

CHAPTER 09

SETS AND DICTIONARY

Learning Objectives

After completing this chapter, the reader will comprehend the following:

- Define and use sets in Python
- Create, access, update, and delete values in sets.
- Understand mathematical set operations.
- Understand the basics of a dictionary
- Use Built-in dictionary functions and methods
- Write simple Python programs using sets and a dictionary

9.1 Introduction to Sets

The chief characteristics of sets are given below:

- A Python set is a list of immutable elements that may or may not be ordered. It can contain various elements.
- A set may be modified, but the elements of the Set are immutable. Sets can include many immutable data types, like integers, strings, and tuples

- Python sets, can contain any hashable data type. Mutable data types or changeable items (variables) cannot be included as set elements. Lists can't be set elements, but tuples can.
- All set elements are unique.
- Set does not support indexing.

The primary advantage of sets is listed below:

- Sets have a solid mathematical foundation. Sets can support many mathematical functions, such as set union and set intersection.
- Searching for an element in a list takes lots of time. But items can be searched quickly in sets.
- A list can contain duplicates that may affect applications, like finding the count of unique words.
- Sets are better as there cannot be duplicate elements in sets.

9.2 Creating a Set

Python sets are constructed and specified by using curly braces. Set object items, which may or may not be the same type, are separated by commas (,) and then encircled by curly braces.

Python Syntax



```
set_variable = {<element>, <element>, ... <element>}
```

9.3 Basic Operation in sets

The length or size of the set is determined by using the built-in `len()` function. This function takes one argument, set name, and returns the total number of elements present in the set. The syntax of length function is given below:



`len()` is a standard function that applies to all sequence types, such as lists, strings, and tuples, and also for sets.

9.3 Basic Operation in sets

Membership operator in set

Table 9.1. Membership Operator

Operator	Syntax	Description
in	<element> in <set_variable_name>	It yields True if it identifies an element in set. Otherwise, it returns False.
not in	<element> not in <Set_variable_name>	If a set element is not in the set, the operator returns True; otherwise, it returns False.

Adding elements to sets

In certain circumstances, it is necessary to construct a set containing either no items or a certain number of items. This method is also known as "adding elements," which is another name for it. The `add()` function can include elemental values in the set. The syntax of `add` or `set.add(iterator)` is given as:



The function returns no value. It is added if the specified value does not exist in the set.

Deletion of elements from a set

Table 9.2: Set Deletion Methods

Methods	Descriptions
<code>remove(element)</code>	This function is used to remove a specific item of the Set.
<code>discard(element)</code>	Removes the element from the set by calling <code>discard(element)</code> . The Set is unaltered if the element is not present.
<code>pop()</code>	<code>pop()</code> can randomly delete an item from the Set. Then, the function returns the item that is removed.
<code>clear()</code>	To remove all the elements from a set, call <code>clear()</code>

Removal of an element in set

- When an element is to be removed from the set, the method `remove(element)` is called. The method removes the specific element and returns it. If the specified element is not present in the set, the method raises an `keyError` exception.



Discarding an element in a set

- The method `discard()` is used to discard an element of the set similar to the `remove()`, but the difference is that it does not raise an error, and the set remains unchanged

Python Syntax



```
set_variable.discard(element)
```

Deletion of a set

- A set deletion can be done using the set clear method.



Set copy

- The `set.copy()` method returns a shallow copy of the set. The result is a superficial replica of the set.



