```
Start coding or generate with AI.
```

```
def max_of_three(a, b, c):
  """Finds the maximum of three numbers.

  Args:
    a: The first number.
    b: The second number.
    c: The third number.

  Returns:
    The maximum of the three numbers.
  """
  return max(a, b, c)

# Example usage
result = max_of_three(1, 5, 3)
print(result)  # Output: 5
```

```
def sum_of_list(numbers):
  """Calculates the sum of all numbers in a list.

  Args:
    numbers: A list of numbers.

  Returns:
    The sum of the numbers in the list.
  """
  total = 0
  for number in numbers:
    total += number
  return total

# Example usage
numbers = [8, 2, 3, 0, 7]
result = sum_of_list(numbers)
print(result)  # Output: 20
```

```
def multiply_list(numbers):
  """Multiplies all numbers in a list.

  Args:
    numbers: A list of numbers.

  Returns:
    The product of the numbers in the list.
  """
  result = 1
  for number in numbers:
    result *= number
  return result

# Example usage
numbers = [8, 2, 3, -1, 7]
result = multiply_list(numbers)
print(result)  # Output: -336
```

```
def reverse_string(string):
  """Reverses a string.

  Args:
    string: The string to reverse.

  Returns:
    The reversed string.
  """
  return string[::-1]

# Example usage
string = "1234abcd"
result = reverse_string(string)
print(result)  # Output: dcba4321
```

```
def factorial(n):
  """Calculates the factorial of a non-negative integer.

  Args:
```

```
    n: The non-negative integer.

  Returns:
    The factorial of n.
  """
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)

# Example usage
result = factorial(5)
print(result)  # Output: 120


def is_in_range(number, start, end):
  """Checks if a number is within a given range.

  Args:
    number: The number to check.
    start: The start of the range.
    end: The end of the range.

  Returns:
    True if the number is within the range, False otherwise.
  """
  return start <= number <= end

# Example usage
result = is_in_range(5, 1, 10)
print(result)  # Output: True


def count_case(string):
  """Counts the number of upper and lower case letters in a string.

  Args:
    string: The string to analyze.

  Returns:
    A tuple containing the number of upper case letters and the number of lower
    case letters.
  """
  upper_count = 0
  lower_count = 0
  for char in string:
    if char.isupper():
      upper_count += 1
    elif char.islower():
      lower_count += 1
  return upper_count, lower_count

# Example usage
string = 'The quick Brow Fox'
upper_count, lower_count = count_case(string)
print("No. of Upper case characters :", upper_count)  # Output: 3
print("No. of Lower case Characters :", lower_count)  # Output: 12


def get_distinct_elements(lst):
  """Returns a new list with distinct elements from the first list.

  Args:
    lst: The original list.

  Returns:
    A new list with distinct elements.
  """
  return list(set(lst))

# Example usage
lst = [1, 2, 3, 3, 3, 3, 4, 5]
result = get_distinct_elements(lst)
print(result)  # Output: [1, 2, 3, 4, 5]


def is_prime(n):
  """Checks if a number is prime.

  Args:
    n: The number to check.

  Returns:
```

```
    True if the number is prime, False otherwise.
  """
  if n <= 1:
    return False
  for i in range(2, int(n**0.5) + 1):
    if n % i == 0:
      return False
  return True

# Example usage
result = is_prime(7)
print(result)  # Output: True


def print_even_numbers(numbers):
  """Prints the even numbers from a given list.

  Args:
    numbers: The list of numbers.
  """
  even_numbers = [number for number in numbers if number % 2 == 0]
  print(even_numbers)

# Example usage
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print_even_numbers(numbers)  # Output: [2, 4, 6, 8]


def is_perfect_number(number):
  """Checks if a number is a perfect number.

  Args:
    number: The number to check.

  Returns:
    True if the number is perfect, False otherwise.
  """
  if number <= 1:
    return False

  total = 0
  for i in range(1, number):
    if number % i == 0:
      total += i

  return total == number

# Example usage
number = 6
result = is_perfect_number(number)
print(f"Is {number} a perfect number? {result}")  # Output: True


def is_palindrome(string):
  """Checks if a string is a palindrome.

  Args:
    string: The string to check.

  Returns:
    True if the string is a palindrome, False otherwise.
  """
  string = string.lower()  # Ignore case
  return string == string[::-1]

# Example usage
string = "madam"
result = is_palindrome(string)
print(f"Is '{string}' a palindrome? {result}")  # Output: True


def find_numbers():
  """Finds numbers divisible by 7 but not a multiple of 5 between 2000 and 3200.

