(end_number, int)
    and start_number > 0
    and start_number < end_number
):
    print(f"Armstrong numbers in the interval [{start_number}, {end_number}]:")

    # Check for Armstrong numbers in the interval
    for i in range(start_number, end_number + 1):
        original_number = i
        number_of_digits = len(str(original_number))
        sum_of_digits = 0

        # Calculate the sum of digits each raised to the power of the number of
digits
        while original_number > 0:
            digit = original_number % 10
            sum_of_digits += digit ** number_of_digits
            original_number //= 10

        # Check if the number is an Armstrong number
        if sum_of_digits == i:
            print(i)
else:
    print(
        "Please enter valid positive integers, ensuring that the starting
number is less than the ending number and both are greater than 0."
    )
```

20. Make a Simple Calculator

```
# Prompt user for two numbers
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Check if inputs are valid numbers
if not (math.isnan(num1) or math.isnan(num2)):
    # Prompt user for the operation
    operation = input("Choose an operation (+ for addition, - for subtraction,
* for multiplication, / for division): ")

    # Perform the selected operation
    if operation in ['+', '-', '*', '/']:
        if operation == '+':
            result = num1 + num2
        elif operation == '-':
            result = num1 - num2
        elif operation == '*':
            result = num1 * num2
        elif operation == '/':
            if num2 != 0:
                result = num1 / num2
            else:
                print("Cannot divide by zero.")
                result = None
    else:
        print("Invalid operation.")
        result = None

    # Display the result
    if result is not None:
        print(f"Result of {num1} {operation} {num2} is: {result}")
else:
    print("Please enter valid numbers.")
```

21. Find the Sum of Natural Numbers

```
# Prompt user for a positive integer
n = int(input("Enter a positive integer: "))

# Check if input is a valid positive integer
if isinstance(n, int) and n > 0:
    # Calculate the sum of natural numbers
    sum_of_natural_numbers = (n * (n + 1)) // 2
    print(f"The sum of natural numbers from 1 to {n} is:
{sum_of_natural_numbers}")
else:
    print("Please enter a valid positive integer.")
```

22. Check if the Numbers Have the Same Last Digit

```
# Prompt user for two numbers
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

# Check if inputs are valid integers
if isinstance(num1, int) and isinstance(num2, int):
    # Extract the last digit of each number
    last_digit_1 = abs(num1 % 10)
    last_digit_2 = abs(num2 % 10)

    # Check if the last digits are the same
    if last_digit_1 == last_digit_2:
        print(f"The last digit of {num1} is the same as the last digit of
{num2}.")
    else:
        print(f"The last digit of {num1} is different from the last digit of
{num2}.")
else:
    print("Please enter valid integers.")
```

23. Find HCF or GCD

```
# Prompt user for two positive integers
num1 = int(input("Enter the first positive integer: "))
num2 = int(input("Enter the second positive integer: "))

# Check if inputs are valid positive integers
if isinstance(num1, int) and isinstance(num2, int) and num1 > 0 and num2 > 0:
    # Find the HCF or GCD
    smaller_number = min(num1, num2)
    hcf = 1

    for i in range(1, smaller_number + 1):
        if num1 % i == 0 and num2 % i == 0:
            hcf = i

    print(f"The HCF (GCD) of {num1} and {num2} is: {hcf}")
else:
    print("Please enter valid positive integers.")
```

24. Find LCM

```python
# Prompt user for two positive integers
num1 = int(input("Enter the first positive integer: "))
num2 = int(input("Enter the second positive integer: "))

# Check if inputs are valid positive integers
if isinstance(num1, int) and isinstance(num2, int) and num1 > 0 and num2 > 0:
    # Find the LCM
    larger_number = max(num1, num2)
    lcm = larger_number

    while True:
        if lcm % num1 == 0 and lcm % num2 == 0:
            print(f"The LCM of {num1} and {num2} is: {lcm}")
            break
        lcm += larger_number
else:
    print("Please enter valid positive integers.")
```

25. Find the Factors of a Number

