What is dictionary in Python?

Python dictionary is an unordered collection of items. While other compound data types have only value as an element, a dictionary has a key: value pair. Dictionaries are optimized to retrieve values when the key is known.

How to create a dictionary?

Creating a dictionary is as simple as placing items inside curly braces {} separated by comma. An item has a key and the corresponding value expressed as a pair, key: value. While values can be of any data type and can repeat, keys must be of immutable type (string, number or tuple with immutable elements) and must be unique.

```
# empty dictionary
my_dict = {}

# dictionary with integer keys
my_dict = {1: 'apple', 2: 'ball'}

# dictionary with mixed keys
my_dict = {'name': 'John', 1: [2, 4, 3]}

# using dict()
my_dict = dict({1:'apple', 2:'ball'})

# from sequence having each item as a pair
my_dict = dict([(1,'apple'), (2,'ball')])
```

As you can see above, we can also create a dictionary using the built-in function dict().

How to access elements from a dictionary?

While indexing is used with other container types to access values, dictionary uses keys. Key can be used either inside square brackets or with the get() method. The difference while using get() is that it returns None instead of KeyError, if the key is not found.

```
my_dict = {'name':'Jack', 'age': 26}
```

```
# Output: Jack
print(my_dict['name'])

# Output: 26
print(my_dict.get('age'))

# Trying to access keys which doesn't exist throws error
# my_dict.get('address')
# my_dict['address']
```

How to change or add elements in a dictionary?

Dictionary are mutable. We can add new items or change the value of existing items using assignment operator. If the key is already present, value gets updated, else a new key: value pair is added to the dictionary.

```
my_dict = {'name':'Jack', 'age': 26}

# update value
my_dict['age'] = 27

#Output: {'age': 27, 'name': 'Jack'}
print(my_dict)

# add item
my_dict['address'] = 'Downtown'

# Output: {'address': 'Downtown', 'age': 27, 'name': 'Jack'}
print(my_dict)
```

How to delete or remove elements from a dictionary?

We can remove a particular item in a dictionary by using the method pop(). This method removes as item with the provided key and returns the value.
The method, popitem() can be used to remove and return an arbitrary item (key, value) form the dictionary. All the items can be removed at once using the clear() method. We can also use the del keyword to remove individual items or the entire dictionary itself.

```
# create a dictionary
squares = {1:1, 2:4, 3:9, 4:16, 5:25}
```

```
# remove a particular item
# Output: 16
print(squares.pop(4))

# Output: {1: 1, 2: 4, 3: 9, 5: 25}
print(squares)

# remove an arbitrary item
# Output: (1, 1)
print(squares.popitem())

# Output: {2: 4, 3: 9, 5: 25}
print(squares)

# delete a particular item
del squares[5]

# Output: {2: 4, 3: 9}
print(squares)

# remove all items
squares.clear()

# Output: {}
print(squares)

# delete the dictionary itself
del squares

# Throws Error
# print(squares)
```
When you run the program, the output will be:
```
16
{1: 1, 2: 4, 3: 9, 5: 25}
(1, 1)
{2: 4, 3: 9, 5: 25}
{2: 4, 3: 9}
{}
```

Python Dictionary Methods

Methods that are available with dictionary are tabulated below. Some of them have already been used in the above examples.

Python Dictionary Methods

Method	Description
clear()	Remove all items form the dictionary.
copy()	Return a shallow copy of the dictionary.
fromkeys(seq[, v])	Return a new dictionary with keys from seq and value equal to v (defaults to None).
get(key[,d])	Return the value of key. If key doesnot exit, return d (defaults to None).
items()	Return a new view of the dictionary's items (key, value).
keys()	Return a new view of the dictionary's keys.
pop(key[,d])	Remove the item with key and return its value or d if key is not found. If d is not provided and key is not found, raises KeyError.
popitem()	Remove and return an arbitary item (key, value). Raises KeyError if the dictionary is empty.
setdefault(key[,d])	If key is in the dictionary, return its value. If not, insert key with a value of d and return d (defaults to None).
update([other])	Update the dictionary with the key/value pairs from other, overwriting existing keys.

values()	Return a new view of the dictionary's values

Here are a few example use of these methods.

```
marks = {}.fromkeys(['Math','English','Science'], 0)

# Output: {'English': 0, 'Math': 0, 'Science': 0}
print(marks)

for item in marks.items():
    print(item)

# Output: ['English', 'Math', 'Science']
list(sorted(marks.keys()))
```

Python Dictionary Comprehension

Dictionary comprehension is an elegant and concise way to create new dictionary from an iterable in Python. Dictionary comprehension consists of an expression pair (key: value) followed by for statement inside curly braces {}. Here is an example to make a dictionary with each item being a pair of a number and its square.

```
squares = {x: x*x for x in range(6)}

# Output: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
print(squares)
```

This code is equivalent to

```
squares = {}
for x in range(6):
   squares[x] = x*x
```

A dictionary comprehension can optionally contain more for or if statements. An optional if statement can filter out items to form the new dictionary. Here are some examples to make dictionary with only odd items.

```
odd_squares = {x: x*x for x in range(11) if x%2 == 1}

# Output: {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
print(odd_squares)
```

Other Dictionary Operations

Dictionary Membership Test

We can test if a key is in a dictionary or not using the keyword in. Notice that membership test is for keys only, not for values.

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

# Output: True
print(1 in squares)

# Output: True
print(2 not in squares)

# membership tests for key only not value
# Output: False
print(49 in squares)
```

Iterating Through a Dictionary

Using a for loop we can iterate though each key in a dictionary.

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
for i in squares:
    print(squares[i])
```

Built-in Functions with Dictionary

Function	Description

all()	Retusquares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
	
	# Output: 5
	print(len(squares))
	
	# Output: [1, 3, 5, 7, 9]
	print(sorted(squares))rn True if all keys of the dictionary are true (or if the dictionary is empty).

any()	Return True if any key of the dictionary is true. If the dictionary

	is empty, return False.
len()	Return the length (the number of items) in the dictionary.
cmp()	Compares items of two dictionaries.
sorted()	Return a new sorted list of keys in the dictionary.

Here are some examples that uses built-in functions to work with dictionary.

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

# Output: 5
print(len(squares))

# Output: [1, 3, 5, 7, 9]
print(sorted(squares))
```

Practice Questions on dictionary

1. Write a Python script to merge two Python dictionaries.
2. Write a Python program to iterate over dictionaries using for loops.
3. Write a Python program to sum all the items in a dictionary.
4. Write a Python program to multiply all the items in a dictionary.
5. Write a Python program to remove a key from a dictionary.
6. Write a Python program to map two lists into a dictionary.
7. Write a Python program to sort a dictionary by key.
8. Write a Python program to get the maximum and minimum value in a dictionary.
9. Write a Python program to get a dictionary from an object's fields.
10. Write a Python program to remove duplicates from Dictionary.
11. Write a Python program to check a dictionary is empty or not.