Chapter 3 Dictionary

A dictionary is a collection that stores key/value pairs. The keys are like an index operator. In a list, the indexes are integers.

A dictionary cannot contain duplicate keys. Each key maps to one value. It is called a "dictionary" because it resembles a word dictionary, where the words are the keys and the words' definitions are the values. A dictionary is also known as a map, which maps each key to a value.

1. Creating a Dictionary

You can create a dictionary by enclosing the items inside a pair of curly braces ({}). Each item consists of a key, followed by a colon, followed by a value. The items are separated by commas. The item is in the form **key:value**.

Example 01: Create a dictionary with two items.

```
students = {"Student1":"John", "Student2":"Peter"}

print(students)
```

Output:

```
{'Student1': 'John', 'Student2': 'Peter'}
```

Keys and values can be any type. For example:

```
d1 = {"id": 1, 15: "ABC", "price": 10.5, 5.8: "Five", True: 1}
```

You can create an empty dictionary:

```
students = {}
```

2. Adding, Modifying, and Retrieving Values

To add an item to a dictionary, use the syntax:

```
dictionaryName[key] = value
```

Example 02: Add a new item to a dictionary.

```
students = {"Student1":"John", "Student2":"Peter"}

students["Student3"] = "Susan"

print(students)
```

Output:

```
{'Student1': 'John', 'Student2': 'Peter', 'Student3': 'Susan'}
```

If the key is already in the dictionary, the preceding statement replaces the value for the key.

Example 03: Add a new item to a dictionary.

```
students = {"Student1":"John", "Student2":"Peter"}

students["Student1"] = "Susan"

print(students)
```

Output:

```
{'Student1': 'Susan', 'Student2': 'Peter'}
```

To retrieve a value, simply write an expression using `dictionary_name[key]`. If the key is in the dictionary, the value for the key is returned. Otherwise, a KeyError exception is raised.

Example 04: Retrieve value from a dictionary.

```
students = {"Student1":"John", "Student2":"Peter"}

print(students["Student1"])
print(students["Student2"])
```

Output:

```
John
Peter
```

3. Deleting Items

To delete an item from a dictionary, use the syntax:

```
del dictionary_name[key]
```

Example 05: Delete an item from a dictionary.

```
students = {"Student1":"John", "Student2":"Peter"}

del students["Student1"]

print(students)
```

Output:

```
{'Student2': 'Peter'}
```

```
del students["234-56-9010"]
```

If the key is not in the dictionary, a KeyError exception is raised.

4. Looping Items

You can use a `for` loop to traverse all keys in the dictionary.

Example 06: Iterate a dictionary.

```
students = {"Student1":"John", "Student2":"Peter", "Student3":"Susan"}

for key in students:
    print(key, ":", students[key])
```

Output:

```
Student1 : John
Student2 : Peter
Student3 : Susan
```

5. The Dictionary Methods

Here are methods for manipulating a dictionary object:

Method	*Description*
`keys`()	Returns a sequence of keys.
`values`()	Returns a sequence of values.
`items`()	Returns a sequence of key-value pairs
`clear`()	Deletes all items in the dictionary.
`get`(key)	Returns the value for the key.
`pop`(key)	Removes the item for the key and returns its value.
`popitem`()	Removes the last element (key, value) pair from the dictionary
`copy`()	Returns a copy of a dictionary.

Example 07: Use the `keys`() method to extracts the keys of the dictionary and returns the list of keys as a view object.

```
numbers = {1: 'one', 2: 'two', 3: 'three'}

# Extracts the keys of the dictionary
keys = list(numbers.keys())

print(keys)
```

Output:

```
[1, 2, 3]
```

Example 08: Use the `values`() method returns a view object that displays a list of all the values in the dictionary.

```
marks = {'Physics':67, 'Maths':87}

list1 = list(marks.values())

print(list1)
```

Output

```
[67, 87]
```

Example 09: Use the `items()` method returns a view object that displays a list of all the items in the dictionary.

```
marks = {'Physics':67, 'Maths':87}

for k, v in marks.items():
    print(k, v)
```

Output

```
Physics 67
Maths 87
```

Example 10: Use the `clear()` method removes all items from the dictionary.

```
numbers = {1: "one", 2: "two"}

# Removes all the items from the dictionary
numbers.clear()

print(numbers)
```

Output

```
{}
```

Example 11: Use the `get()` method returns the value for the specified key if the key is in the dictionary.

```
marks = {"Physics":67, "Maths":87}

print(marks.get("Physics"))
```

Output

```
67
```

Example 12: The `pop()` method removes and returns an element from a dictionary having the given key.

```
marks = { "Physics": 67, "Chemistry": 72, "Math": 89 }

marks.pop("Chemistry")

print(marks)
```