9.4 Mathematical Operations of Sets

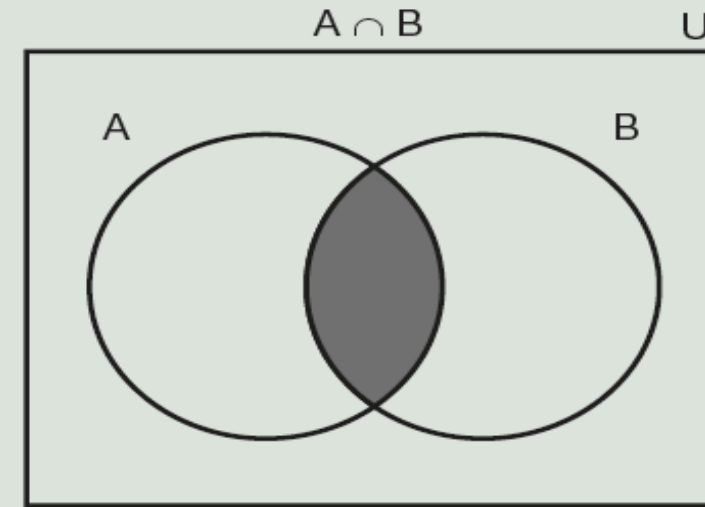
Table 9.3: Set Operations

Set Operations	Definition	Venn Diagram
Union	The union of two sets results in a set where the elements belong to either of the sets.	<p>Set Union</p> <p>$A \cup B$</p> <p>U</p> <p>A</p> <p>B</p>

Intersection

The intersection of two sets results in a set where the elements belong to both sets. (Common elements of both the sets)

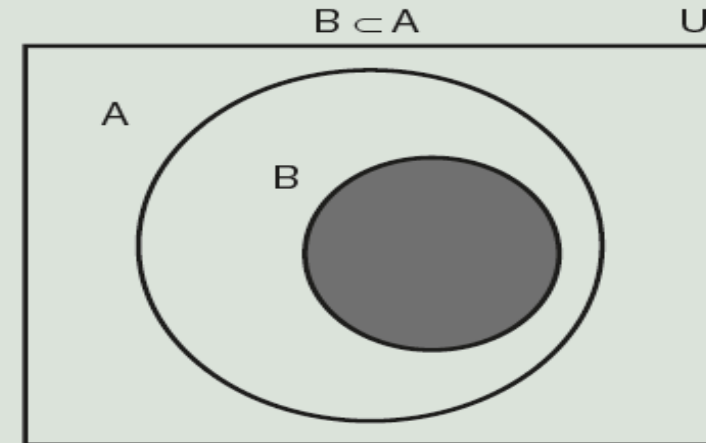
Set Intersection

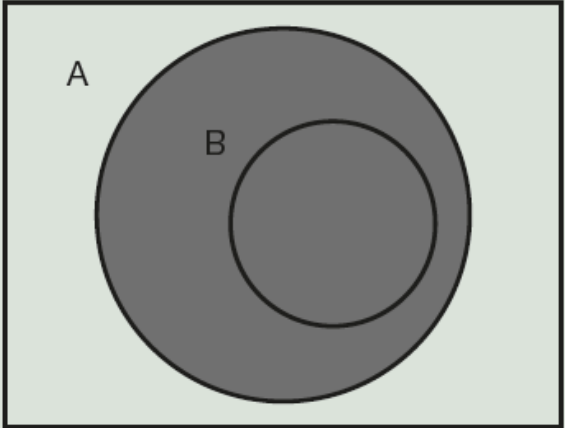
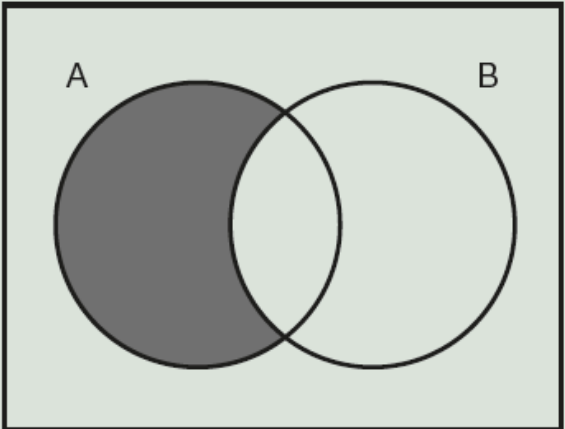


Proper Subset

If set B is the proper subset of set A, then all the elements of the set B belong to set A, and also set A and set B are not equal.