```
# Prompt user for a positive integer
number = int(input("Enter a positive integer: "))

# Check if input is a valid positive integer
if isinstance(number, int) and number > 0:
    print(f"Factors of {number}:")

    # Find and display the factors
    for i in range(1, number + 1):
        if number % i == 0:
            print(i)
else:
    print("Please enter a valid positive integer.")
```

26. Find Sum of Natural Numbers Using Recursion

```
# Define a recursive function to calculate the sum of natural numbers
def sum_of_natural_numbers(n):
    if n == 1:
        return 1
    else:
        return n + sum_of_natural_numbers(n - 1)

# Prompt user for a positive integer
number = int(input("Enter a positive integer: "))

# Check if input is a valid positive integer
if isinstance(number, int) and number > 0:
    # Calculate and display the sum using recursion
    sum_result = sum_of_natural_numbers(number)
    print(f"The sum of natural numbers up to {number} is: {sum_result}")
else:
    print("Please enter a valid positive integer.")
```

27. Guess a Random Number

```
import random

# Generate a random number between 1 and 100
random_number = random.randint(1, 100)

# Initialize variables
user_guess = None
attempts = 0

# Game loop
while user_guess != random_number:
    # Prompt user for a guess
    user_input = input("Guess the random number (between 1 and 100): ")

    # Check if the input is a valid number
    if user_input.isdigit():
        user_guess = int(user_input)
        attempts += 1

        # Provide feedback to the user
        if user_guess < random_number:
            print("Too low! Try again.")
        elif user_guess > random_number:
            print("Too high! Try again.")
        else:
            print(f"Congratulations! You guessed the correct number
{random_number} in {attempts} attempts.")
    else:
        print("Please enter a valid number.")
```

28. Shuffle Deck of Cards

```
import random

# Function to create a standard deck of cards
def create_deck():
    suits = ["Hearts", "Diamonds", "Clubs", "Spades"]
    ranks = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen",
"King", "Ace"]

    deck = []

    for suit in suits:
        for rank in ranks:
            deck.append(f"{rank} of {suit}")

    return deck

# Function to shuffle the deck of cards
def shuffle_deck(deck):
    random.shuffle(deck)

# Create and display an initial deck of cards
initial_deck = create_deck()
print("Initial Deck:")
print(initial_deck)

# Shuffle the deck and display the shuffled deck
shuffled_deck = initial_deck.copy()  # Create a copy to avoid modifying the
original
shuffle_deck(shuffled_deck)
print("\nShuffled Deck:")
print(shuffled_deck)
```

29. Display Fibonacci Sequence Using Recursion

```
# Function to generate the Fibonacci sequence using recursion
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

# Prompt user for the number of terms in the Fibonacci sequence
num_terms = int(input("Enter the number of terms in the Fibonacci sequence: "))

# Check if input is a valid non-negative integer
if isinstance(num_terms, int) and num_terms >= 0:
    print(f"Fibonacci sequence of {num_terms} terms:")

    # Display the Fibonacci sequence using recursion
    for i in range(num_terms):
        print(fibonacci(i))
else:
    print("Please enter a valid non-negative integer for the number of terms.")
```

30. Find Factorial of Number Using Recursion

```python
# Function to calculate the factorial using recursion
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Prompt user for a non-negative integer
number = int(input("Enter a non-negative integer: "))

# Check if input is a valid non-negative integer
if isinstance(number, int) and number >= 0:
    # Calculate and display the factorial using recursion
    result = factorial(number)
    print(f"The factorial of {number} is: {result}")
else:
    print("Please enter a valid non-negative integer.")
```

31. Convert Decimal to Binary

