6 Dictionaries and Sets

Objectives

In this chapter, you'll:

- Use dictionaries to represent unordered collections of key–value pairs.
- Use sets to represent unordered collections of unique values.
- Create, initialize and refer to elements of dictionaries and sets.
- Iterate through a dictionary's keys, values and key–value pairs.
- Add, remove and update a dictionary's key–value pairs.
- Use dictionary and set comparison operators.
- Combine sets with set operators and methods.
- Use operators `in` and `not in` to determine if a dictionary contains a key or a set contains a value.
- Use the mutable set operations to modify a set's contents.
- Use comprehensions to create dictionaries and sets quickly and conveniently.
- Learn how to build dynamic visualizations and implement more of your own in the exercises.
- Enhance your understanding of mutability and immutability.

Exercises

Unless specified otherwise, use IPython sessions for each exercise.

6.1 *(Discussion: Dictionary Methods)* Briefly explain the operation of each of the following dictionary methods:

a) `update`

Answer: Adds a new key–value pair or updates an existing one if the key is already in the dictionary. [**Instructor Note: In the first and second printings, Part (a) specified the method named `add`, which does not exist.**]

b) `keys`

Answer: Returns an object that can be used to iterate over a dictionary's keys.

c) `values`

Answer: Returns an object that can be used to iterate over a dictionary's values.

d) `items`

Answer: Returns an object that can be used to iterate over a dictionary's key–value pairs.

6.2 *(What's Wrong with This Code?)* The following code should display the unique words in the string `text` and the number of occurrences of each word.

```
from collections import Counter
text = ('to be or not to be that is the question')
counter = Counter(text.split())
for word, count in sorted(counter):
    print(f'{word:<12}{count}')
```

Answer: `sorted(counter)` should be `sorted(counter.items())`; otherwise, a `ValueError` occurs because `sorted(counter)` returns a list that has too many values to unpack into `word` and `count`.

6.3 *(What Does This Code Do?)* The dictionary `temperatures` contains three Fahrenheit temperature samples for each of four days. What does the `for` statement do?

```
temperatures = {
    'Monday': [66, 70, 74],
    'Tuesday': [50, 56, 64],
    'Wednesday': [75, 80, 83],
    'Thursday': [67, 74, 81]
}

for k, v in temperatures.items():
    print(f'{k}: {sum(v)/len(v):.2f}')
```

Answer: This code calculates and displays the average temperature for each day:

```
Monday: 70.00
Tuesday: 56.67
Wednesday: 79.33
Thursday: 74.00
```

6.4 *(Fill in the Missing Code)* In each of the following expressions, replace the ***s with a set operator that produces the result shown in the comment. The last operation should check whether the left operand is an improper subset of the right operand. For each of the first four expressions, specify the name of the set operation that produces the specified result.

a) `{1, 2, 4, 8, 16} *** {1, 4, 16, 64, 256}  # {1,2,4,8,16,64,256}`
b) `{1, 2, 4, 8, 16} *** {1, 4, 16, 64, 256}  # {1,4,16}`
c) `{1, 2, 4, 8, 16} *** {1, 4, 16, 64, 256}  # {2,8}`
d) `{1, 2, 4, 8, 16} *** {1, 4, 16, 64, 256}  # {2,8,64,256}`
e) `{1, 2, 4, 8, 16} *** {1, 4, 16, 64, 256}  # False`

Answer:

a) Union:

```
{1, 2, 4, 8, 16} | {1, 4, 16, 64, 256}  # {1,2,4,8,16,64,256}
```

b) Intersection:

```
{1, 2, 4, 8, 16} & {1, 4, 16, 64, 256}  # {1,4,16}
```

c) Difference:

```
{1, 2, 4, 8, 16} - {1, 4, 16, 64, 256}  # {2,8}
```

d) Symmetric difference:

```
{1, 2, 4, 8, 16} ^ {1, 4, 16, 64, 256}  # {2,8,64,256}
```

e) `{1, 2, 4, 8, 16} <= {1, 4, 16, 64, 256}  # False`