Problem #1

Subject: Formatting a large integer with arbitrary separators.
Modules: none
Filename: pretty_number.py

WORK

The goal is to modify the existing pretty_int() function to support customizable separators and group sizes. The function originally formats positive integers using commas as separators and groups digits by thousands. However, the modification allows users to specify their desired separator character and group size through keyword arguments sep and group, with defaults set to , and 3, respectively. If the group size is 0 or negative, no separators are used. Additionally, non-integer group sizes are rounded to the nearest integer. The test code in the module ensures the function works correctly by executing various test cases and printing the input values and corresponding results. Each test case includes different combinations of input values to thoroughly test the function's functionality and versatility. The test report presents the inputs and outputs in a readable format, aiding in verifying the correctness of the pretty_int() function and its modifications. Finally, the main code calls a separate test function to keep the test code organized and manageable.

OUTPUT

```
PRETTY INT FUNCTION

Input: 320875, Separator: ';', Group: 5
Output:  3;20875

Input: 9762, Separator: '-', Group: 2
Output:  97-62

Input: 54123, Separator: ',', Group: 3
Output:  54,123

Input: 8825614, Separator: '.', Group: 4
Output:  882.5614

Input: 379041, Separator: '-', Group: 3
Output:  379-041

Input: 86542, Separator: ',', Group: 2
Output:  8,65,42

Input: 1854320, Separator: '.', Group: 6
Output:  1.854320

Input: 70819, Separator: ';', Group: 4
Output:  7;0819

Input: 963274, Separator: '-', Group: 3
Output:  963-274

Input: 5429, Separator: ',', Group: 5
Output:  5429
```

CODE

```
'''
PROGRAMMER: Jakob K. West
USERNAME: jwest21
PROGRAM: pretty_number.py

DESCRIPTION: Module to neatly format numbers using various user-created functions
'''


"""""""""""""""""""""
PRETTY INT FUNCTION
"""""""""""""""""""""

def pretty_int(n, sep=',', group=3):
    """
    Convert a non-negative integer to a string with arbitrary separators.

    Parameters:
    n (int): A non-negative integer.
    sep (str): The separator character or string. Default is ','.
    group (int): The number of digits in each group. Default is 3.

    Returns:
    str: A string representing the number with specified separators and grouping.
    """


    # Round group to the nearest integer
    group = int(round(group))

    # Handle case when group size is 0 or less
    if group <= 0:
        return str(n)

    # Convert integer to string
    num_str = str(n)

    # Insert separators
    result = ''
    for i in range(len(num_str)):
        if i > 0 and (len(num_str) - i) % group == 0:
            result += sep
        result += num_str[i]

    return result


def test_pretty_int():
    """
    Test function for pretty_int().
    """
    test_cases = [
        (320875, ';', 5),          # Test case with a large positive number and a
semicolon separator with a group size of 5
        (9762, '-', 2),            # Test case with a small positive number and a
hyphen separator with a group size of 2
        (54123, ',', 3),           # Test case with a medium positive number and a
comma separator with a group size of 3
        (8825614, '.', 4),         # Test case with a large positive number and a
period separator with a group size of 4
        (379041, '-', 3),          # Test case with a medium positive number and a
hyphen separator with a group size of 3
        (86542, ',', 2),           # Test case with a medium positive number and a
comma separator with a group size of 2
        (1854320, '.', 6),         # Test case with a large positive number and a
period separator with a group size of 6
        (70819, ';', 4),           # Test case with a medium positive number and a
semicolon separator with a group size of 4
        (963274, '-', 3),          # Test case with a large positive number and a
hyphen separator with a group size of 3
        (5429, ',', 5)             # Test case with a small positive number and a
comma separator with a group size of 5
    ]


    print("\nPRETTY INT FUNCTION")
    print()

    for n, sep, group in test_cases:
        print(f"Input: {n}, Separator: '{sep}', Group: {group}")
        try:
            result = pretty_int(n, sep, group)
            print("Output: ", result)
        except Exception as e:
            print("Error: ", e)
        print()
```

Problem #2

Subject: Formatting a floating-point value with arbitrary separators.
Modules: none
Filename: pretty_number.py

WORK

The module **pretty_number.py** introduces a new function **pretty_num()** that behaves similarly to **pretty_int()** but supports negative values and floating-point numbers. Floating-point values are represented with six digits after the decimal point, which can be adjusted using the **places** keyword argument. If **places** is negative, rounding occurs to the left of the decimal point. Additionally, users can specify an alternate decimal point character (radix mark) using the **mark** keyword argument. Similar to Problem #1, appropriate test code is written to validate the functionality of the **pretty_num()** function, covering various scenarios and input combinations to ensure robustness and accuracy.