Proper Subset

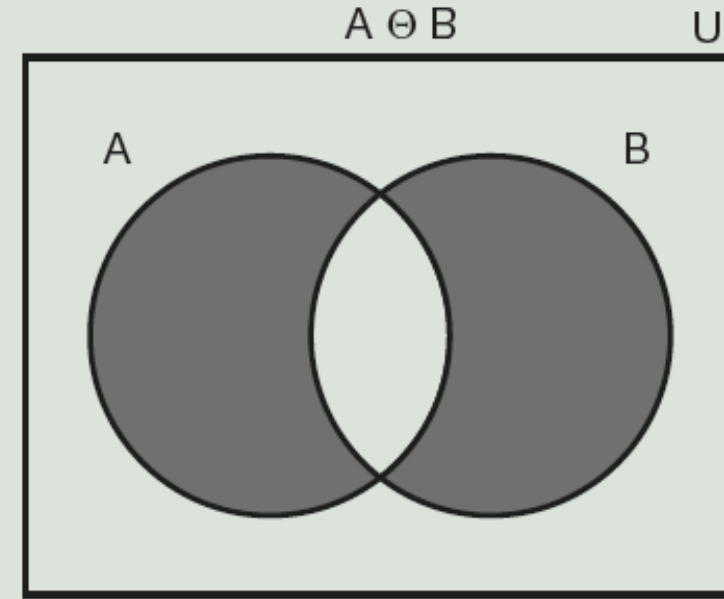


Set Operations	Definition	Venn Diagram
Proper Superset	If set A is the proper superset of set B, then all the elements of the set B belong to set A and set A contains more elements than set B.	<p>Proper Superset</p> <p>$A \supset B$ U</p> 
Set Difference	The set difference of set A from set B results in a set where its elements only belong to set A, not to set B.	<p>Set Difference</p> <p>$A - B$ U</p> 

Set Symmetric Difference

The symmetric set difference operator on Set A from Set B results in a set of elements that belongs either Set A or set B and not both sets.

Set Symmetric Difference



Python Syntax

```
set_variable1.union(set_variable2)  
set_variable1 | set_variable2
```

Subsets

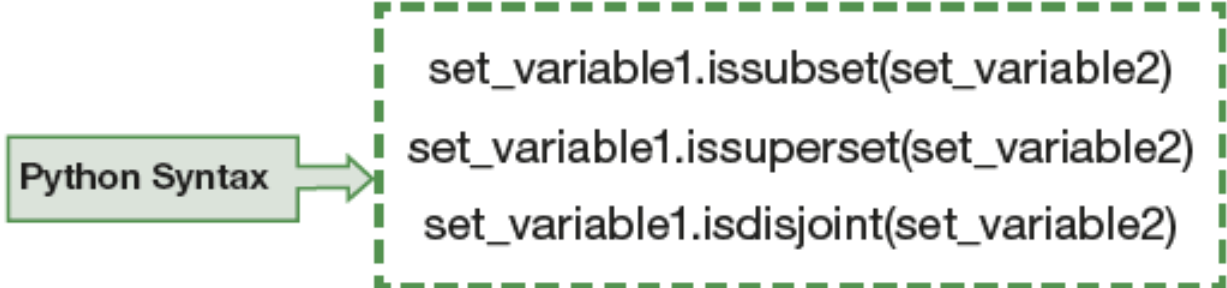
Table 9.4: Subset operations

Notation	Reference
$X == Y$	True is returned if the sets are equivalent; otherwise, it is False.
$X != Y$	True is returned if the sets are not equivalent; otherwise, it is False.
$X \leq Y$ or $X.issubset(Y)$	The true value for the operator is returned when the two sets are indistinguishable. X is a subset of Y. Test whether every element in X is in Y.
$X \geq Y$ or $X.issuperset(Y)$	The operator returns true only when the X and Y are supersets. X is a superset of Y. Test whether every element in X is in Y.
$X < Y$	X is a proper subset of Y
$X > Y$	X is a proper superset of Y

Boolean Functions

- three Boolean functions, `issubset()`, `issuperset()`, and `isdisjoint()`, to check whether a set is a subset, superset, or disjoint Set.

Python Syntax



```
set_variable1.issubset(set_variable2)  
set_variable1.issuperset(set_variable2)  
set_variable1.isdisjoint(set_variable2)
```

`X.issubset(Y)` checks if set X is a subset of set Y. The method returns True if X is a subset of set Y; otherwise, it returns False. `X.issuperset(Y)` checks if set X is a superset of Y. It returns True if X is a superset of Y; otherwise, it returns False.

