Dictionary

• A sequence of key-value pairs:
  \{key: value, key: value, ... key: value\}

• Optimized for efficient search based on the key
• May not always print the elements in the same order
  – As you make modifications (insert/delete elements) the order of the pairs may change to enable fast searching
Legal Keys

• No duplicate keys
  – Cannot have multiple entries for a key
• Keys must be immutable types
  – Strings
  – Integers
  – Tuples
• Mix and match is okay
Working with dictionaries

• \{key: value, key: value, \ldots \ key: \ value\}

• What operations would you like? What operations do you know from the other collections?
Working with dictionaries

• They are mutable
• Shallow copies
• \(d = \{\text{key: value, key: value, ... key: value}\}\)
• Functions:
  - \(d[\text{new\_key}] = \text{new\_value}\)
  - \textbf{del} \ d[\text{key}]
  - \textbf{d.pop}(\text{key})
  - \text{d.items()}, \ d.keys(), \ d.values()
    • \text{list}(d.keys()), \ \text{sorted}(d.keys())
  - \textbf{len}(d): \text{number of key-value pairs}
  - \textbf{in}: \text{verifies that a key is in the dictionary}
  - \textbf{min}(\underbrace{\_\_\_}), \ \textbf{max}(\underbrace{\_\_\_})
keys(), values(), items()

• These methods return view objects
• They will reflect changes made to the dictionary
  – If you add or delete items in the dictionary, the views will reflect that

• To get a list of them use the list constructor:
  – list(d.keys()), list(d.values()), list(d.items())
  – How connected are the lists and the views they are based on?
    • d = {1:[1,2]}, lv = list(d.values()), d[1][0] = 10, d, lv, lv[0][1] = 200, lv, d
Examples

• A dictionary does not have an obvious order of the key-value pairs:

```python
>>> d = {'g': 6, 'A':1, 'c':[4,5,6,7], 'x':67}
>>> d
{'g': 6, 'c': [4, 5, 6, 7], 'A': 1, 'x': 67}
```