OUTPUT

```
PRETTY NUM FUNCTION

Input: 17348.235, Separator: ',', Group: 3, Places: 6, Mark: '.'
Output: 17,348.235

Input: -9075.7689, Separator: ';', Group: 4, Places: 3, Mark: ','
Output: -9075,768

Input: 1352.47, Separator: '.', Group: 2, Places: 4, Mark: '.'
Output: 13.52.47

Input: 450987.13572, Separator: '-', Group: 5, Places: 5, Mark: ':'
Output: 4-50987:13572

Input: -36842.689, Separator: ',', Group: 2, Places: 7, Mark: ','
Output: -3,68,42,689

Input: 123456.789, Separator: ';', Group: 3, Places: 2, Mark: ';'
Output: 123;456;78

Input: 99999, Separator: '.', Group: 0, Places: 3, Mark: '.'
Output: 99999

Input: -753.91, Separator: '-', Group: 2, Places: 5, Mark: ','
Output: -7-53,91

Input: 5287.6301, Separator: ',', Group: -3, Places: 4, Mark: '.'
Output: 5287.6301

Input: 825731.45, Separator: '-', Group: 7, Places: 2, Mark: ':'
Output: 825731:45
```

CODE

```
"""""""""""""""""""""
PRETTY NUM FUNCTION
"""""""""""""""""""""

def pretty_num(n, sep=',', group=3, places=6, mark='.'):

    """
    Convert a number to a string with arbitrary separators and decimal formatting.

    Parameters:
    n (float or int): The number to format.
    sep (str): The separator character or string. Default is ','.
    group (int): The number of digits in each roup Default is 3.
    places (int): The number of decimal places. Default is 6.
    mark (str): The character used as the decimal point. Default is '.'.

    Returns:
    str: A string representing the number with specified separators and decimal
formatting.
    """

    # Round group to the nearest integer
    group = int(round(group))

    # Handle case when group size is 0 or less
    if group <= 0:
        return str(n)

    # Handle negative numbers
    if n < 0:
        sign = '-'
        n = abs(n)
    else:
        sign = ''

    # Convert to string and split integer and decimal parts
    num_str = str(n)
    integer_part, _, decimal_part = num_str.partition('.')

    # Insert separators for integer part
    result = ''
    for i in range(len(integer_part)):
        if i > 0 and (len(integer_part) - i) % group == 0:
            result += sep
        result += integer_part[i]

    # Add decimal part with specified number of places
    result += mark + decimal_part[:places]

    return sign + result


def test_pretty_num():
    """
    Test function for pretty_num().
    """

    test_cases = [
        (17348.235, ',', 3, 6, '.'),    # Random positive number with separators and
decimal places
        (-9075.7689, ';', 4, 3, ','),   # Random negative number with different
separators and decimal places
        (1352.47, '.', 2, 4, '.'),      # Random positive number with decimal point
and specified group size
        (450987.13572, '-', 5, 5, ':'), # Random positive number with separators and
decimal places
        (-36842.689, ',', 2, 7, ','),   # Random negative number with different
separators and decimal places
        (123456.789, ';', 3, 2, ';'),   # Random positive number with different
separators and decimal places
        (99999, '.', 0, 3, '.'),        # Random positive number with zero decimal
places
        (-753.91, '-', 2, 5, ','),      # Random negative number with different
separators and decimal places
        (5287.6301, ',', -3, 4, '.'),   # Random positive number with negative group
size and decimal places
        (825731.45, '-', 7, 2, ':')     # Random positive number with separators and
decimal places
    ]



    print("\nPRETTY NUM FUNCTION")
    print()

    for n, sep, group, places, mark in test_cases:
        print(f"Input: {n}, Separator: '{sep}', Group: {group}, Places: {places},
Mark: '{mark}'")
        try:
            result = pretty_num(n, sep, group, places, mark)
            print("Output:", result)
        except Exception as e:
            print("Error:", e)
        print()
```

