```
'''
    listsorting.py -- brief samples of built-in list sorting

    Jeff Ondich, 30 Jan 2008
    Tianna Avery, November 2018, updated for Python 3
'''

# Ordinary list sorting, using the default object ordering.
my_list = [23, 4, -3, 19, 29]
print("Unsorted list:", my_list)
my_list = sorted(my_list)
print("Sorted:", my_list)


# Using the keyword parameter "reverse" with the sort method.
my_list = [23, 4, -3, 19, 29]
my_list = sorted(my_list, reverse=True)
print("Sorted in reverse:", my_list)


# Using a custom comparison function with the sort method.
def backwards_cmp(a, b):
    return (b > a) - (b < a)

# Convert a cmp= function (Python 2) into a key= function (Python 3)
def cmp_to_key(mycmp):
    class K:
        def __init__(self, obj, *args):
            self.obj = obj
        def __lt__(self, other):
            return mycmp(self.obj, other.obj) < 0
        def __gt__(self, other):
            return mycmp(self.obj, other.obj) > 0
        def __eq__(self, other):
            return mycmp(self.obj, other.obj) == 0
        def __le__(self, other):
            return mycmp(self.obj, other.obj) <= 0
        def __ge__(self, other):
            return mycmp(self.obj, other.obj) >= 0
        def __ne__(self, other):
            return mycmp(self.obj, other.obj) != 0
    return K

my_list = [23, 4, -3, 19, 29]
my_list = sorted(my_list, key=cmp_to_key(backwards_cmp))
print("Sorted in reverse using a custom comparison:", my_list)
```