```
# Function to convert decimal to binary
def decimal_to_binary(decimal_number):
    if decimal_number == 0:
        return "0"

    binary_result = ""
    while decimal_number > 0:
        remainder = decimal_number % 2
        binary_result = str(remainder) + binary_result
        decimal_number = decimal_number // 2

    return binary_result

# Prompt user for a decimal number
decimal_number = int(input("Enter a decimal number: "))

# Check if input is a valid integer
if isinstance(decimal_number, int) and decimal_number >= 0:
    # Convert and display the binary equivalent
    binary_equivalent = decimal_to_binary(decimal_number)
    print(f"The binary equivalent of {decimal_number} is: {binary_equivalent}")
else:
    print("Please enter a valid non-negative integer.")
```

32. Find ASCII Value of Character

```
# Prompt user for a character
character = input("Enter a character: ")

# Check if input is a valid single character
if len(character) == 1:
    # Calculate and display the ASCII value
    ascii_value = ord(character)
    print(f"The ASCII value of '{character}' is: {ascii_value}")
else:
    print("Please enter a valid single character.")
```

33. Check Whether a String is Palindrome or Not

```
# Function to check if a string is a palindrome
def is_palindrome(s):
    # Remove non-alphanumeric characters and convert to lowercase
    clean_str = ''.join(char.lower() for char in s if char.isalnum())

    # Compare the original and reversed strings
    return clean_str == clean_str[::-1]

# Prompt user for a string
input_string = input("Enter a string: ")

# Check if input is a valid string
if len(input_string) > 0:
    # Check and display if the string is a palindrome
    if is_palindrome(input_string):
        print(f'"{input_string}" is a palindrome.')
    else:
        print(f'"{input_string}" is not a palindrome.')
else:
    print("Please enter a valid string.")
```

34. Sort Words in Alphabetical Order

```
# Prompt user for a sentence or a list of words
input_string = input("Enter a sentence or a list of words: ")

# Check if input is a valid string
if len(input_string) > 0:
    # Split the input into a list of words
    words_list = input_string.split()

    # Sort the list of words in alphabetical order
    sorted_words = sorted(words_list)

    # Display the sorted words
    print("Sorted Words:")
    print(", ".join(sorted_words))
else:
    print("Please enter a valid string.")
```

35. Replace Characters of a String

```
# Prompt user for a string
input_string = input("Enter a string: ")

# Check if input is a valid string
if len(input_string) > 0:
    # Prompt user for a target character and a replacement character
    target_char = input("Enter the target character: ")
    replacement_char = input("Enter the replacement character: ")

    # Check if target_char is a single character and input is not empty
    if len(target_char) == 1:
        # Replace all occurrences of target_char with replacement_char
        modified_string = input_string.replace(target_char, replacement_char)

        # Display the modified string
        print(f"Modified String: {modified_string}")
    else:
        print("Please enter a valid target character (single character).")
else:
    print("Please enter a valid string.")
```

36. Reverse a String

```
# Prompt user for a string
input_string = input("Enter a string: ")

# Check if input is a valid string
if len(input_string) > 0:
    # Reverse the string
    reversed_string = input_string[::-1]

    # Display the reversed string
    print(f"Reversed String: {reversed_string}")
else:
    print("Please enter a valid string.")
```

37. Check the Number of Occurrences of a Character in the String

```
# Prompt user for a string and a character
input_string = input("Enter a string: ")
target_char = input("Enter the character to count: ")

# Check if input is valid
if len(input_string) > 0 and len(target_char) == 1:
    # Count occurrences of the target character
    count = input_string.count(target_char)

    # Display the result
    print(f"Number of occurrences of '{target_char}' in '{input_string}':
{count}")
else:
    print("Please enter a valid string and a single character.")
```

38. Convert the First Letter of a String into UpperCase

```
# Prompt user for a string
input_string = input("Enter a string: ")

# Check if input is a valid string
if len(input_string) > 0:
    # Capitalize the first letter
    result_string = input_string.capitalize()

    # Display the result
    print(f"Original String: {input_string}")
    print(f"String with First Letter Uppercase: {result_string}")
else:
    print("Please enter a valid string.")
```

39. Count the Number of Vowels in a String

