MASTER OF COMPUTER APPLICATIONS (MCA)

Post Graduate Department of Computer Science and Technology

Sardar Patel University

Vallabh Vidyanagar, Gujarat, INDIA

PS01CMCA51 [Python Programming]

Unit-III

MR. BHARAT B. PATEL (ASSOCIATE PROFESSOR)

(✉) bb_patel@spuvvn.edu

Mapping type - dict

- ***The dictionary is an unordered collection that contains key:value pairs separated by commas inside curly brackets.***
- **Dictionary in Python is an unordered collection of data values, used to store data values like a map, which, unlike other Data Types that hold only a single value as an element, Dictionary holds key:value pair.**
- **Dictionaries are optimized to retrieve values when the key is known.**
- **Dictionaries are used to store data values in key:value pairs.**
- **Dictionaries are written with curly brackets, and have keys and values.**

Mapping type - dict

- A dictionary is a collection which is ordered [When we say that dictionaries are ordered, it means that the items have a defined order, and that order will not change] / unordered [means that the items does not have a defined order, you cannot refer to an item by using an index], changeable and do not allow duplicates.
- All the list objects are the objects of the dict class in Python.
- Use the dict() constructor is use to declare dictionary object.
- A dictionary items can be iterated using the for loop.

Mapping type - dict

Creating a Dictionary

- In Python, a Dictionary can be created by placing a sequence of elements within curly {} braces, separated by 'comma'.
- Dictionary holds pairs of values, one being the Key and the other corresponding pair element being its Key:value.
- Values in a dictionary can be of any data type and can be duplicated, whereas keys can't be repeated and must be immutable.
- Note – Dictionary keys are case sensitive, the same name but different cases of Key will be treated distinctly.
- Dictionary can also be created by the built-in function dict().
- An empty dictionary can be created by just placing to curly braces{}.

Example of dict

```
>>> dct1=dict ( )
>>> dct2= {}
>>> type ( dct1)
<class 'dict'>
>>> type ( dct2)
<class 'dict'>
>>> dct1= { 1:"One", 2:"Two", 3:"Three", 4:"Four"}
>>> print ( dct1)
 { 1: 'One', 2: 'Two', 3: 'Three', 4: 'Four'}
>>> print ( dct1[2])
Two
>>> print ( dct1["2"])
Traceback  ( most recent call last)  :
  File "<pyshell#7>", line 1, in <module>
    print ( dct1["2"])
KeyError: '2'
>>>
```

Python Dictionary Methods

Dictionary Method	Description
dict.clear()	Removes all the key-value pairs from the dictionary.
dict.copy()	Returns a shallow copy of the dictionary.
dict.fromkeys()	Creates a new dictionary from the given iterable (string, list, set, tuple) as keys and with the specified value.
dict.get()	Returns the value of the specified key.
dict.items()	Returns a dictionary view object that provides a dynamic view of dictionary elements as a list of key-value pairs. This view object changes when the dictionary changes.
dict.keys()	Returns a dictionary view object that contains the list of keys of the dictionary.
dict.pop()	Removes the key and return its value. If a key does not exist in the dictionary, then returns the default value if specified, else throws a KeyError.
dict.popitem()	Removes and return a tuple of (key, value) pair from the dictionary. Pairs are returned in Last In First Out (LIFO) order.

Python Dictionary Methods

Dictionary Method	Description
dict.setdefault()	Returns the value of the specified key in the dictionary. If the key not found, then it adds the key with the specified defaultvalue. If the defaultvalue is not specified then it set None value.
dict.update()	Updates the dictionary with the key-value pairs from another dictionary or another iterable such as tuple having key-value pairs.
dict.values()	Returns the dictionary view object that provides a dynamic view of all the values in the dictionary. This view object changes when the dictionary changes.
dict.clear()	Removes all the key-value pairs from the dictionary.
dict.copy()	Returns a shallow copy of the dictionary.
dict.fromkeys()	Creates a new dictionary from the given iterable (string, list, set, tuple) as keys and with the specified value.
dict.get()	Returns the value of the specified key.
dict.items()	Returns a dictionary view object that provides a dynamic view of dictionary elements as a list of key-value pairs. This view object changes when the dictionary changes.

Python Dictionary Methods

Dictionary Method	Description
dict.keys()	Returns a dictionary view object that contains the list of keys of the dictionary.
dict.pop()	Removes the key and return its value. If a key does not exist in the dictionary, then returns the default value if specified, else throws a KeyError.
dict.popitem()	Removes and return a tuple of (key, value) pair from the dictionary. Pairs are returned in Last In First Out (LIFO) order.
dict.setdefault()	Returns the value of the specified key in the dictionary. If the key not found, then it adds the key with the specified defaultvalue. If the defaultvalue is not specified then it set None value.
dict.update()	Updates the dictionary with the key-value pairs from another dictionary or another iterable such as tuple having key-value pairs.
dict.values()	Returns the dictionary view object that provides a dynamic view of all the values in the dictionary. This view object changes when the dictionary changes.

Python set type

- A set is a mutable collection of distinct hashable objects, same as the list and tuple.
- It is an unordered collection of objects, meaning it does not record element position or order of insertion and so cannot access elements using indexes.
- The set is a Python implementation of the set in Mathematics.
- A set object has suitable methods to perform mathematical set operations like union, intersection, difference, etc.
- A set object contains one or more items, not necessarily of the same type, which are separated by a comma and enclosed in curly brackets {}.
- A set doesn't store duplicate objects. Even if an object is added more than once inside the curly brackets, only one copy is held in the set object.
- Hence, indexing and slicing operations cannot be done on a set object.

Python set type

- The order of elements in the set is not necessarily the same as the order given at the time of assignment.
- Python optimizes the structure of a set for performing operations over it, as defined in mathematics.
- Only immutable (and hashable) objects can be a part of a set object.
- Numbers (integer, float, as well as complex), strings, and tuple objects are accepted, but set, list, and dictionary objects are not.
- The set() function is used to convert string, tuple, or dictionary object to a set object.

Mapping type - dict

- **Creating set objects :**

```
>>> set1 = set ()
>>> set2 = set ( {2,2,2,12,12,12} )
>>> type ( set1)
<class 'set'>
>>> type ( set2)
<class 'set'>
>>> set3 = { ( 1,2) , ( 3,4) , "Test", True, 12, "a",}
>>> len ( set3)
6
```

Python Set Methods

Dictionary Method	Description
set.add()	Adds an element to the set. If an element is already exist in the set, then it does not add that element.
set.clear()	Removes all the elements from the set.
set.copy()	Returns a shallow copy of the set.
set.difference()	Returns the new set with the unique elements that are not in the another set passed as a parameter.
set.discard()	Removes a specific element from the set.
set.intersection()	Returns a new set with the elements that are common in the given sets.
set.isdisjoint()	Returns true if the given sets have no common elements. Sets are disjoint if and only if their intersection is the empty set.

Python Set Methods

Dictionary Method	Description
set.issubset()	Returns true if the set (on which the issubset() is called) contains every element of the other set passed as an argument.
set.pop()	Removes and returns a random element from the set.
set.remove()	Removes the specified element from the set. If the specified element not found, raise an error.
set.union()	Returns a new set with distinct elements from all the given sets.
set.update()	Updates the set by adding distinct elements from the passed one or more iterables.