Problem #3

Subject: Formatting a value with significant figures.
Modules: none
Filename: pretty_number.py

WORK

In enhancing the **pretty_sf()** function, the focus is on supporting the specification of significant figures using the **sigfigs** keyword. Rules dictate that if both **places** and **sigfigs** are provided, **places** is ignored. Determining significant figures involves considering the first non-zero or non-one digit as the start, with trailing digits being placeholders. Any significant zeros are always included. Notably, values less than 1 have a leading zero before the decimal point. A leading 1 isn't counted as a significant figure due to precision considerations. Appropriate test code should validate the function's behavior across varied inputs.

OUTPUT

```
PRETTY SF FUNCTION

Input: 0.56342897, Separator: ',', Group: 4, Places: 6, Sigfigs: 3, Mark: '.'
Output: 0.6

Input: -987.2176098, Separator: '-', Group: 3, Places: 4, Sigfigs: 2, Mark: ','
Error: substring not found

Input: 123456.789124, Separator: ';', Group: 5, Places: 7, Sigfigs: 4, Mark: ':'
Error: substring not found

Input: 654.93786, Separator: '.', Group: 0, Places: 3, Sigfigs: 5, Mark: '.'
Error: integer division or modulo by zero

Input: -0.43521678, Separator: '-', Group: 2.5, Places: 5, Sigfigs: 3, Mark: ','
Output: -0,4

Input: 1234567890.123, Separator: ',', Group: -4, Places: 4, Sigfigs: 6, Mark: ';'
Error: substring not found

Input: 0.98654321, Separator: '.', Group: 2, Places: 6, Sigfigs: 4, Mark: '.'
Output: 0.99

Input: -0.12456789, Separator: ';', Group: 5, Places: 6, Sigfigs: 3, Mark: ','
Output: -0,1

Input: 456.79321, Separator: ',', Group: 6, Places: 3, Sigfigs: 2, Mark: '.'
Error: substring not found

Input: -87654321.098, Separator: '-', Group: 4, Places: 4, Sigfigs: 4, Mark: ':'
Error: substring not found
```

CODE

```
"""""""""""""""""""""
PRETTY SF FUNCTION
"""""""""""""""""""""

def pretty_sf(n, sep=',', group=3, places=6, sigfigs=3, mark='.'):
    """
    Convert a number to a string with arbitrary separators and specified
    significant figures

    Parameters:
    n (float or int): The number to format.
    sep (str): The separator character or string. Default is ','.
    group (int): The number of digits in each group. Default is 3.
    places (int): The number of decimal places. Default is 6.
    sigfigs (int): The number of significant figures. Default is 3.
    mark (str): The character used as the decimal point. Default is '.'.

    Returns:
    str: A string representing the number with specifed separators and significant
figures.
    """
    # Check if significant figures are specified
    if sigfigs is not None:
        # Calculate the number of decimal places to round to based on significant
figures
        places = max(0, sigfigs - len(str(int(abs(n)))) - (0 if n == 0 else 1))

    # Round the number to the specified decimal places
    rounded_num = round(n, places)

    # Convert the rounded number to a string with the desired format
    formatted_str = '{:.{places}f}'.format(rounded_num, places=max(0, places))

    # Insert separators
    result = ''
    decimal_index = formatted_str.index('.')
    for i in range(len(formatted_str)):
        if formatted_str[i] == '.':
            result += mark

        else:
            result += formatted_str[i]
            if i < decimal_index and (decimal_index - i - 1) % group == 0 and i !=
decimal_index - 1:
                result += sep

    return result



def test_pretty_sf():
    """
    Test function for pretty_sf.
    """
    test_cases = [
        (0.56342897, ',', 4, 6, 3, '.'),       # Random positive number with 
separators and 6 significant figures
        (-987.2176098, '-', 3, 4, 2, ','),     # Random negative number with 
different separator and 4 significant figures
        (123456.789124, ';', 5, 7, 4, ':'),    # Random large positive number with 
different separator and 7 significant figures
        (654.93786, '.', 0, 3, 5, '.'),        # Random positive number with no 
decimal places and 3 significant figures
        (-0.43521678, '-', 2.5, 5, 3, ','),    # Random negative number with non-
integer group size and 5 significant figures
        (1234567890.123, ',', -4, 4, 6, ';'),   # Random large positive number with 
negative group size and different separator
        (0.98654321, '.', 2, 6, 4, '.'),       # Random small positive number with 2 
decimal places and 6 significant figures
        (-0.12456789, ';', 5, 6, 3, ','),      # Random negative number with 
different separator and 6 significant figures
        (456.793210, ',', 6, 3, 2, '.'),       # Random positive number with 
different group size and 3 significant figures
        (-87654321.098, '-', 4, 4, 4, ':')     # Random negative number with 
different separator and 4 significant figures
    ]



    print("\nPRETTY SF FUNCTION")
    print()

    for n, sep, group, places, sigfigs, mark in test_cases:
        print(f"Input: {n}, Separator: '{sep}', Group: {group}, Places: {places}, 
Sigfigs: {sigfigs}, Mark: '{mark}'")
        try:
            result = pretty_sf(n, sep, group, places, sigfigs, mark)
            print("Output:", result)
        except Exception as e:
            print("Error:", e)
        print()
```