```
# Function to count vowels in a string
def count_vowels(string):
    vowels = "aeiouAEIOU"
    vowel_count = 0

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

    return vowel_count

# Prompt user for a string
input_string = input("Enter a string: ")

# Check if input is a valid string
if len(input_string) > 0:
    # Call the function and display the result
    number_of_vowels = count_vowels(input_string)
    print(f"Number of vowels in '{input_string}': {number_of_vowels}")
else:
    print("Please enter a valid string.")
```

40. Check Whether a String Starts and Ends With Certain Characters

```
# Prompt user for a string
input_string = input("Enter a string: ")

# Check if input is a valid string
if len(input_string) > 0:
    # Prompt user for start and end characters
    start_char = input("Enter the starting characters: ")
    end_char = input("Enter the ending characters: ")

    # Check if the string starts and ends with the specified characters
    if input_string.startswith(start_char) and input_string.endswith(end_char):
        print(f"The string '{input_string}' starts with '{start_char}' and ends
with '{end_char}'.")
    else:
        print(f"The string '{input_string}' does not start with '{start_char}'
or does not end with '{end_char}'.")
else:
    print("Please enter a valid string.")
```

41. Replace All Occurrences of a String

```
# Example string
original_string = "Hello world, world!"

# String to replace
search_string = "world"

# Replacement string
replacement_string = "universe"

# Replace all occurrences using replace() with a global regex
modified_string = original_string.replace(search_string, replacement_string)

# Display the result
print(f"Original String: {original_string}")
print(f"Modified String: {modified_string}")
```

42. Create Multiline Strings

```
# Multiline string using triple quotes
multiline_string = '''
    This is a multiline string.
    It spans multiple lines.
    You can include line breaks and indentation easily.
'''

print(multiline_string)
```

43. Format Numbers as Currency Strings

```
import locale

# Set the locale to United States
locale.setlocale(locale.LC_ALL, 'en_US.UTF-8')

# Example number
amount = 1234567.89

# Format as currency string
formatted_amount = locale.currency(amount, grouping=True)

# Display the formatted currency string
print(f"Formatted Amount: {formatted_amount}")
```

44. Generate Random String

```
import random
import string

def generate_random_string(length):
    characters = string.ascii_letters + string.digits
    random_string = ''.join(random.choice(characters) for _ in range(length))
    return random_string

# Generate a random string of length 8
random_string = generate_random_string(8)

# Display the random string
print(f"Random String: {random_string}")
```

45. Check if a String Starts with Another String

```
# Example strings
main_string = "Hello, World!"
search_string = "Hello"

# Check if main_string starts with search_string
starts_with = main_string.startswith(search_string)

# Display the result
print(f"Does the string start with '{search_string}'? {starts_with}")
```

46. Trim a String

```
# Example string with leading and trailing whitespaces
string_with_spaces = "   Hello, World!   "

# Trim the string
trimmed_string = string_with_spaces.strip()

# Display the result
print(f"Original String: '{string_with_spaces}'")
print(f"Trimmed String: '{trimmed_string}'")
```

47. Check Whether a String Contains a Substring

```
# Example string
main_string = "Hello, World!"
substring_to_check = "World"

# Check if main_string contains substring_to_check
contains_substring = substring_to_check in main_string

# Display the result
print(f"Does the string contain '{substring_to_check}'? {contains_substring}")
```

48. Compare Two Strings

```
# Example strings
string1 = "Hello"
string2 = "hello"

# Case-sensitive comparison
case_sensitive_comparison = string1 == string2

# Case-insensitive comparison
case_insensitive_comparison = string1.lower() == string2.lower()

# Display the results
print(f"Case-sensitive comparison: {case_sensitive_comparison}")
print(f"Case-insensitive comparison: {case_insensitive_comparison}")
```

49. Encode a String to Base64

```
import base64

# Original string
original_string = "Hello, 你好!"

# Encode the string to Base64
base64_encoded_string = base64.b64encode(original_string.encode('utf-
8')).decode('utf-8')

# Display the result
print(f"Original String: {original_string}")
print(f"Base64 Encoded String: {base64_encoded_string}")
```

50. Replace all Instances of a Character in a String

```
# Example string
original_string = "Hello, World!"

