Chapter 6: Dictionaries

Contents

Red light: No AI

Time required: 90 minutes

DRY

Don't **R**epeat **Y**ourself

What Is a Dictionary?

In real life, a dictionary is a collection of words and their meanings.

In Python, a **dictionary** stores information in a similar way—each piece of data has a **key** (like the word) and a **value** (like the meaning).

Think of it like a mini-database. You look up a key, and Python gives you back the value.

Why Use Dictionaries?

Dictionaries are useful when:

- You want to label your data.
- You want to look things up quickly.
- You need a flexible, easy way to store and retrieve values.

How Are Dictionaries Different from Lists?

Feature	**List**	**Dictionary**
Access	By index (e.g., `mylist[0]`)	By key (e.g., `mydict['name']`)
Keys/Indexes	Numbers only	Can be strings, numbers, or tuples
Order Matters?	Yes	Not logically, but preserved (3.7+)
Duplicate Keys?	Not applicable	Keys must be unique

Both lists and dictionaries:

- Can grow or shrink as needed.
- Can hold any type of data.
- Can be nested (a list inside a dict, or a dict inside a list).

Creating a Dictionary

Dictionaries use curly braces `{}` and key-value pairs, with a colon `:` separating each pair.

```
# Create a dictionary of baseball teams
mlb_team = {
    'Colorado': 'Rockies',
    'Boston': 'Red Sox',
    'Minnesota': 'Twins'
}
```

Accessing Data in a Dictionary

To get the value, use the key inside square brackets:

```
print(mlb_team['Boston'])  # Output: Red Sox
```

If you try to access a key that doesn't exist, Python will give an error.

Use **.get()** to avoid that error:

```
print(mlb_team.get('Texas'))  # Output: None (no error)
```

Adding or Changing Items

You can add new key-value pairs or change existing ones using this syntax:

```
mlb_team['Seattle'] = 'Mariners'  # Add
mlb_team['Boston'] = 'Patriots'   # Change existing value
```

Removing Items

To delete a key-value pair, use `del`:

```
del mlb_team['Boston']
```

Example: English to Spanish Dictionary

```
# Create an empty dictionary
spanish = {}

# Add word translations
spanish['hello'] = 'hola'
spanish['yes'] = 'si'
spanish['one'] = 'uno'

# Access values
print(Spanish.get('hello'))     # Output: hola
print(spanish.get('goodbye'))   # Output: None
```

Dictionaries vs Lists – Which Should You Use?

With a list:

```
# Days in months using list
days = [31, 28, 31, 30, 31]
print(days[0])  # Output: 31 (January)
```

But what does index 0 mean to a new reader?

With a dictionary:

```
days = {
    'January': 31,
    'February': 28,
    'March': 31
}
print(days['January'])  # Output: 31
```

The dictionary version is more readable and easier to understand.

Key Notes

- **Keys must be unique**. If you assign a new value to an existing key, the old value is replaced.
- **Keys must be immutable** (strings, numbers, tuples). Lists cannot be keys.
- **Values can be anything**: strings, numbers, lists, even other dictionaries.

Common Dictionary Operations

Operation	Syntax	Description
Create empty	`d = {}`	Makes a new, empty dictionary
Add/update item	`d['A'] = 100`	Adds or updates key 'A'
Access value	`d['A']`	Returns value of key 'A'
Safe access	`d.get('A')`	Returns value or None (no error)
Delete item	`del d['A']`	Removes key 'A' and its value

Tutorial 6.1: Create a Dictionary

You can use a dictionary as an actual dictionary of definitions. Create the following program and save it as **cats_and_dogs.py**

```
"""
    Name: cats_and_dogs.py
    Author:
    Created:
    Purpose: Create and use a dictionary
"""


def main():
    # Define the key : value pairs of the dictionary
    dictionary = {
        'dog': 'has a tail and goes woof!',
        'cat': 'says meow',
        'mouse': 'is chased by cats'
    }

    # Prompt the user to enter a dictionary key
    print(" This dictionary contains values for dog, cat, or mouse.")
    word = input(" Enter a word (key): ")

    # Use the key entered by the user to access the value
    print(f" The value associated with {word}: {dictionary.get(word)}")
    print(f" A {word} {dictionary.get(word)}")


# Call the main function
if __name__ == '__main__':
    main()
```

Example run:

```
This dictionary contains values for dog, cat, or mouse.
Enter a word (key): mouse
The value associated with mouse: is chased by cats
A mouse is chased by cats
```

for Loop With a Dictionary

In Python, you can use a for loop to go through all the key-value pairs in a dictionary. This is useful when you want to **read**, **print**, or **process** all the data stored in the dictionary.

Basic Dictionary

```
student_grades = {
    'Alice': 90,
    'Bob': 85,
    'Clara': 92
}
# Loop Through Keys
for name in student_grades:
    print(name)
# Loop Through Keys and Access Values
for name in student_grades:
    print(name, "has a grade of", student_grades.get(name))
# Loop Through Key-Value Pairs
for name, grade in student_grades.items():
    print(name, "->", grade)
```

Example run:

```
Alice
Bob
Clara
Alice has a grade of 90
Bob has a grade of 85
Clara has a grade of 92
Alice -> 90
Bob -> 85
Clara -> 92
```

Summary

Loop Type	Syntax	Use
Keys only	for key in dict:	Loop through keys

Keys + Values	for key in dict: + dict[key]	Access values manually
Key-value pairs	for key, value in dict.items():	Clean and readable

Assignment 6.1: Student Grade Tracker

Create a Python file named: **dictionary_practice.py**

Exercise 1: Creating a Dictionary

Task:
Create a dictionary named **student_grades** that stores the following student names and their grades:

- Alice: 90
- Bob: 85
- Clara: 92

Then:

- Print the entire dictionary
- Print Clara's grade

TODO to get your program started.

```
# Your code here
student_grades = {
    # Add key-value pairs
}

# Print the full dictionary

# Print Clara's grade
```

Exercise 2: Adding and Changing Values

Task:
Using the **student_grades** dictionary from Exercise 1:

- Add a new student: "David" with a grade of 88
- Change Bob's grade to 89

- Print the updated dictionary

Exercise 3: Using .get() Safely

Task:
Use **.get()** to retrieve grades for:

- "Alice" (should return a grade)
- "Emma" (should return None)

Print the results with a message for each.

