Chapter 12

How to work with dictionaries

Applied objectives

1. Use dictionaries in your programs.
2. Use dictionaries that contain complex objects like lists and other dictionaries.

Knowledge objectives

1. Differentiate between a list and a dictionary.
2. Describe the use of these dictionary methods when creating view objects: key(), items(), and values().
3. Describe the use of the dict() method for converting a list or tuple to a dictionary.
4. Describe the use of the merge and update operators with dictionaries.
5. Describe the way you access items when working with a dictionary of dictionaries, a dictionary of lists, or a list of dictionaries.

The syntax for creating a dictionary

```
dictionary_name = {key1:value1, key2:value2 ...}
```

Code that creates dictionaries

```
# strings as keys and values
countries = {"CA": "Canada",
             "US": "United States",
             "MX": "Mexico"}

# numbers as keys, strings as values
numbers = {1: "One", 2: "Two", 3: "Three",
           4: "Four", 5: "Five"}

# strings as keys, values of mixed types
movie = {"name": "The Holy Grail",
         "year": 1975,
         "price": 9.99}

# an empty dictionary
book_catalog = {}
```

Code that prints a dictionary to the console

```
print(countries)
```

The console

```
{'MX': 'Mexico', 'CA': 'Canada', 'US': 'United States'}
```

The countries dictionary

```
countries = {"CA": "Canada",
             "US": "United States",
             "GB": "Great Britain",
             "MX": "Mexico"}
```

The syntax for accessing a value

```
dictionary_name[key]
```

Code that gets a value from a dictionary

```
country = countries["MX"]   # "Mexico"
country = countries["IE"]   # KeyError: Key doesn't exist
```

Code that sets a value if the key is in the dictionary

```
countries["GB"] = "United Kingdom"
```

Code that adds a key/value pair if the key isn't in the dictionary

```
countries["FR"] = "France"
```

The syntax for checking if a key exists

```
key in dictionary
```

Code that checks the key before getting its value

```
code = "IE"
if code in countries:
    country = countries[code]
    print(country)
else:
    print(f"There is no country for this code: {code}")
```

The get()method of a dictionary object

```
get(key[, default_value])
```

Code that uses the get() method

```
country = countries.get("MX")                # "Mexico"
country = countries.get("IE")                # None
country = countries.get("IE", "Unknown")     # "Unknown"
```

The syntax for deleting an item

```
del dictionary_name[key]
```

Code that uses the del keyword to delete an item

```
del countries["MX"]
del countries["IE"]      # KeyError: Key doesn't exist
```

Code that checks a key before deleting the item

```
code = "IE"
if code in countries:
    country = countries[code]
    del countries[code]
    print(f"{country} was deleted.")
else:
    print(f"There is no country for this code: {code}")
```

Two dictionary methods for deleting items

```
pop(key[, default_value])
clear()
```

Code that uses the pop() method to delete an item

```
country = countries.pop("US")        # "United States"
country = countries.pop("IE")              # KeyError
country = countries.pop("IE", "Unknown")   # "Unknown"
```

Code that prevents a KeyError from occuring

```
code = "IE"
country = countries.pop(code, "Nothing")
print(f"{country} was deleted.")
```

Code that uses the clear() method to delete all items

```
countries.clear()
```

Three dictionary methods for getting all keys and values

```
keys()
items()
values()
```

Code that loops through all keys and values

```
for code in countries.keys():
    print(f"{code}      {countries[code]}")
```

Another way to get the same result since the default iterator contains keys

```
for code in countries:
    print(f"{code}      {countries[code]}")
```

The console

```
MX      Mexico
US      United States
CA      Canada
```

Code that unpacks a tuple as it loops through all keys and values

```
for code, name in countries.items():
    print(f"{code}      {name}")
```

The console

```
MX      Mexico
US      United States
CA      Canada
```

Code that loops through all values

```
for name in countries.values():
    print(name)
```

The console

```
Mexico
United States
Canada
```

Built-in constructors for creating dictionaries and lists

```
list(view)
dict(list)
```

Code that converts the keys of a dictionary to a list and sorts them

```
countries = {"CA": "Canada",
             "US": "United States",
             "MX": "Mexico"}
codes = list(countries.keys())
codes.sort()
for code in codes:
    print(f"{code}     {countries[code]}")
```

