Dictionary in Python

Of all the built-in Python data types, the dictionary is easily the most interesting. It is the only standard mapping type, and it is the backbone of every Python object.

A dictionary maps keys to values. Keys need to be hashable objects, while values can be of any arbitrary type. Dictionaries are also mutable objects.

In Python, a dictionary is an unordered collection of items, with each item consisting of a key: value pair (separated by a colon).

Creating Dictionary in Python

You can create a dictionary by enclosing comma separated key: value pairs within curly braces {}. Like this:

```
my_dict = {"Key1": "Value1", "Key2": "Value2"}
```

Here is an example of creating a dictionary, then printing it out, along with its type:

```
student_info = {"name": "James", "roll": 45, "gender": "Male"}

# Print the dictionary
print(student_info)

# Print the type
print(type(student_info))
```

Output

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
<class 'dict'>
```

But that is not the only way to create a dictionary. There's also a **dict()** function for creating dictionaries. And you can also use syntax variations within that function. Here are some examples:

```
student_info = {"name": "James", "roll": 45, "gender": "Male"}
print(student_info)

student_info = dict({"name": "James", "roll": 45, "gender":
"Male"})
print(student_info)

student_info = dict([("name", "James"), ("roll", 45),
("gender", "Male")])
print(student_info)

student_info = dict(name="James", roll=45, gender="Male")
print(student_info)
```

Output

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
{'name': 'James', 'roll': 45, 'gender': 'Male'}
{'name': 'James', 'roll': 45, 'gender': 'Male'}
{'name': 'James', 'roll': 45, 'gender': 'Male'}
```

NOTE:

There are a few important things to remember when creating dictionaries in Python:

- Dictionaries do not support duplicate keys. Each key should be unique. For example, you cannot have two dictionary items with a key of "name". If you do this, Python will use the last one and ignore the first one.
- Dictionary keys must be immutable. That is, they must be of a data type that cannot be changed. So, you can use strings, numbers, or even tuples as dictionary keys, but you cannot use lists, or other dictionaries.
- Dictionary values do not have the above restrictions.

Access the Values in a Dictionary

There are two ways to access the values in a dictionary; the square bracket notation, and the get() function.

With both methods, you access an item by referring to its key. This is slightly different to accessing items within a list or tuple (where you access an item by referring to its index).

The key can be enclosed in square brackets, or passed as an argument to the get() function.

Here is an example using both methods:

```
student_info = {"name": "James", "roll": 45, "gender": "Male"}
print(student_info)

# Square brackets
print(student_info["name"])

# get() function
print(student_info.get("name"))
```

Output

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
James
James
```

Update a Dictionary

Dictionaries are mutable, so that means you can update items within the dictionary.

You can update the dictionary by referring to the item's key. If the key exists, it will update its value to the new value. If the key does not exist, it will create a new key:value pair.

```
student_info = {"name": "James", "roll": 45, "gender": "Male"}
print(student_info)

# Key exists, so it will update the value
student_info["name"] = 'Max'
print(student_info)

# Key does not exists, so it will add the key:value
student_info["phnumber"] = 5874747847
print(student_info)
```

Output

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
{'name': 'Max', 'roll': 45, 'gender': 'Male'}
{'name': 'Max', 'roll': 45, 'gender': 'Male', 'phnumber': 5874747847}
```

Deleting a Dictionary Item

You can use the del keyword to delete a given dictionary item. To do this, simply refer to the item's key.

```
student_info = {"name": "James", "roll": 45, "gender": "Male"}
print(student_info)

# It will delete a single item
del student_info["name"]
print(student_info)

# It will delete the whole dictionary
del student_info
```

Output

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
{'roll': 45, 'gender': 'Male'}
```

Built-In Dictionary Methods

Python has a set of built-in methods that you can use on dictionaries.

Method	Description
clear()	Removes all the elements from the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Python dictionary clear() method

The clear() method removes all the elements from a dictionary.

```
student_info = {"name": "James", "roll": 45, "gender": "Male"}
print(student_info)

student_info.clear()
print(student_info)
```

Output

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
{}
```

Python dictionary fromkeys() method

The fromkeys() method returns a dictionary with the specified keys and the specified value.

```
x = ('key1', 'key2', 'key3')
y = 0

mydict = dict.fromkeys(x, y)
print(mydict)
```

Output

```
{'key1': 0, 'key2': 0, 'key3': 0}
```

Python dictionary get() method

The get() method returns the value of the item with the specified key.

```python
student_info = {"name": "James", "roll": 45, "gender": "Male"}
print(student_info)

# If key exists, it will return the value of that key
print(student_info.get("roll"))

# If key does not exists, it will return None
print(student_info.get("address"))
```

Output

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
45
None
```

Python dictionary items() method

The items() method returns a view object. The view object contains the key-value pairs of the dictionary, as tuples in a list.

```python
student_info = {"name": "James", "roll": 45, "gender": "Male"}
print(student_info)

print(student_info.items())
```

Output

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
dict_items([('name', 'James'), ('roll', 45), ('gender', 'Male')])
```

Python dictionary keys() method

The keys() method returns a view object. The view object contains the keys of the dictionary, as a list.

```python
student_info = {"name": "James", "roll": 45, "gender": "Male"}

print(student_info.keys())
```

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
dict_keys(['name', 'roll', 'gender'])
```

Python dictionary pop() method

The pop() method removes the specified item from the dictionary.