`X.isdisjoint(Y)` method checks if a common element exists between sets X and Y. If there are no common elements, then sets X and Y are set to be disjoint sets. The method returns the value True if the sets are disjoint; otherwise, it is False.

Set updates using Mathematical Operations

Sets can be updated using the following methods.

- Symmetric_difference_update
- Difference_update
- Intersection_update
- Update

Symmetric_difference_update(): updates the calling set to include elements that are in one set but not the other.

Python Syntax

`set_variable1.set_difference_update(set_variable2)`

```
>>>setA = {1, 2, 3,4}
>>>setB = {2, 3, 4,5}
>>>setA.symmetric_difference_update(setB)
print(setA) # Output: {1, 5}
```

Difference_update(): Updates the calling set to include elements that are currently present in the calling set but not in the other set.

Python Syntax

`set_variable1.difference_update(set_variable2)`

```
>>>setA = {1, 2, 3,4}
>>>setB = {2, 3, 4,5}
>>>setA.difference_update(setB)
print(setA) # Output: {1}
```

Intersection_update()

The intersection_update() method is invoked to update the invoking set with the set intersection operation.

Python Syntax



```
set_variable1.intersection_update(set_variable2)
```

9.5 Looping Operations in Sets

- It is impossible to use an index to retrieve items in a set because it is an unordered container; hence, there are no available indexes. The problem is that they're iterable.
- With the for statement, it is possible to iterate over a set's elements

9.6 Frozen Sets

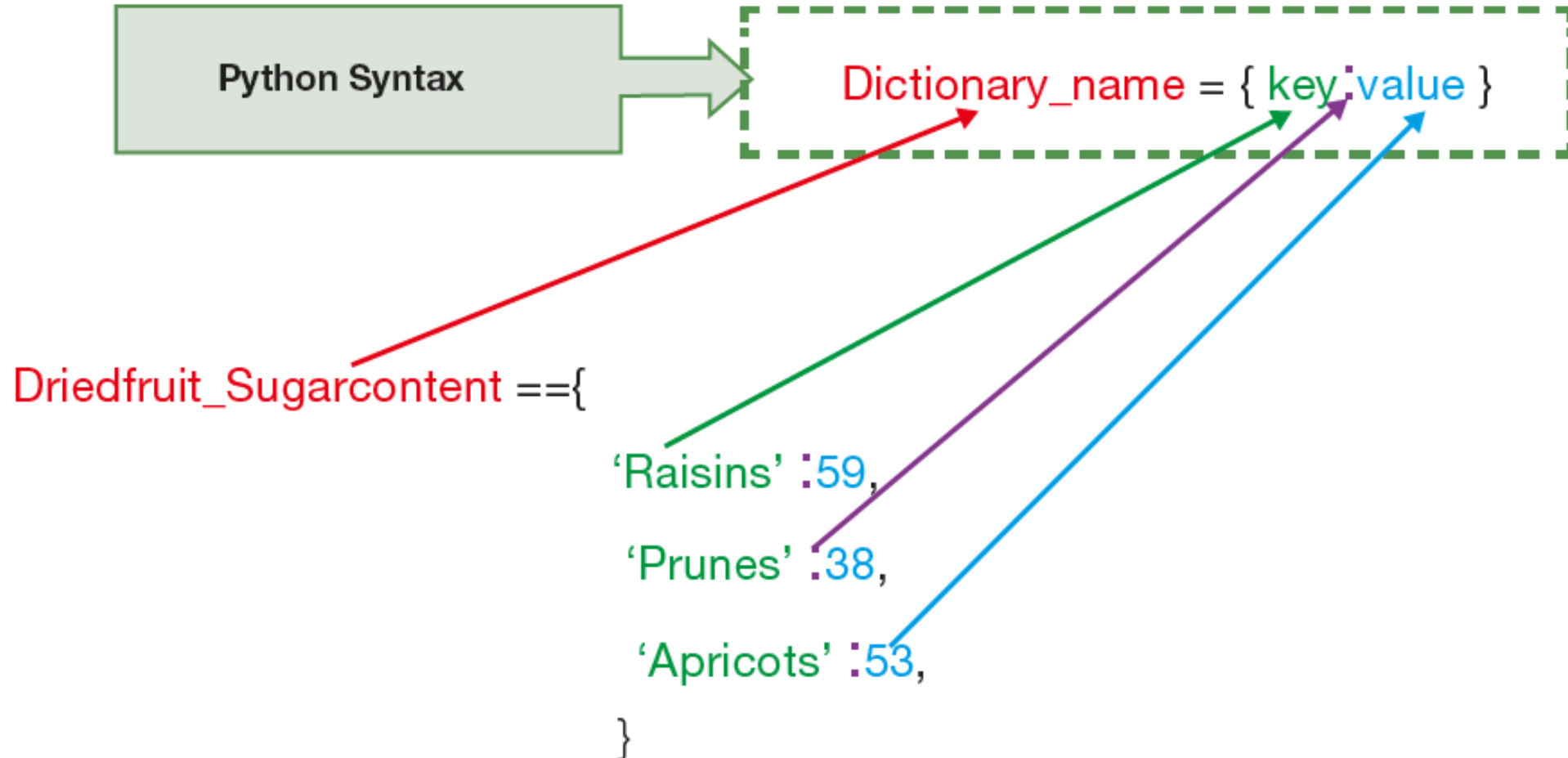
Table 9.5: Differences between set and frozenset