The console

```
CA      Canada
MX      Mexico
US      United States
```

Code that converts a two-dimensional list to a dictionary

```
countries = [["GB", "United Kingdom"],
             ["NL", "Netherlands"],
             ["DE", "Germany"]]
countries = dict(countries)
print(countries)
```

The console

```
{'NL': 'Netherlands', 'GB': 'United Kingdom',
'DE': 'Germany'}
```

The user interface for the Country Code program

```
COMMAND MENU
view - View country name
add  - Add a country
del  - Delete a country
exit - Exit program

Command: view
Country codes: CA MX US
Enter country code: mx
Country name: Mexico.

Command: add
Enter country code: nl
Enter country name: netherlands
Netherlands was added.

Command: view
Country codes: CA MX NL US
Enter country code: nl
Country name: Netherlands.

Command: del
Enter country code: us
United States was deleted.

Command: exit
Bye!
```

The code for the Country Code program (part 1)

```
def display_menu():
    print("COMMAND MENU")
    print("view - View country name")
    print("add  - Add a country")
    print("del  - Delete a country")
    print("exit - Exit program")
    print()

def display_codes(countries):
    codes = list(countries.keys())
    codes.sort()
    codes_line = "Country codes: "
    for code in codes:
        codes_line += code + " "
    print(codes_line)
```

The code for the Country Code program (part 2)

```
def view(countries):
    display_codes(countries)
    code = input("Enter country code: ")
    code = code.upper()
    if code in countries:
        name = countries[code]
        print(f"Country name: {name}.\n")
    else:
        print("There is no country with that code.\n")

def add(countries):
    code = input("Enter country code: ")
    code = code.upper()
    if code in countries:
        name = countries[code]
        print(f"{name} is already using this code.\n")
    else:
        name = input("Enter country name: ")
        name = name.title()
        countries[code] = name
        print(f"{name} was added.\n")
```

The code for the Country Code program (part 3)

```
def delete(countries):
    code = input("Enter country code: ")
    code = code.upper()
    if code in countries:
        name = countries.pop(code)
        print(f"{name} was deleted.\n")
    else:
        print("There is no country with that code.\n")
```

The code for the Country Code program (part 4)

```
def main():
    countries = {"CA": "Canada",
                 "US": "United States",
                 "MX": "Mexico"}

    display_menu()
    while True:
        command = input("Command: ")
        command = command.lower()
        if command == "view":
            view(countries)
        elif command == "add":
            add(countries)
        elif command == "del":
            delete(countries)
        elif command == "exit":
            print("Bye!")
            break
        else:
            print("Not a valid command. Please try again.\n")

if __name__ == "__main__":
    main()
```

The user interface for the Word Counter program

```
The Word Counter program

a = 7
above = 1
add = 1
...
```

The code for the Word Counter program (part 1)

```
def get_words_from_file(filename):
    with open(filename) as file:
        text = file.read()    # read str from file

    text = text.replace("\n", "")
    text = text.replace(",", "")
    text = text.replace(".", "")
    text = text.lower()

    words = text.split(" ")   # convert str to list
return words
```

The code for the Word Counter program (part 2)

```
def count_words(words):
    # define a dict to store the word count
    word_count = {}
    for word in words:
        if word in word_count:
            word_count[word] += 1  # increment count for word
        else:
            word_count[word] = 1   # add word with count of 1
    return word_count

def display_word_count(word_count):
    words = list(word_count.keys())
    words.sort(key=str.lower)
    for word in words:
        count = word_count[word]
        print(word, "=", count)
```

The code for the Word Counter program (part 3)

```
def main():
    print("The Word Counter program")
    print()

    # change filename to switch text file
    filename = "gettysburg_address.txt"

    # get words, count, and display
    words = get_words_from_file(filename) # get list of words
    word_count = count_words(words)       # create dict from list
    display_word_count(word_count)

if __name__ == "__main__":
    main()
```

A dictionary that contains other dictionaries as values (part 1)

```
contacts = {
    "Joel":
        {"address": "1500 Anystreet", "city": "San Francisco",
         "state": "California", "postalCode": "94110",
         "phone": "555-555-1111"},
    "Anne":
        {"address": "1000 Somestreet", "city": "Fresno",
         "state": "California", "postalCode": "93704",
         "phone": "555-555-2222"},
    "Ben"
        {"address": "1400 Another Street", "city": "Fresno",
         "state": "California", "postalCode": "93704",
         "phone": "555-555-4444"}
}
```

