Dictionaries in Python

What we will learn?

- Introduction
- List item

Introduction

A dictionary represents a group of elements arranged in the form of key-value pairs.

In the dictionary, the first element is considered as 'key' and immediate next element is taken as its 'value'.

The key and values are separated by colon (:)

All the key-value pairs in a dictionary are inserted in curly braces

Example:

```
dict = { 'Name':'Sam', 'ID':200, 'Salary':20000 }
```

There are three Key - Value pairs in above dictionary.

These pairs are separated by comma.

When the 'Key' is provided, we can get back its values.

Program 1: To create a dictionary with employee details and retrieves the values upon giving the key values

```
# Creating a dictionary with Key - Values pairs
# Create Dictionary with employee details

dict = { 'Name':'Sam', 'ID':200, 'Salary':20000 }

# Access values by giving key
print('Name of the employee=', dict['Name'])
print('Id Number=',dict['ID'])
print('Salary=',dict['Salary'])
```

```
Name of the employee= Sam
Id Number= 200
Salary= 20000
```

Operations on Dictionaries

Indexing and Slicing operaions are not allowed in Dictionary.

Accessing Value: The values associated with Key can be accessed by mentioning Key in square brackets as dict [Keyname]

len ( ) : It returns number of Key Value pairs

Modifying value: Value associated with Key can be modified using assignment operator (=)

Syntax:

```
dict['Keyname'] = <New Value>
```

Adding New Key-Value Pair: New Key-Value pair can be mentioned as

```
dict['New Key'] = <New Value>
```

Deleting Key-Value pair: Existing key-value pair can be deleted using 'del' statement as

```
del dict['Key']
```

Existence of Key-value Pair: Membership operators 'in' and 'not in' can be used to test whether the pair is present or not in dictionary

Note:

1. Keys must be unique. If same key is entered again then old value will be overwritten by new value.
2. Keys must be immutable type. Lists or dictionaries are not allowed as Keys.

```
# Creating Dictionary
d = {'US':'Dollar', "UK":'Pound', 'Japan':'Yen', 'Germany':'Deustch Mark'}

print(d)

# Finding Length of dictionary
print("Length of the dictionary:", len(d))

# Modifying value of Key
d['Germany'] = 'Euro'
print(d)

# Adding new Key-value pair
```

```
# Adding new key-value pair
d['India'] = 'Rupee'
print(d)

# Deleting Key-value pair
del d['UK']
print(d)

# Checking Membership
if 'India' in d:
  print('Indian currency Present')
else:
  print('Indian currency Absent')
```

```
{'US': 'Dollar', 'UK': 'Pound', 'Japan': 'Yen', 'Germany': 'Deustch Mark'}
Length of the dictionary: 4
{'US': 'Dollar', 'UK': 'Pound', 'Japan': 'Yen', 'Germany': 'Euro'}
{'US': 'Dollar', 'UK': 'Pound', 'Japan': 'Yen', 'Germany': 'Euro', 'India': 'Rupee'}
{'US': 'Dollar', 'Japan': 'Yen', 'Germany': 'Euro', 'India': 'Rupee'}
Indian currency Present
```

Dictionary Methods

If d is a dictionary object then

The methods to retrieve or to manipulate content of dictionaries are:

> **d.clear( ):** Remove all key-value pairs from dictionary d

> **d1 = d.copy( ):** Copies all elements from d into a new dictionary d1

> **d1 = d.fromkeys(s, v):** Create new dictionary d1 with keys from sequence 's' and values all set to 'v'

> **d.get(k, v):** Returns the value associated with key 'k'. If key is not found it returns value 'v'

> **d.items( ):** Returns an object that contains key-value pairs of 'd'. The pairs are stored as tuple in the object.

> **d.keys():** Returns a sequence of keys from the dictionary 'd'

> **d.values():** Returns a sequence of values from the dictionary 'd'

> **d.update(x):** Adds all elements from dictionary 'x' to 'd'

> **d.pop(k, v):** Removes the key-value pair identified by key 'k' and returns the value associated with 'k'. If key is not found then return value 'v' and if 'v' is not mentioned then 'KeyError' is raised

> **d.setdefault(k, v):** If key 'k' is found then its value is returned. If Key is not found then 'v' is returned and pair (k, v) is stored into dictionary.

```
# Program to demonstrate use of methods for dictionary
s = ['s1', 's2', 's3' ]

v = 15

# Creating new dictionary using formkeys method
f = d.fromkeys(s,v)
print("New dictionary:",f)

# To get the value associated with key
k = f.get('s1','false')
print("The value associated with s1:", k)

# To remove key-value pair by mentioning key
t = f.pop('s1')
print("The value associated with s1 is:",t)
print("Dictionary after performing pop:",f)

# Use of setdefault method
a=f.setdefault('s2',7)
print("Key s2 is present. The value associated with s2 is:",a)
print("Dictionary f:",f)

b=f.setdefault('s1',7)
print('Key S1 is not present. Hence default value is returned as:',b)
print("Dictionary after appending new pair:",f)
```

```
New dictionary: {'s1': 15, 's2': 15, 's3': 15}
The value associated with s1: 15
The value associated with s1 is: 15
Dictionary after performing pop: {'s2': 15, 's3': 15}
Key s2 is present. The value associated with s2 is: 15
Dictionary f: {'s2': 15, 's3': 15}
Key S1 is not present. Hence default value is returned as: 7
Dictionary after appending new pair: {'s2': 15, 's3': 15, 's1': 7}
```