Exercise 4: Deleting a Key

Task:
Delete "Alice" from the **student_grades** dictionary using **del**

Print the dictionary to confirm removal.

Example run:

```
Student Grades:
{'Alice': 90, 'Bob': 85, 'Clara': 92}

Clara's Grade:
92

Added David:
{'Alice': 90, 'Bob': 85, 'Clara': 92, 'David': 88}

Updated Bob's grade:
{'Alice': 90, 'Bob': 89, 'Clara': 92, 'David': 88}

Using .get() to find a grade:
Emma's grade: None
Alice's grade: 90

After deleting Alice:
{'Bob': 89, 'Clara': 92, 'David': 88}

List all students and their grades:
Bob has a grade of 89
Clara has a grade of 92
David has a grade of 88
```

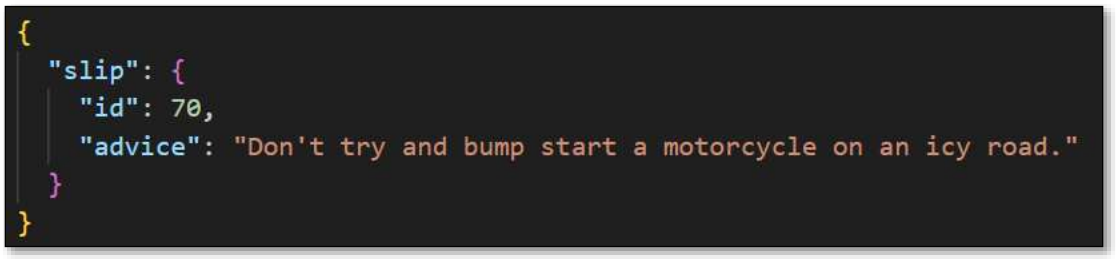
Tutorial 6.2: Advice to Live By API

Web API's return information in JSON (Javascript Object Notation) format. JSON is very similar to a Python dictionary.

Click on the following URL to see the raw JSON format in a web browser.

https://api.adviceslip.com/advice

When you copy and paste this into a file with a .json extension in VSCode, you get this. This file is named **advice_json.json** With this, we can figure out how to access the data.



This JSON is made up of a dictionary inside of a dictionary.

```
# Get the key of the first dictionary, which is a dictionary
data.get('slip')
# Get the key of the second dictionary, which return the advice value
.get('advice')
# All together
data.get('slip').get('advice')}
```

To access information on the web with Python, we need to install or update a Python module, requests.

```
# Install requests
pip install requests
# Update requests
pip install requests -U
```

This is the specific URL we are using to make our API request. This will return random quote. You can copy and paste this into your code.

```
URL = "https://api.adviceslip.com/advice"
```

Create a Python program named: **advice.py**

Enter the following code. This code is similar to earlier API projects.

```python
"""
    Name: advice_cli.py
    Author: William Loring
    Created: 05/01/2021
    Purpose: Get random advice
"""

# Import the requests module
# pip install requests
import requests

# URL for single random advice
URL = "https://api.adviceslip.com/advice"


def main():
    fetch_data()


def fetch_data():
    # Use the requests.get() function to get JSON data
    response = requests.get(URL)
    # Converts json to Python dictionary
    data = response.json()

    # Print the data using the dictionary created from the API JSON data
    print()
    print(" ------------ Advice to Live By ------------ ")
    print(f" {data.get('slip').get('advice')}")


# If a standalone program, call the main function
# Else, use as a module
if __name__ == "__main__":
    main()
```

Assignment 6.2: Language Translator App

Write a Language Translator App that translates English to Spanish.

1. Create a Python program named **english_to_spanish_dictionary.py**
2. Use a main() function.

3. Import and use the **utils.py** module to print a nice title block for your program.
4. Create a dictionary with the following key value pairs.
    a. one uno
    b. two dos
    c. three tres
5. Print out the keys with a loop.
6. Ask the user to input a word to translate.
7. Pass the dictionary to a function.
8. Display the translation. Example run:

```
+--------------------------------+
|  English to Spanish Dictionay  |
+--------------------------------+
These are the Spanish words that you can translate.
one
two
three
Enter the word you wish to translate: three
three translates to: tres
```

Challenges

1. Add more words to your dictionary.
2. Translate to another or multiple languages.
3. Whatever you think might make your translator app more interesting.

Glossary

aliasing A circumstance where two or more variables refer to the same object.

delimiter A character or string used to indicate where a string should be split.

element One of the values in a list (or other sequence); also called items.

equivalent Having the same value.

index An integer value that indicates an element in a list.

identical Being the same object (which implies equivalence).

list A sequence of values.

list traversal Sequential accessing of each element in a list.

nested list A list that is an element of another list.

object Something a variable can refer to. An object has a type and a value.

reference The association between a variable and its value.

Assignment Submission

1. Attach the pseudocode or create a TODO.
2. Attach all tutorials and assignments.
3. Attach screenshots showing the successful operation of each tutorial program.
4. Submit in Blackboard.