# Character to replace
char_to_replace = "l"

# Replacement character
replacement_char = "x"

# Replace all instances of char_to_replace with replacement_char
modified_string = original_string.replace(char_to_replace, replacement_char)

# Display the result
print(f"Original String: {original_string}")
print(f"Modified String: {modified_string}")
```

51. Replace All Line Breaks with

```
# Example string with line breaks
string_with_line_breaks = "Hello,\nWorld!\nThis is a new line."

# Replacement string or character
replacement_string = "-"

# Replace all line breaks with the replacement string
string_without_line_breaks = string_with_line_breaks.replace("\n",
replacement_string)

# Display the result
print("Original String:")
print(string_with_line_breaks)
print("\nString without Line Breaks:")
print(string_without_line_breaks)
```

52. Check Leap Year

```
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

# Example usage
year_to_check = 2024
if is_leap_year(year_to_check):
    print(f"{year_to_check} is a leap year.")
else:
    print(f"{year_to_check} is not a leap year.")
```

53. Format the Date

```
from datetime import datetime

# Get the current date
current_date = datetime.utcnow()

# Format the date
formatted_date = current_date.strftime("%A, %B %d, %Y")

# Display the result
print(f"Formatted Date: {formatted_date}")
```

54. Display Current Date

```
from datetime import datetime

# Get the current date
current_date = datetime.now()

# Format the current date as a string
formatted_date = current_date.strftime("%m/%d/%Y")  # Adjust the format as
needed

# Display the result
print(f"Current Date: {formatted_date}")
```

55. Compare The Value of Two Dates

```
from datetime import datetime

# Example dates
date1 = datetime.strptime('2022-01-01', '%Y-%m-%d')
date2 = datetime.strptime('2023-01-01', '%Y-%m-%d')

# Compare dates
if date1 < date2:
    print(f"{date1} is earlier than {date2}")
elif date1 > date2:
    print(f"{date1} is later than {date2}")
else:
    print(f"{date1} is equal to {date2}")
```

56. Create Countdown Timer

```
import datetime
import time

# Function to update the countdown
def update_countdown(target_date):
    while True:
        # Get the current date and time
        current_datetime = datetime.datetime.now()

        # Calculate the time difference
        time_difference = target_date - current_datetime

        # Check if the countdown has reached zero
        if time_difference.total_seconds() <= 0:
            print("Countdown expired!")
            break

        # Format and display the countdown
        days = time_difference.days
        hours, remainder = divmod(time_difference.seconds, 3600)
        minutes, seconds = divmod(remainder, 60)
        countdown_str = f"{days}d {hours}h {minutes}m {seconds}s"
        print("Countdown:", countdown_str)

        # Wait for one second
        time.sleep(1)

# Set the target date for the countdown
target_date = datetime.datetime(2023, 1, 1)

# Start the countdown
update_countdown(target_date)
```

57. Remove Specific Item From an Array

```
# Example list
original_list = [1, 2, 3, 4, 5]
item_to_remove = 3

# Find the index of the item to remove
try:
    index_to_remove = original_list.index(item_to_remove)
    # Remove the item from the list
    original_list.pop(index_to_remove)
    print("Original List:", original_list)
except ValueError:
    print("Item not found in the list.")
```

58. Check if An Array Contains a Specified Value

```
# Example list
my_list = [1, 2, 3, 4, 5]
value_to_check = 3

# Check if the list includes the specified value
contains_value = value_to_check in my_list

# Display the result
print(f"Does the list include {value_to_check}? {contains_value}")
```

59. Insert Item in an Array

```
# Example list
my_list = [1, 2, 3, 4, 5]
item_to_insert = 6

# Insert the item at the end of the list
my_list.append(item_to_insert)

# Display the result
print("List after inserting:", my_list)
```

60. Get Random Item From an Array

```
import random

# Example list
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Get a random item from the list
random_item = random.choice(my_list)

