# Calculating with Dictionaries

Calculating minimum, maximum, and sorted … on a dictionary data

Consider a dictionary that maps stock names to prices:

prices = {

    'ACME': 45.23,

    'AAPL': 612.78,

    'IBM': 205.55,

    'HPQ': 37.20,

    'FB': 10.75

}

In order to perform useful calculations on the dictionary contents, it is often useful to invert the keys and values of the dictionary using zip(). For example, here is how to find the minimum and maximum price and stock name:

min\_val = min(zip(prices.values(), prices.keys()))

print(min\_val)

# (10.75, 'FB')

max\_val = max(zip(prices.values(), prices.keys()))

print(max\_val)

# (612.78, 'AAPL')

Similarly, to rank the data, use zip() with sorted(), as in the following:

sorted\_dict = sorted(zip(prices.values(), prices.keys()))

print(sorted\_dict)

# [(10.75, 'FB'), (37.2, 'HPQ'), (45.23, 'ACME'),   
# (205.55, 'IBM'), (612.78, 'AAPL')]

# Note:

If you try to perform common data reductions on a dictionary, you’ll find that they only process the keys, not the values. For example:

print(min(prices))  # AAPL

print(max(prices))  # IBM