```
def remove_negative_values(input_list):
  return [x for x in input_list if x >= 0]

original_list = [1, -2, 3, -4, 0, 5, -6]
print(f"Original list: {original_list}")

filtered_list = remove_negative_values(original_list)
print(f"List after filtering negative values: {filtered_list}")
```

```
Original list: [1, -2, 3, -4, 0, 5, -6]
List after filtering negative values: [1, 3, 0, 5]
```

Start coding or generate with AI.

```
def count_chars(input_string):
  """
  Counts the number of vowels, consonants, and digits in a string.

  Args:
    input_string: The string to analyze.

  Returns:
    A tuple containing (vowel_count, consonant_count, digit_count).
  """
  vowels = "aeiouAEIOU"
  vowel_count = 0
  consonant_count = 0
  digit_count = 0

  for char in input_string:
    if char.isalpha(): # Check if the character is an alphabet
      if char in vowels:
        vowel_count += 1
      else:
        consonant_count += 1
    elif char.isdigit(): # Check if the character is a digit
      digit_count += 1

  return vowel_count, consonant_count, digit_count
```

```
# Example usage:
my_string = "Hello World! 123"
v, c, d = count_chars(my_string)

print(f"Original string: '{my_string}'")
print(f"Number of vowels: {v}")
print(f"Number of consonants: {c}")
print(f"Number of digits: {d}")
```

```
Original string: 'Hello World! 123'
Number of vowels: 3
Number of consonants: 7
Number of digits: 3
```

Start coding or generate with AI.

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

  A palindrome is a word, phrase, number, or other sequence of characters
  which reads the same backward as forward, ignoring punctuation, case, and spacing.

  Args:
    input_string: The string to check.

  Returns:
    True if the string is a palindrome, False otherwise.
  """
  # Convert to lowercase and remove non-alphanumeric characters
  cleaned_string = ''.join(char.lower() for char in input_string if char.isalnum())
```

```
    # Compare the cleaned string with its reverse
    return cleaned_string == cleaned_string[::-1]
```

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

```
# Example usage:
print(f"'madam' is a palindrome: {is_palindrome('madam')}")
print(f"'racecar' is a palindrome: {is_palindrome('racecar')}")
print(f"'A man, a plan, a canal: Panama' is a palindrome: {is_palindrome('A man, a plan, a canal: Panama')}")
print(f"'hello' is a palindrome: {is_palindrome('hello')}")
print(f"'Was it a car or a cat I saw?' is a palindrome: {is_palindrome('Was it a car or a cat I saw?')}")
```

```
'madam' is a palindrome: True
'racecar' is a palindrome: True
'A man, a plan, a canal: Panama' is a palindrome: True
'hello' is a palindrome: False
'Was it a car or a cat I saw?' is a palindrome: True
```

```
def is_palindrome(text):
    text = text.lower()
    return text == text[::-1]
```

is_palindrome function line by line:

def is_palindrome(text):

This line defines a Python function named is_palindrome that accepts one argument: text (which is expected to be a string). text = text.lower()

This line takes the input_string (passed as text) and converts all its characters to lowercase using the .lower() string method. It then reassigns this entirely lowercase string back to the text variable. This step ensures that the palindrome check is case-insensitive (e.g., 'Racecar' is treated the same as 'racecar'). return text == text[::-1]

This line performs the core palindrome check and returns the boolean result: text[::-1]: This is Python's slice notation used to reverse a string. [::-1] creates a reversed copy of the text string. text == ...: It compares the lowercase text string with its reversed version. If they are identical, the expression evaluates to True, indicating it's a palindrome. Otherwise, it evaluates to False. return: The boolean result (True or False) of this comparison is returned by the function.