# Display the result
print("Random Item:", random_item)
```

61. Perform Intersection Between Two Arrays

```
# Example lists
list1 = [1, 2, 3, 4, 5]
list2 = [3, 4, 5, 6, 7]

# Find the intersection
intersection = [value for value in list1 if value in list2]

# Display the result
print("Intersection:", intersection)
```

62. Split Array into Smaller Chunks

```
def chunk_array(array, chunk_size):
    result = []
    for i in range(0, len(array), chunk_size):
        result.append(array[i:i+chunk_size])
    return result

# Example array
my_array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Split the array into chunks of size 3
chunks = chunk_array(my_array, 3)

# Display the result
print("Original Array:", my_array)
print("Chunks:", chunks)
```

63. Get File Extension

```
def get_file_extension(file_name):
    # Split the file name based on the dot
    parts = file_name.split('.')

    # Get the last part of the array (the file extension)
    extension = parts[-1]

    return extension

# Example usage
file_name = "example.txt"
file_extension = get_file_extension(file_name)

print(f"File Extension: {file_extension}")
```

64. Check If a Variable Is undefined or null

```
# Example variable
variable = None

# Check if the variable is None
if variable is None:
    print("Variable is None")
else:
    print("Variable has a value")
```

65. Generate a Random Number Between Two Numbers

```
import random

def get_random_number(min_val, max_val):
    # Generate a random number in the range [min_val, max_val]
    return random.randint(min_val, max_val)

# Example usage: Generate a random number between 1 and 100
random_num = get_random_number(1, 100)
print("Random Number:", random_num)
```

66. Get The Current URL

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/')
def get_url_details():
    # Get the domain
    domain = request.host

    # Get the path
    path = request.path

    # Get the query parameters
    query_params = request.query_string.decode()

    # Display the results
    print("Domain:", domain)
    print("Path:", path)
    print("Query Parameters:", query_params)
    return "Check the console for details."

if __name__ == '__main__':
    app.run()
```

67. Validate An Email Address

```python
import re

def validate_email(email):
    # Regular expression for a basic email validation
    email_regex = r'^[^\s@]+@[^\s@]+\.[^\s@]+$'

    # Test the email against the regular expression
    return re.match(email_regex, email) is not None

# Example usage:
email_to_validate = "example@email.com"
if validate_email(email_to_validate):
    print("Email is valid")
else:
    print("Email is not valid")
```

68. Check If a Variable is of Function Type

```
# Example variable
def my_function():
    print("Hello, world!")

# Check if the variable is a function
if callable(my_function):
    print("The variable is of function type")
else:
    print("The variable is not of function type")
```

69. Work With Constants

```
# Define a constant (by convention, use all uppercase)
PI = 3.14159

# Attempt to reassign a constant (will not result in an error, but it's against the convention)
# PI = 3.14  # Uncommenting this line will not result in an error, but it's against the convention

# Display the value of the constant
print("The value of PI is:", PI)
```

70. Pass Parameter to a threading.Timer Function

```
import threading

def my_function(parameter):
    print("Parameter received:", parameter)

# Define the parameter
my_parameter = "Hello, world!"

# Define a function to be executed after a delay
def delayed_execution():
    my_function(my_parameter)

# Schedule the function to be executed after a delay
timer = threading.Timer(1.0, delayed_execution)
timer.start()
```

71. Generate a Range of Numbers and Characters

```
def generate_number_range(start, end):
    result = []
    for i in range(start, end + 1):
        result.append(i)
    return result

# Example usage: Generate numbers from 1 to 5
number_range = generate_number_range(1, 5)
print("Number Range:", number_range)
```