```python
student_info = {"name": "James", "roll": 45, "gender": "Male"}
print(student_info)

student_info.pop("roll")
print(student_info)
```

Output

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
{'name': 'James', 'gender': 'Male'}
```

Python dictionary update() method

The update() method inserts the specified items to the dictionary.

```python
student_info = {"name": "James", "roll": 45, "gender": "Male"}
print(student_info)

new_dict = {"phno": 5656, "total_marks": 450}
student_info.update(new_dict)
print(student_info)
```

Output

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
{'name': 'James', 'roll': 45, 'gender': 'Male', 'phno': 5656, 'total_marks': 450}
```

Python dictionary values() method

The values() method returns a view object. The view object contains the values of the dictionary, as a list.

```python
student_info = {"name": "James", "roll": 45, "gender": "Male"}
print(student_info)

print(student_info.values())
```

Output

```
{'name': 'James', 'roll': 45, 'gender': 'Male'}
dict_values(['James', 45, 'Male'])
```

Dictionary containing Lists & Tuples

Dictionaries can also contain lists and tuples. Example:

```
student_info = {"name": "James", "roll": 45, "marks": [55, 78,
63, 23, 88]}

print(student_info)
```

Output

```
{'name': 'James', 'roll': 45, 'marks': [55, 78, 63, 23, 88]}
```

Access a List Item within a Dictionary

You can access a list item within a dictionary by referring to the list's key within the dictionary, followed by the list item's index within the list (each index surrounded by its own set of square brackets).

```
student_info = {"name": "James", "roll": 45, "marks": [55, 78,
63, 23, 88]}

print(student_info)
print(student_info["marks"])
print(student_info["marks"][0])
print(student_info["marks"][3])
```

Output

```
{'name': 'James', 'roll': 45, 'marks': [55, 78, 63, 23, 88]}
[55, 78, 63, 23, 88]
55
23
```

Update a List Item within a Dictionary

If your dictionary contains lists, you can update list items within those lists by referring to the list's key within the dictionary, followed by the list item's index within the list.

```
student_info = {"name": "James", "roll": 45, "marks": [55, 78, 
63, 23, 88]}
print(student_info["marks"])

student_info["marks"][2] = 100
print(student_info["marks"])
```

Output

```
[55, 78, 63, 23, 88]
[55, 78, 100, 23, 88]
```

Looping through dictionary in Python

You can loop through a dictionary by using a for loop. When looping through a dictionary, the return value are the keys of the dictionary, but there are methods to return the values as well.

Print all the key in dictionary one by one

```
student_info = {"name": "James", "roll": 45, "totalMarks": 
485}

for i in student_info:
    print(i)
```

Output

```
name
roll
totalMarks
```

Print all the values in dictionary one by one

```
student_info = {"name": "James", "roll": 45, "totalMarks":
485}

for i in student_info:
    print(student_info[i])
```

Output

```
James
45
485
```

Print all the values in dictionary using VALUES() method

```
student_info = {"name": "James", "roll": 45, "totalMarks":
485}

for i in student_info.values():
    print(i)
```

Output

```
James
45
485
```

Print both keys and values in dictionary using ITEMS() method

```
student_info = {"name": "James", "roll": 45, "totalMarks":
485}

for k, v in student_info.items():
    print(f"Key -> {k} and value -> {v}")
```

Output

```
Key -> name and value -> James
Key -> roll and value -> 45
Key -> totalMarks and value -> 485
```

Practice Examples (Dictionary)

1. Merge two Python dictionaries into one.
2. Write a python program to find the sum of all items in a dictionary.
3. Write a python program to check whether a given key already exists in a dictionary.
4. Write a Python program to iterate over dictionaries using for loops.
5. Write a Python program to generate and print a dictionary that contains a number (between 1 and n) in the form (x, x*x).
   Sample Dictionary: (n = 6)
   Expected Output: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}
6. Write a Python program to map two lists into a dictionary.
7. Write a Python program to get the maximum and minimum value in a dictionary.
8. Write a Python program to create a dictionary from a string.
   Note: Track the count of the letters from the string.
   Sample string: 'hellopython'
   Expected output: {'h': 2, 'e': 1, 'l': 2, 'o': 2, 'p': 1, 'y': 1, 't': 1, 'n': 1}
9. Write a Python program to print a dictionary in table format.
10. A Python Dictionary contains List as value. Write a Python program to clear the list values in the said dictionary.
    Original Dictionary:
    {'C1': [10, 20, 30], 'C2': [20, 30, 40], 'C3': [12, 34]}
    Clear the list values in the said dictionary:
    {'C1': [], 'C2': [], 'C3': []}
11. Write a Python program to convert a given dictionary into a list of lists.
    Original Dictionary:
    {1: 'red', 2: 'green', 3: 'black', 4: 'white', 5: 'black'}
    Convert the said dictionary into a list of lists:
    [[1, 'red'], [2, 'green'], [3, 'black'], [4, 'white'], [5, 'black']]

    Original Dictionary:
    {'1': 'Austin Little', '2': 'Natasha Howard', '3': 'Alfred Mullins', '4': 'Jamie Rowe'}
    Convert the said dictionary into a list of lists:
    [['1', 'Austin Little'], ['2', 'Natasha Howard'], ['3', 'Alfred Mullins'], ['4', 'Jamie Rowe']]
    For solutions, check the end of the book.