Dictionary

Dictionary is a data structure in which we store values as a pair of key and value. Each key is separated from its value by a colon (:), and consecutive items are separated by commas. The entire items in a dictionary are enclosed in curly brackets({}).Python dictionary is an unordered collection of items.

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.

Example

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

Create a dictionary by taking userinput

```
squares = {}
print ("Enter your key and integer vlaues")
for x in range(3):
   x=input("Enter your key")
   y=int(input("Enter your value"))
   squares[x] = y
print (squares)
```

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.

Example

```
# dictionary with mixed keys
my_dict = {'name':'John', 1: [2, 4, 3]}
print (my_dict['name'])#John
print (my_dict.get('age'))#None
print (my_dict.get('age',' key is not found'))#key is not found(if the key is exist it will print value)
print (my_dict.get('name',' key is not found'))#'John'
print (my_dict['age'])#Keyerror
```

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.

Example

```
my_dict = {'name': 'John', 1: [2, 4, 3]}

# update value

my_dict['name'] = 'jack'
```

```
print(my_dict)

# add item

my_dict['address'] = 'Downtown'

print(my_dict)

my_dict[1][2]=10

print (my_dict)
```

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.

Example

```
# 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 it is not defined
# print(squares)
```

Iterating Through a Dictionary

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

Example

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

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.

Example

```
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)
```

We can test if a key have that value or not in a sequence by using membership operaor.

Example

```
d={'a':[1,2,3],'b':[4,5,6]}
print 2 in d['a']#Output: True
print 4 in d['a']#Output:False
```

Built-in Functions with Dictionary

all() Return True if all keys of the dictionary are true (or if the dictionary is empty).
Syntax:all(iterable)

any() Return True if any key of the dictionary is true. If the dictionary is empty, return False.
Syntax:any(iterable)

len() Return the length (the number of items) in the dictionary.
Syntax:len(s)

sorted() Return a new sorted list of keys in the dictionary.
Syntax: sorted(iterable[, key][, reverse])

Example

```
import operator
mydict={12:3,19:1,0:0,21:8}
print sorted(mydict)#print the keys in ascending order
print sorted(mydict,reverse=True)#Print the keys in descending order
print sorted(mydict.values())#Values in ascending order
print sorted(mydict.values(),reverse=True)#Values in descending order
print sorted(mydict.items())#sort the keys in ascending order and print keys and values
```

print sorted(mydict.items(),reverse=True)#sort the keys in descending order and print keys and values

print sorted(mydict.items(), key=operator.itemgetter(1))#sort the values in ascending order and print keys and values

print sorted(mydict.items(), key=operator.itemgetter(0))#sort the keys in ascending order and print keys and values

print all(mydict)

print any(mydict)

print len(mydict)

Note:itemgetter(position) returns a callable, that takes a sequence, and returns the value at position in sequence.

Dictionary Methods

values() return a new view of the dictionary's values
Syntax:dictionary.values()

keys() return a new view of the dictionary's keys.
Syntax:dict.keys()

copy() returns a shallow copy of the dictionary. It doesn't modify the original dictionary. The difference between copy method and = operator is When copy() method is used, a new dictionary is created which is filled with a copy of the references from the original dictionary.When = operator is used, a new reference to the original dictionary is created.
Syntax:dict.copy()

Clear() removes all items in the dictionary but not the dictionary itself.It will return an empty dictionary
Syntax:dict.clear()

update([other]) The update() method adds element(s) to the dictionary if the key is not in the dictionary. If the key is in the dictionary, it updates the key with the new value.The update() method takes either a dictionary or an iterable object of key/value pairs (generally tuples).

items() The items() method returns a view object that displays a list of a given dictionary's (key, value) tuple pair.
Syntax:dictionary.items()

pop Remove the item with key and return its value or if key is not found, raises KeyError.
Syntax:dictionary.pop(key[, default])

popitem() Remove and return an arbitary item (the element was not chosen by you it is unpredictable key, value pair). Raises KeyError if the dictionary is empty.
Syntax:dict.popitem()

Example

```
sales = { 'apple': 2, 'orange': 3, 'grapes': 4 }
print sales.values()#print only values
print sales.keys()#print only keys
d1 = {'bananas': 6}
sales.update(d1)# adds element
sales.update({'bananas':7})#update the value
print sales.items()#print key and value pair as a tuple
```

```
salescopy = sales.copy()#copy of the reference
salesoperator=sales#new reference to old one
print salescopy
sales.update({'orange':5})#update the sales dict and salesoperator dict
print sales['orange']
print salescopy['orange']
print salesoperator['orange']
print sales.pop('orange')#Remove the key value pair and return the value
print sales
print sales.popitem()#Remove and return an arbitary item(key,value pair)
print sales
sales.clear()#Clear the elements in a dict and retrun empty dict
print sales
```

Exercise programs

1. Write a Python script to add key to a dictionary?
   Sample Dictionary : {0: 10, 1: 20}
   Expected Result : {0: 10, 1: 20, 2: 30}

2. Write a Python script to concatenate following dictionaries to create a new one?

   Sample Dictionary :
   dic1={1:10, 2:20}
   dic2={3:30, 4:40}
   dic3={5:50,6:60}
   Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

3. Write a Python script to check if a given key already exists in a dictionary?

4. Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys?
Sample Dictionary
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}

5. Python Program to Create a Dictionary with Key as First Character of the string and Value as Words Starting with that Character in that string

6. Write a Python program to count number of items in a dictionary value that is a list.

7. Write a Python program to count of the letters from the string and create a dictionary from a string.

8. Create a dictionary to hold information about pets. Each key is an animal's name, and each value is the animal baby name.
   - For example, 'cat': 'kitten'
- Put at least 3 key-value pairs in your dictionary.
- Use a for loop to print out a series of statements such as "cat baby name is kitten"
- Modify one of the values in your dictionary.
- Add a new key-value pair to your dictionary.
  - Use a for loop to print out a series of statements .
- Remove one of the key-value pairs from your dictionary.
  - Use a for loop to print out a series of statements

10.Create a dictionary of products purchased and their MRPs. Calculate the bill and display to the customer

12.Write a program that has a dictionary of your friends name (as keys) and their birthdays. Print the items in the dictionary in a sorted order. Prompt the user to enter a name and check if it is present in the dictionary. If the name does not exist, then ask the user to enter DOB. Add the details in the dictionary.

13.Write a program tha display a menu and its price. Take the order from the customer. Check if the orderd product is in the mennu. In case it is not there, the customer shpuld be asked to reorder and if it is present, the product should be added in the bill

14.Write a program that prints the maximum and minimum value in a dictionary