72. Perform Function Overloading

```
def example_function(*args):
    if len(args) == 0:
        # No arguments provided
        print("No arguments")
    elif len(args) == 1 and isinstance(args[0], int):
        # One argument of type number provided
        print("One number argument:", args[0])
    elif len(args) == 2 and isinstance(args[0], str) and isinstance(args[1],
int):
        # Two arguments: a string followed by a number
        print("String and number arguments:", args[0], args[1])
    else:
        # Default case
        print("Invalid arguments")

# Example usage
example_function()
example_function(42)
example_function("Hello", 7)
example_function(True, "world")  # Invalid arguments
```

73. Implement a Stack

```
class Stack:
    def __init__(self):
        self.items = []

    # Push an element onto the stack
    def push(self, element):
        self.items.append(element)

    # Pop the top element from the stack
    def pop(self):
        if self.is_empty():
            return "Underflow"
        return self.items.pop()

    # Peek at the top element without removing it
    def peek(self):
        return self.items[-1] if self.items else None

    # Check if the stack is empty
    def is_empty(self):
        return len(self.items) == 0

    # Get the size of the stack
    def size(self):
        return len(self.items)

    # Print the stack elements
    def print(self):
        print(self.items)

# Example usage
stack = Stack()

stack.push(1)
stack.push(2)
```

```
stack.push(3)

print("Stack elements:")
stack.print()  # Outputs: [1, 2, 3]

print("Top element:", stack.peek())  # Outputs: 3

print("Popped element:", stack.pop())  # Outputs: 3

print("Stack size:", stack.size())  # Outputs: 2

print("Is the stack empty?", stack.is_empty())  # Outputs: False
```

74. Implement a Queue

```
class Queue:
    def __init__(self):
        self.items = []

    # Enqueue an element at the end of the queue
    def enqueue(self, element):
        self.items.append(element)

    # Dequeue the element from the front of the queue
    def dequeue(self):
        if self.is_empty():
            return "Underflow"
        return self.items.pop(0)

    # Peek at the front element without removing it
    def front(self):
        if self.is_empty():
            return "Queue is empty"
        return self.items[0]

    # Check if the queue is empty
    def is_empty(self):
        return len(self.items) == 0

    # Get the size of the queue
    def size(self):
        return len(self.items)

    # Print the queue elements
    def print(self):
        print(self.items)

# Example usage
queue = Queue()

queue.enqueue(1)
```

```
queue.enqueue(2)
queue.enqueue(3)

print("Queue elements:")
queue.print()  # Outputs: [1, 2, 3]

print("Front element:", queue.front())  # Outputs: 1

print("Dequeued element:", queue.dequeue())  # Outputs: 1

print("Queue size:", queue.size())  # Outputs: 2

print("Is the queue empty?", queue.is_empty())  # Outputs: False
```

75. Check if a Number is Float or Integer

```
def check_number_type(number):
    # Check if the number has a fractional part
    if isinstance(number, int):
        print(str(number) + " is an integer.")
    elif isinstance(number, float):
        print(str(number) + " is a float.")
    else:
        print(str(number) + " is not a valid number.")

# Example usage:
check_number_type(5)        # Outputs: 5 is an integer.
check_number_type(3.14)     # Outputs: 3.14 is a float.
check_number_type(7.0)      # Outputs: 7.0 is a float.
check_number_type(-2.5)     # Outputs: -2.5 is a float.
check_number_type(float('nan'))  # Outputs: nan is not a valid number.
check_number_type("abc")    # Outputs: abc is not a valid number.
```

76. Pass a Function as Parameter

```
# Function that takes another function as a parameter
def operate_on_numbers(a, b, operation):
    return operation(a, b)

# Example functions to be passed as parameters
def add(x, y):
    return x + y

def multiply(x, y):
    return x * y

# Example usage
result1 = operate_on_numbers(3, 5, add)
print("Result of addition:", result1)  # Outputs: 8

result2 = operate_on_numbers(3, 5, multiply)
print("Result of multiplication:", result2)  # Outputs: 15
```