Sets	Frozen Sets
Python sets are unordered and unindexed.	Frozen sets also are unordered and unindexed.
Sets can be altered.	Frozenset object is immutable (cannot be altered).
The elements of a set can be changed at any moment	The elements of a frozen set cannot be changed.

Table 9.6: Frozenset operators and operations

Operations on frozenset	Syntax
Union	<code>frozenset1.union(frozenset2)</code>
Intersection	<code>frozenset1.intersection(frozenset2)</code>
Difference	<code>frozenset1.difference(frozenset2)</code>
Symmetric difference	<code>frozenset1.symmetric_difference(frozenset2)</code>

9.7 Introduction to Dictionary



- The value can be of any type. The values may be unique or mutable.
- Each shall occur only once in the dictionary.
- The ordering of the words does not matter, and the order in which users enter the words into the dictionary may not be the order in which the words appear

9.8 Creating Dictionaries

Door No	Name
2931	Harshita
2932	Kaira
2933	Zoya
2934	Radha
2936	Veena

A dictionary can be created primarily in two ways:

- By separating some or more key-value pairs enclosed by curly braces
- By using dict() method

An empty dictionary- opening and closing curly braces are used to form an empty dictionary.

The items included in a dictionary may not even be known when it is put together. Later, it can be filled in. It's thus possible to generate an empty dictionary either as {} or by using the dict() method.

Another way of creating a dictionary using keys and optimal values is using a function called fromkeys(). The method fromkeys() creates a new dictionary with keys from sequence and values set to value. The syntax is given as :



9.9 Basic Dictionary Operations

use membership operators(`in` and `not in`) to check whether a specific key is present.

To access the value is to use the `get()` method. returns a value for the given key. If the key is unavailable, it returns the default value `None`.

Python Syntax



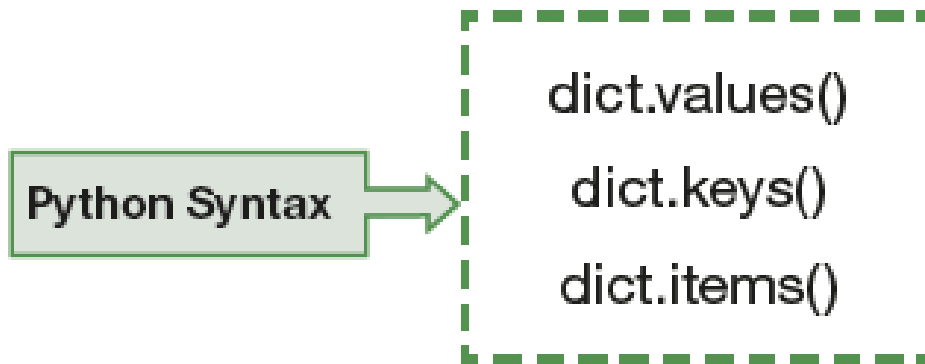
```
dict.get(key, default=None)
```

9.9 Basic Dictionary Operations

Instead of getting values one by one, one can get all the values in a single go using the methods listed below. One can do more with a dictionary with these functions

- `dict.keys()` isolates keys.
- `dict.values()` isolates values.
- `dict.items()` returns items in a list format of (key, value) tuple pairs.