  Returns:
    A comma-separated sequence of numbers as a string.
  """
  numbers = []
  for i in range(2000, 3201):
    if (i % 7 == 0) and (i % 5 != 0):
      numbers.append(str(i))
  return ",".join(numbers)
```

```
# Example usage
result = find_numbers()
print(result)  # Output: 2002,2009,2016,2023,...


import math

def calculate_p(c_values):
  """Calculates P using the given formula for a sequence of C values.

  Args:
    c_values: A comma-separated sequence of C values as a string.

  Returns:
    A comma-separated sequence of calculated P values as a string.
  """
  A = 50
  B = 30
  results = []
  c_values = c_values.split(",")  # Split the input string into a list of C values

  for c in c_values:
    c = float(c)  # Convert C to a float
    p = math.sqrt((2 * A * B) / c)
    results.append(str(round(p)))  # Round P to the nearest integer and convert to string

  return ",".join(results)

# Example usage
c_values = "100,150,180"
result = calculate_p(c_values)
print(result)  # Output: 18,22,24


def sort_words(words):
  """Sorts a comma-separated sequence of words alphabetically.

  Args:
    words: A comma-separated sequence of words as a string.

  Returns:
    A comma-separated sequence of sorted words as a string.
  """
  words_list = words.split(",")
  words_list.sort()
  return ",".join(words_list)

# Example usage
words = "without,hello,bag,world"
result = sort_words(words)
print(result)  # Output: bag,hello,without,world


def capitalize_lines(lines):
  """Capitalizes all characters in a sequence of lines.

  Args:
    lines: A sequence of lines as a string, where lines are separated by newlines.

  Returns:
    The capitalized lines as a string.
  """
  capitalized_lines = lines.upper()
  return capitalized_lines

# Example usage
lines = "Hello world\nPractice makes perfect"
result = capitalize_lines(lines)
print(result)  # Output: HELLO WORLD\nPRACTICE MAKES PERFECT


def count_vowels(sentence):
  """Counts the number of vowels in a sentence.

  Args:
    sentence: The input sentence as a string.

  Returns:
    None. Prints the vowel counts to the console.
  """
  vowels = "aeiou"
  vowel_counts = {}

  for vowel in vowels:
```

```
      vowel_counts[vowel] = sentence.lower().count(vowel)

  for vowel, count in vowel_counts.items():
    print(f"{vowel} appeared {count} times")

# Example usage
sentence = "Hello world\nPractice makes perfect"
count_vowels(sentence)  # Output: a appeared 2 times, e appeared 5 times, etc.


def find_even_digit_numbers():
  """Finds numbers from 1000 to 3000 with all even digits.

  Returns:
    A list of numbers with all even digits.
  """
  numbers = []
  for i in range(1000, 3001):
    if all(int(digit) % 2 == 0 for digit in str(i)):
      numbers.append(i)
  return numbers

# Example usage
result = find_even_digit_numbers()
print(result)  # Output: [2000, 2002, 2004, ...]


def find_divisible_by_5(binary_numbers):
  """Finds 4-digit binary numbers divisible by 5 from a comma-separated sequence.

  Args:
    binary_numbers: A comma-separated sequence of 4-digit binary numbers as a string.

  Returns:
    A comma-separated sequence of binary numbers divisible by 5 as a string.
  """
  divisible_numbers = []
  binary_numbers_list = binary_numbers.split(",")

  for binary_number in binary_numbers_list:
    decimal_number = int(binary_number, 2)  # Convert binary to decimal
    if decimal_number % 5 == 0:
      divisible_numbers.append(binary_number)

  return ",".join(divisible_numbers)

# Example usage
binary_numbers = "0100,0011,1010,1001"
result = find_divisible_by_5(binary_numbers)
print(result)  # Output: 1010


def count_letters_digits(sentence):
  """Counts the number of letters and digits in a sentence.

  Args:
    sentence: The input sentence as a string.

  Returns:
    None. Prints the letter and digit counts to the console.
  """
  letter_count = 0
  digit_count = 0

  for char in sentence:
    if char.isalpha():
      letter_count += 1
    elif char.isdigit():
      digit_count += 1

  print("LETTERS", letter_count)
  print("DIGITS", digit_count)

# Example usage
sentence = "hello world! 123"
count_letters_digits(sentence)  # Output: LETTERS 10, DIGITS 3
```