77. Get the Dimensions of an Image

```
from PIL import Image
import requests
from io import BytesIO

def get_image_dimensions(image_url):
    try:
        # Make a request to get the image content
        response = requests.get(image_url)
        # Open the image using PIL
        img = Image.open(BytesIO(response.content))
        # Get the dimensions
        width, height = img.size
        # Display the dimensions
        print("Width:", width)
        print("Height:", height)
    except Exception as e:
        print("Error loading the image:", e)

# Example usage
image_url = "path/to/your/image.jpg"
get_image_dimensions(image_url)
```

78. Remove All Whitespaces From a Text

```
import re

def remove_whitespaces(input_text):
    # Use a regular expression to replace all whitespaces with an empty string
    return re.sub(r'\s', '', input_text)

# Example usage
text_with_whitespaces = "This is a text with spaces"
text_without_whitespaces = remove_whitespaces(text_with_whitespaces)

print("Original Text:", text_with_whitespaces)
print("Text without Whitespaces:", text_without_whitespaces)
```

79. Write to Console

```
# Write a message to the console
print("Hello, world!")

# You can also print variables or expressions
number = 42
print("The answer is:", number)

# Multiple values can be printed in a single statement
first_name = "John"
last_name = "Doe"
print("Full Name:", first_name, last_name)
```

80. Convert Date to Number

```
from datetime import datetime

# Create a datetime object representing the current date and time
current_date = datetime.now()

# Get the numeric representation of the date (milliseconds since the Unix
Epoch)
numeric_date = int(current_date.timestamp() * 1000)

print("Current Date:", current_date)
print("Numeric Representation:", numeric_date)
```

81. Find the Average of Two Numbers

```
def find_average(num1, num2):
    # Calculate the sum of the two numbers
    total = num1 + num2

    # Calculate the average by dividing the sum by 2
    average = total / 2

    return average

# Example usage
number1 = 10
number2 = 20

result = find_average(number1, number2)

print(f"The average of {number1} and {number2} is: {result}")
```

82. Calculate the Area of a Circle

```python
import math

def calculate_circle_area(radius):
    # Check if the radius is a valid number
    if not isinstance(radius, (int, float)) or radius <= 0:
        return "Invalid radius. Please provide a positive number."

    # Calculate the area
    area = math.pi * radius ** 2

    return area

# Example usage:
radius = 5
area = calculate_circle_area(radius)

print(f"The area of a circle with radius {radius} is: {area}")
```

83. Random Color Generator

```
import random

def generate_random_color():
    # Generate random values for red, green, and blue components
    red = random.randint(0, 255)
    green = random.randint(0, 255)
    blue = random.randint(0, 255)

    # Create the RGB color string
    color = f"rgb({red}, {green}, {blue})"

    return color

# Example usage:
random_color = generate_random_color()

print("Random Color:", random_color)
```

84. Check if a String is Empty

```
def is_empty_string(string):
    return len(string) == 0

# Example usage:
empty_string = ""
non_empty_string = "Hello, world!"

print("Is empty_string empty?", is_empty_string(empty_string))  # Outputs: True
print("Is non_empty_string empty?", is_empty_string(non_empty_string))  #
Outputs: False
```

85. Capitalize the First Letter of a String

```
def capitalize_first_letter(string):
    # Check if the string is not empty
    if len(string) == 0:
        return "Empty string"

    # Capitalize the first letter and concatenate the rest of the string
    return string[0].upper() + string[1:]

# Example usage:
original_string = "hello, world!"
capitalized_string = capitalize_first_letter(original_string)

print("Original String:", original_string)
print("Capitalized String:", capitalized_string)
```

86. Find the Maximum Element in an Array

```
def find_max_element(arr):
    # Check if the array is not empty
    if len(arr) == 0:
        return "Empty array"

    # Use the max() function to find the maximum element
    max_element = max(arr)

    return max_element

# Example usage:
numbers = [5, 2, 9, 1, 7]
max_number = find_max_element(numbers)

print("Array:", numbers)
print("Maximum Element:", max_number)
```