Problem #4

Subject: Formatting a value with SI prefixes and units.
Modules: none
Filename: pretty_number.py

WORK

In the final function **pretty_si()**, a boolean keyword argument **si** is introduced, defaulting to False, functioning similarly to **pretty_sf()** when False. When True, the value passed is scaled using standard SI prefixes, ensuring it remains between 1 and 1000. For instance, 0.0103 becomes '10.3 m', while 9999 becomes '9.999 k'. Additionally, a 'units' keyword argument permits specifying base units like 'm' for meters or 'V' for volts. The function accommodates any string as a unit, with the scaling prefix preceding the units if provided. One space separates the value and the (possibly scaled) units. The output string includes a space following the value if **si** is True or **units** are provided; otherwise, no space is present. SI prefixes spanning from queto- (10^-30) to que�a- (10^30) are supported, excluding non-factor-of-1000 prefixes. Exponential notation, divisible by 3, is employed for values ≥ 1000 Q or < 1 q. Test code should be included to ensure functionality across varied inputs.

OUTPUT

```
PRETTY SI FUNCTION

Input: 0.001234, Separator: ',', Group: 3, Sigfigs: 2, Mark: '.', SI: False, Units:
Output: None

Input: -4000000, Separator: '-', Group: 4, Sigfigs: 3, Mark: ',', SI: True, Units: m
Output: -400-0000 M m

Input: 0.000987, Separator: ';', Group: 2, Sigfigs: 1, Mark: ':', SI: False, Units:
Output: None

Input: 543210987, Separator: '.', Group: 0, Sigfigs: 4, Mark: '-', SI: True, Units: A
Output: 543210987

Input: 7654.32, Separator: '-', Group: 2, Sigfigs: 3, Mark: ',', SI: False, Units:
Output: None

Input: -5678.9012, Separator: ',', Group: 3, Sigfigs: 2, Mark: '.', SI: True, Units:
g
Output: -567,8.9 k g

Input: 3210987, Separator: ';', Group: 4, Sigfigs: 1, Mark: ':', SI: False, Units:
Output: None

Input: -0.007654, Separator: '.', Group: 6, Sigfigs: 3, Mark: '-', SI: True, Units:
Hz
Output: -0-0077 m Hz

Input: 98765432, Separator: '-', Group: 5, Sigfigs: 4, Mark: ',', SI: False, Units:
Output: None

Input: 0.002345, Separator: ',', Group: 2, Sigfigs: 2, Mark: '.', SI: True, Units: N
```

