```
'''
    dictionarysorting.py -- examples of sorting dictionaries by key
and value

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


print('============= a dictionary, its items, and its keys
===============')
dictionary = {'goat':5, 'zebu':9, 'kudu':9, 'moose':27, 'aardvark':9}
print("dictionary:", dictionary)
print("dictionary.items():", dictionary.items())
print("dictionary keys():", dictionary.keys())
print()

print('============= a dictionary\'s keys & values, sorted by key
===============')
keys = dictionary.keys()
keys = sorted(keys)
for key in keys:
    print(key, dictionary[key])
print()

print('============= a dictionary\'s keys & values, sorted by value
===============')
print('=============    version 1, using the "sorted" function and
===============')
print('=============    default comparison function for the values
===============')
for key in sorted(dictionary, key=dictionary.get, reverse=True):
    print(key, dictionary[key])
print()

print('============= a dictionary\'s keys & values, sorted by value
===============')
print('=============    version 2, using a global dictionary and a
===============')
print('=============    custom comparison function
===============')

# 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

def value_cmp(a, b):
    value_comparison = (dictionary[b] > dictionary[a])
    value_comparison = value_comparison - (dictionary[b] <
dictionary[a])
    if value_comparison == 0:
        return (a > b) - (a < b)
    else:
        return value_comparison

keys = dictionary.keys()
keys = sorted(keys, key=cmp_to_key(value_cmp))
for key in keys:
    print(key, dictionary[key])
print()


print('============= a dictionary\'s keys & values, sorted by value
===============')
print('=============    version 3, using a locally defined custom
===============')
print('=============    comparison function
===============')

def get_keys_sorted_by_value(dict):
    def valCmp(a, b):
        value_comparison = (dictionary[b] > dictionary[a])
        value_comparison = value_comparison - (dictionary[b] <
dictionary[a])
        if value_comparison == 0:
            return (a > b) - (a < b)
        else:
            return value_comparison

    keys = dict.keys()
    keys = sorted(keys, key=cmp_to_key(valCmp))
    return keys

sorted_keys = get_keys_sorted_by_value(dictionary)
for key in sorted_keys:
```

```
    print(key, dictionary[key])
print()


print('============= a dictionary\'s keys & values, sorted by value
===============')
print('=============    version 3, using list comprehensions
===============')
items = dictionary.items()

items = [(v, k) for (k, v) in items]
items = sorted(items)
items.reverse()
items = [(k, v) for (v, k) in items]

for (k, v) in items:
    print(k, v)
```