The syntax is given as:



9.10. Operations on Python Dictionary

Adding an element

Enter the key's name between square brackets, and then use the assignment operator to assign the value to the key.

The index key will include the new key's name and matching value.



The dictionary must be updated if ABC Company moves to Bengaluru.

CityID	City Name
1	Delhi
2	Mumbai
3	Kolkatta
4	Chennai
5	Bengaluru

The extra information can be added as

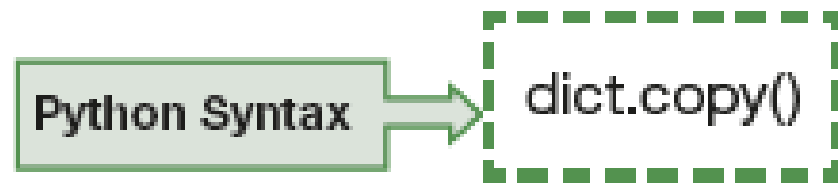
- `d1[cityId] = 5`
- `d1[city] = Bengaluru`

Copying the elements of the dictionary

In Python, there are numerous ways to clone a dictionary, as listed below:

- Element-by-element dictionary cloning: Every element pointed to a key is copied to a new dictionary, that was previously declared. A for loop can transfer the original dictionary's elements into a new, empty dictionary. A shallow copy is created using this technique. Changing this copy of the dictionary will not affect the original dictionary, (which will remain unchanged).
- `dictionary.copy()`: This Python function copies a dictionary shallowly. Without any parameters, it returns a copy of the dictionary. The original dictionary will stay untouched when changes are made to an abridged version. The iterable object's exception is present here, as it is in the for loop method.

The `copy()` method creates a new dictionary with a copy of the references from the original dictionary. The original dictionary is recreated each time the `=` operator is invoked. The syntax is given as :



Has no parameters.

A shallow copy creates a new compound object that references the original's objects. Recursively, the original object's components are copied into the new one, creating a new compound object. Aliasing is an issue because changes to one can affect the other when two variables refer to the same object. The copy method allows keeping a copy of the original and modifying the copy of the dictionary.

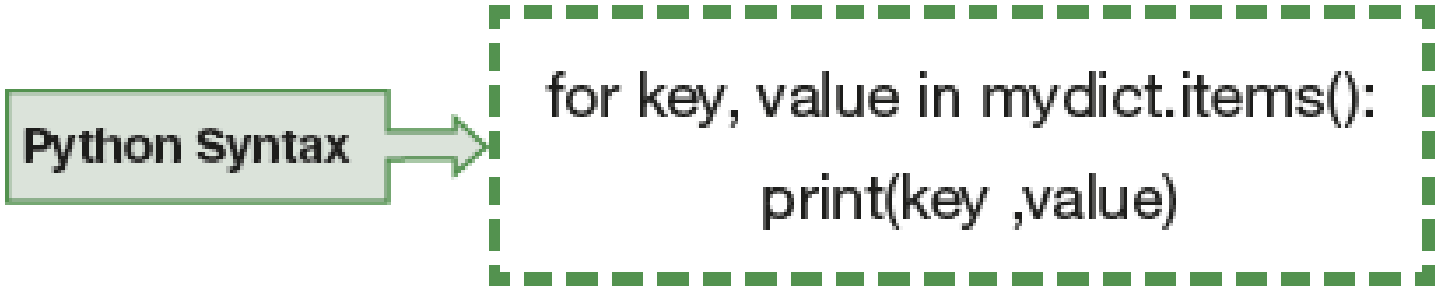
9.11 Nested dictionary

The idea of a nested dictionary is to stack dictionaries on top of one another. It's structured like a dictionary, with keys and values. The nested dictionary is not arranged in any particular order.

9.12 Looping over a dictionary

Access key deprived of using a `key()`, iterate through all key, and value pairs using `items()`, iterate through all values using `.values()`, print items in Key-Value in pair.

Python Syntax



```
for key, value in mydict.items():  
    print(key ,value)
```