Output: 0,.0,02,34 m N

CODE

```
"""""""""""""""""""""
PRETTY SI FUNCTION
"""""""""""""""""""""
def pretty_si(n, sep=',', group=3, sigfigs=3, mark='.', si=False, units=''):
    """
    Convert a number to a string with SI prefix scaling and significant figures.

    Parameters:
    n (float or int): The number to format.
    sep (str): The separator character or string. Default is ','.
    group (int): The number of digits in each group. Default is 3.
    sigfigs (int): The number of significant figures. Default is 3.
    mark (str): The character used as the decimal point. Default is '.'.
    si (bool): If True, scale the value using SI prefixes. Default is False.
    unit (str): The units of the value. Default is ''.

    Returns:
    str: A string representing the number with SI prefix scaling and
    significant figures.
    """


    # Round group to the nearest integer
    group = int(round(group))

    # Handle case when group size is 0 or less
    if group <= 0:
        return str(n)

    # Handle negative numbers
    if n < 0:
        sign = '-'
        n = abs(n)
    else:
        sign = ''

    # Handle special case for zero
    if n == 0:
        return '0'

    # Convert to string
    num_str = str(n)

    # Determine the number of digits before the decimal point
    if '.' in num_str:
        digits_before_decimal = len(num_str.split('.')[0])
    else:
        digits_before_decimal = len(num_str)

    # Determine the number of digits after the decimal point
    if '.' in num_str:
        digits_after_decimal = len(num_str.split('.')[1])
    else:
        digits_after_decimal = 0

    # Calculate the number of significant digits
    if digits_before_decimal > 1:
        sf_count = digits_before_decimal
    else:
        sf_count = digits_before_decimal + digits_after_decimal

    # Determine the number of digits to remove
    remove_count = sf_count - sigfigs

    # Round the number to the specified significant figures
    rounded_num = round(n, remove_count)

    # Convert the rounded number to string
    rounded_str = str(rounded_num)

    # Format the string with separators
    result = ''
    for i in range(len(rounded_str)):
        if rounded_str[i] == '.':
            result += mark
        else:
            result += rounded_str[i]
            if (len(rounded_str) -i - 1) % group == 0 and i != len(rounded_str) - 1:
                result += sep

    # Append SI prefix and units if specified
    if si:
        prefixes = ['q', 'r', 'y', 'z', 'a', 'f', 'p', 'n', 'μ', 'm', '',
                    'k', 'M', 'G', 'T', 'P', 'E', 'Z', 'Y', 'R', 'Q']
        exp_index = 10
        while exp_index < 21 and rounded_num >= 1000:
            rounded_num /= 1000
            exp_index += 1
        while exp_index > 0 and rounded_num < 1:
            rounded_num *= 1000
            exp_index -= 1

        prefix = prefixes[exp_index]
        if prefix:
            result += ' ' + prefix
        if units:
            result += ' ' + units
        elif units:
            result += ' ' + units

        return sign + result


def test_pretty_si():
    """
    Test Function for pretty_si.
    """

    test_cases = [
        (0.001234, ',', 3, 2, '.', False, ''),       # Random small positive number
with no SI prefix and no units
        (-4000000, '-', 4, 3, ',', True, 'm'),       # Random large negative number
with SI prefix 'M' and units 'm'
        (0.000987, ';', 2, 1, ':', False, ''),       # Random small positive number
with no SI prefix and no units
        (543210987, '.', 0, 4, '-', True, 'A'),      # Random large positive number
with SI prefix 'M' and units 'A'
        (7654.32, '-', 2, 3, ',', False, ''),        # Random positive number with no
SI prefix and no units
        (-5678.9012, ',', 3, 2, '.', True, 'g'),     # Random negative number with SI
prefix 'm' and units 'g'
        (3210987, ';', 4, 1, ':', False, ''),        # Random positive number with no
SI prefix and no units
        (-0.007654, '.', 6, 3, '-', True, 'Hz'),     # Random small negative number
with SI prefix 'm' and units 'Hz'
        (98765432, '-', 5, 4, ',', False, ''),       # Random large positive number
with no SI prefix and no units
        (0.002345, ',', 2, 2, '.', True, 'N')        # Random small positive number
with SI prefix 'm' and units 'N'
    ]


    print("\nPRETTY SI FUNCTION")
    print()

    for n, sep, group, sigfigs, mark, si, units in test_cases:
        print(f"Input: {n}, Separator: '{sep}', Group: {group}, Sigfigs: {sigfigs},
Mark: '{mark}', SI: {si}, Units: {units}")
        try:
            result = pretty_si(n, sep, group, sigfigs, mark, si, units)
            print("Output:", result)
        except Exception as e:
            print("Error:", e)
        print()


"""
Main
"""

if __name__ == "__main__":
    test_pretty_int()
    test_pretty_num()
    test_pretty_sf()
    test_pretty_si()
```