Code that gets values from embedded dictionaries

```
phone = contacts["Anne"]["phone"]   # "555-555-2222"
email = contacts["Anne"]["email"]   # KeyError
```

A dictionary that contains other dictionaries as values (part 2)

Code that checks whether a key exists within another key

```
key = "email"
if key in contacts["Anne"]:
    email = contacts["Anne"][key]
    print(email)
else:
    print("Sorry, there is no email address for this contact.")
```

Code that uses the get() method with embedded dictionaries

```
phone = contacts.get("Anne").get("phone")      # "555-555-2222"
phone = contacts.get("Anne").get("email")      # None
phone = contacts.get("Mike").get("phone")      # AttributeError
phone = contacts.get("Mike", {}).get("phone") # None
```

A dictionary that contains lists as values

```
students = {"Joel":[85, 95, 70],
            "Anne":[95, 100, 100],
            "Mike":[77, 70, 80, 85]}
```

Code that gets a value from an embedded list

```
scores = students["Joel"]           # [85, 95, 70]
joel_score1 = students["Joel"][0]   # 85
```

The user interface for the Book Catalog program

```
COMMAND MENU
show - Show book info
add -  Add book
edit - Edit book
del -  Delete book
exit - Exit program

Command: show
Title: Heart of Darkness
Sorry, Heart of Darkness doesn't exist in the catalog.

Command: add
Title: Heart of Darkness
Author name: Joseph Conrad
Publication year: 1890

Command: edit
Title: Heart of Darkness
Author name: Joseph Conrad
Publication year: 1899

Command:
```

The code the Book Catalog program (part 1)

```
def show_book(book_catalog):
    title = input("Title: ")
    if title in book_catalog:
        book = book_catalog[title]
        print(f"Title:    {title}")
        print(f"Author:   {book['author']}")
        print(f"Pub year: {book['pubyear']}")
    else:
        print(f"Sorry, {title} doesn't exist in the catalog.")
```

The code the Book Catalog program (part 2)

```
def add_edit_book(book_catalog, mode):
    title = input("Title: ")
    if mode == "add" and title in book_catalog:
        print(f"{title} already exists in the catalog.")
        response = input (
            "Would you like to edit it? (y/n): ").lower()
        if(response != "y"):
            return
    elif mode == "edit" and title not in book_catalog:
        print(title + " doesn't exist in the catalog.")
        response = input(
            "Would you like to add it? (y/n): ").lower()
        if (response != "y"):
            return

    # Get remaining book data and create a dictionary for the data
    author = input("Author name: ")
    pubyear = input("Publication year: ")
    book = {title: {"author": author, "pubyear": pubyear}}

    # Add the book data to the catalog using the update operator
    book_catalog |= book
```

The code the Book Catalog program (part 3)

```
def delete_book(book_catalog):
    title = input("Title: ")
    if title in book_catalog:
        del book_catalog[title]
        print(f"{title} removed from catalog.")
    else:
        print(f"{title} doesn't exist in the catalog.")

def display_menu():
    print("The Book Catalog program")
    print()
    print("COMMAND MENU")
    print("show - Show book info")
    print("add -  Add book")
    print("edit - Edit book")
    print("del -  Delete book")
    print("exit - Exit program")
```

The code the Book Catalog program (part 4)

```
def main():
    book_catalog = {
        "Moby Dick":
            {"author" : "Herman Melville",
             "pubyear" : "1851"},
        "The Hobbit":
            {"author" : "J. R. R. Tolkien",
             "pubyear" : "1937"},
        "Slaughterhouse Five":
            {"author" : "Kurt Vonnegut",
             "pubyear" : "1969"}
        }

    display_menu()
```

The code the Book Catalog program (part 5)

```
        while True:
            print()
            command = input("Command: ").lower()
            if command == "show":
                show_book(book_catalog)
            elif command == "add":
                add_edit_book(book_catalog, mode="add")
            elif command == "edit":
                add_edit_book(book_catalog, mode="edit")
            elif command == "del":
                delete_book(book_catalog)
            elif command == "exit":
                print("Bye!")
                break
            else:
                print("Unknown command. Please try again.")

    if __name__ == "__main__":
        main()
```