Output

```
{'Physics': 67, 'Math': 89}
```

Example 13: The pop() method removes and returns an element from a dictionary having the given key.

```
marks = { "Physics": 67, "Chemistry": 72, "Math": 89 }

item = marks.pop("Chemistry")

print(item)
```

Output

```
72
```

Example 14: Use the popitem() method to remove the last element (key, value) pair in the dictionary.

```
marks = { "Physics": 67, "Chemistry": 72, "Math": 89 }

marks.popitem()

print(marks)
```

Output:

```
{'Physics': 67, 'Chemistry': 72}
```

Example 15: Use the copy() method to copy dictionaries.

```
d1 = {'Physics':67, 'Maths':87}

d2 = d1.copy()

print(d2)
```

Output:

```
{'Physics': 67, 'Maths': 87}
```

Example 16: Another example of finding the key of a specified value

```
dic = {1: "John", 2: "Michael", 3: "Shawn"}

for k, v in dic.items():
    if v ==  "Michael":
        break

print(k)
```

Output:

```
2
```

6. Equality Test

You can use the **==** and **!=** operators to test whether two dictionaries contain the same items regardless of the order of the items in the dictionaries.

Example 17: Compare dictionaries

```
d1 = {"red":41, "blue":3}
d2 = {"blue":3, "red":41}

print(d1 == d2)
print(d1 != d2)
```

Output

```
True
False
```

Note: You cannot use the comparison operators (**>**, **>=**, **<=**, and **<**) to compare dictionaries because the items are not ordered.

7. Check Whether a Key/Value Is in a Dictionary

Same as lists and strings, you can use the in or not in operator to determine whether a key is in the dictionary.

Example 18: Check if a key is in a dictionary

```
students = {"Student1":"John", "Student2":"Peter", "Student3":"Susan"}

print("Student1" in students.keys())
print("Student1" not in students.keys())
```

Output

```
True
False
```

Example 19: Check if a value is in a dictionary

```
students = {"Student1":"John", "Student2":"Peter", "Student3":"Susan"}

print("Mary" in students.values())
print("Mary" not in students.values())
```

Output

```
False
True
```

8. The Dictionary of Lists

You can store lists in a dictionary, or dictionaries in a list.

Example 20: Create a dictionary to store the three students (S1, S2, S3), each student has four scores.

```
scores = {"S1":[20, 28, 15, 20],
          "S2":[50, 45, 20, 5],
          "S3":[23, 22, 43, 12]}

print(scores["S1"])
print(scores["S1"][0])
print(scores["S1"][1])
```

Output:

```
[20, 28, 15, 20]
20
28
```

Note: You cannot use a list or dictionary as a dictionary key. If you try to use it as a key, it will throw an error.

9. The Dictionary Functions

Common dictionary functions:

Function	Description
`len()`	returns the number of items in the dictionary.
`min()`	returns the item with the lowest key/value in the dictionary.
`max()`	returns the item with the greatest key/value in the dictionary.
`sum()`	returns the sum of all item in the dictionary.

Example 21: Get the length of a dictionary.

```
students = {"Student1":"John", "Student2":"Peter"}

print(len(students))
```

Output:

```
2
```

Example 22: Use `min()` and `max()` functions

```
d = {"B": 1, "C": 3, "A": 5}

print(min(d.keys()))
print(max(d.keys()))

print(min(d.values()))
print(max(d.values()))
```

Output:

```
A
C
1
5
```

Example 23: Sum values of a dictionary.

```
d = {"B": 1, "C": 3, "A": 5}

print(sum(d.values()))
```

Output:

```
9
```

10. Dictionary Comprehension

Python has a shorthand for looping through dictionaries known as dictionary comprehension that allow you to turn for loops into one-liners.

The syntax for dictionary comprehension is:

```
{key:value for (key,value) in dict.items() if condition}
```

where the `if condition` part is optional.

For example, let's create a new dictionary that will store the square of the numbers in a dictionary by leaving out all the negative numbers.

```
old_dict = {'a': -1, 'b': 2, 'c': -3, 'd': 4, 'e': -5}
new_dict = {}

for k, v in old_dict.items():
    if v > 0:
        new_dict[k] = v * v

print(new_dict)
```

Output:

```
{'b': 4, 'd': 16}
```

Let's make the above `for` loop shorter by using a dictionary comprehension:

```
old_dict = {'a': -1, 'b': 2, 'c': -3, 'd': 4, 'e': -5}

new_dict = {k:v * v for k, v in old_dict.items() if v > 0}

print(new_dict)
```

Output:

```
{'b': 4, 'd': 16}
```

11. Multidimensional Dictionary

Like multidimension list, multidimensional dictionary is a dictionary that contains other dictionaries as its elements. It is considered as nested dictionary.

A two-dimensional dictionary (2D dictionary) consists of dictionaries of one-dimensional dictionaries and a three-dimensional dictionary (3D dictionary) consists of dictionaries of two-dimensional dictionaries, and so on. You can create n-dimensional dictionaries for any integer n.

Example 24: You can use a 2D dictionary to store exam scores for a class of three students (S1, S2, S3) with two exams (E1, E2).

```
scores = {"S1":{"E1":20,"E2":28},
          "S2":{"E1":50,"E2":45},
          "S3":{"E1":34,"E2":25}}

print(scores["S1"])
print(scores["S1"]["E1"])
print(scores["S1"]["E2"])
print(len(scores))
print(len(scores["S1"]))
```

Output:

```
{'E1': 20, 'E2': 28}
20
28
3
2
```

Example 25: You can use a three-dimensional dictionary to store exam scores for a class of three students (S1, S2, S3) with two exams (E1, E2), and each exam has two parts (multiple-choice and essay).

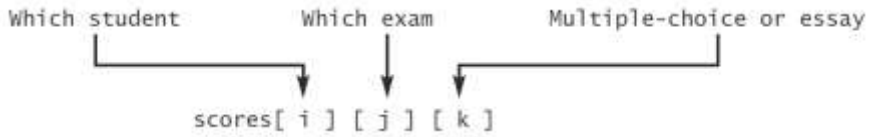
```
scores = {"S1":{"E1":{"MC":11,"Essay":20},"E2":{"MC":11, "Essay":22}},
          "S2":{"E1":{"MC":4, "Essay":21},"E2":{"MC":11, "Essay":22}},
          "S3":{"E1":{"MC":6, "Essay":30},"E2":{"MC":11, "Essay":11}}}

print(scores["S1"])
print(scores["S1"]["E2"])
print(scores["S1"]["E2"]["MC"])
print(scores["S1"]["E2"]["Essay"])
print(len(scores))
print(len(scores["S1"]))
print(len(scores["S1"]["E1"]))
```

Output

```
{'E1': {'MC': 11, 'Essay': 20}, 'E2': {'MC': 11, 'Essay': 22}}
{'MC': 11, 'Essay': 22}
11
22
3
2
2
```

`scores["S1"]["E2"]["MC"]` refers to the multiple-choice score for the first student's second exam, which is `11`. `scores["S1"]["E2"]["Essay"]` refers to the essay score for the first student's second exam, which is `22`.

The following figure depicts the meaning of the values in the dictionary.



13. `exit()` Function

The `exit()` function in Python is used to exit or terminate a program. You can use it to stop the execution of the program at any point. When the `exit()` function is called, the program will immediately stop running and exit.

Example 29: Using the `exit()` function

```
print("Before exit")
exit()
print("After exit")  # This line will not be executed
```

Output

```
Before exit
```

14. Justifying Output

In Python, strings have two methods called `ljust`() and `rjust`() that are used to align the strings.

Example 30: Aligning the output

```
print("Hello".ljust(10), "Python")
print("Hello".rjust(10), "Python")
```

Output

```
Hello      Python
     Hello Python
```

Practice

Display the following table:

ID	Name	Age
1	Chan Dara	20
2	Keo Tola	19
3	Sok Sophea	22

Exercises

Write the following programs. Note: you are NOT allowed to use lists.

1. Write a program that allows the user to create a 1D dictionary. The user should enter a key-value pair (separated by a comma) repeatedly, and type done when finished. Display the final dictionary.

2. Given a dictionary representing students and their subjects' scores:

```
students = {
    "John": {"Subject1": 20, "Subject2": 15},
    "Lucy": {"Subject1": 10, "Subject3": 30}
}
```

   Transform the dictionary so that it groups data by projects instead:

```
subjects = {
    "Subject1": {"John": 20, "Lucy": 10},
    "Subject2": {"John": 15},
    "Subject3": {"Lucy": 30}
}
```

3. Merge the following dictionaries:
   - `dict1 = {'a': 10, 'b': 20, 'c': 30}`
   - `dict2 = {'b': 15, 'c': 25, 'd': 35}`

   If a key exists in both dictionaries, sum their values. The result should be:
   - `{'a': 10, 'b': 35, 'c': 55, 'd': 35}`

4. Given a 2D dictionary representing survey data:

```
survey = {
    "Alice": {'Python': 5, "Java": 3},
    "Jack": {'Python': 4, "C++": 5},
    "Charlie": {'Java': 4, "Python": 3}
}
```

   Write a program to:
   - Calculate the average rating for each language.
   - Find the most popular language (highest average rating).

5. Store id, name, salary, and department of three employees in a dictionary. Then, write a program that includes a menu that will allow user to select any of the following features:
    a. Display all employees
    b. Add a new employee
    c. Delete employee by id
    d. Update employee by id
    e. Search employee by id
    f. Exit the program

When display the employee(s), the data should be arranged in a tabular format, for example:

ID	Name	Salary	Department
1	Chan Dara	80	ITE
2	Sok Sophea	900	BioE
3	Keo Tola	1200	DSE