Program 2: A Python program to retrieve keys, values, and key-value pairs from Dictionary

```
# Create a dictionary with Nation and its currency

d = {'US': 'Dollar', 'UK': 'Pound', 'Japan': 'Yen', 'Germany': 'Euro', 'India': 'Rupee'}

# Print Entire Dictionary
print(d)
```

```
# Display only keys
print("Keys in Dictionary:", d.keys())

# Display only values
print("Values in Dictionary:", d.values())

# Display Key-Value pairs as tuples
print('Items in dict:',d.items())
```

```
{'US': 'Dollar', 'UK': 'Pound', 'Japan': 'Yen', 'Germany': 'Euro', 'India': 'Rupee'}
Keys in Dictionary: dict_keys(['US', 'UK', 'Japan', 'Germany', 'India'])
Values in Dictionary: dict_values(['Dollar', 'Pound', 'Yen', 'Euro', 'Rupee'])
Items in dict: dict_items([('US', 'Dollar'), ('UK', 'Pound'), ('Japan', 'Yen'), ('Germar
```

Program 3: Program to find sum of values in a dictionary

```
# Read the dictionary values using eval() function
M = eval(input('Enter elements in form of Key value pair in { }:'))

# Find sum of values in a dictionary
s = sum(M.values())
print('Sum of the values:',s)
```

```
Enter elements in form of Key value pair in { }:{1:11,2:22,3:33}
Sum of the values: 66
```

Program 4: Program to read the values from Keyboard and update the dictionary

```
# Take an empty dictionary
d = { }

print('How many elements to be inserted in dictionary?')
n = int(input())

for i in range(n):
  k = input("Enter Key:")
  v = int(input('Enter Value:'))    # Assuming value is integer
  d.update({k:v})

# Display the dictionary
print('The Dictionary is:',d)
```

```
How many elements to be inserted in dictionary?
3
Enter Key:1
Enter Value:11
```

```
Enter Key:2
Enter Value:22
Enter Key:3
Enter Value:33
The Dictionary is: {'1': 11, '2': 22, '3': 33}
```

Program 5: Program to create dictionary with cricket player name and their scores in a match. Also retrieve runs by entering player's name

```
# Create Empty Dictionary
player = { }

n = int(input("How many players?"))

for i in range(n):
  name = input("Enter Player's Name:")
  runs = int(input("Enter Runs:"))
  player.update({name:runs})

# Display only Players name
print('Players in this match:')
for pname in player.keys():
  print(pname, end = '\t')

print()

# Accept Player's name to find number of runs scored by him
p = input('Enter player Name to find his score:')
r = player.get(p,-1)

if(r == -1):
  print('Name not found')
else:
  print('Runs scored by ',p,'is ',r)
```

```
How many players?3
Enter Player's Name:abc
Enter Runs:54
Enter Player's Name:xyz
Enter Runs:68
Enter Player's Name:def
Enter Runs:99
Players in this match:
abc     xyz     def
Enter player Name to find his score:def
Runs scored by  def is  99
```

Using for Loop with Dictionaries

For loop is convenient way to retrieve the elements of a dictionary

Program 6: Program to show usage of for loop to retrieve elements of dictionary

```
# Using for loop with dictionaries
color = {'r':"Red", 'g':"Green", 'b':"Blue", 'w':"white"}

# Display only keys
# Here k is holding all the KEYS
for k in color:
  print(k)

# Display only values
for k in color:
  print(color[k])

# items() method returns key and value pair into k, v
for k, v in color.items():
  print('Key = {} Value = {}'.format(k, v))
```

```
r
g
b
w
Red
Green
Blue
white
Key = r Value = Red
Key = g Value = Green
Key = b Value = Blue
Key = w Value = white
```

Program 7: Program to find the number of occurrences of each letter in a string using dictionary

```
# Take input string from the user
st = input("Enter the string:")

# Take an empty dictionary
c = { }

# Store each letter of string as Key and number of occurences as its value
for i in st:
  c[i] = c.get(i, 0) + 1

# Display key-value pair of dictionary
for k,v in c.items( ):
  print("Key = {} \t Its occurrences = {}".format(k,v))
```

```
Enter the string:python programming
Key = p          Its occurrences = 2
Key = y          Its occurrences = 1
Key = t          Its occurrences = 1
Key = h          Its occurrences = 1
Key = o          Its occurrences = 2
Key = n          Its occurrences = 2
Key =            Its occurrences = 1
Key = r          Its occurrences = 2
Key = g          Its occurrences = 2
Key = a          Its occurrences = 1
Key = m          Its occurrences = 2
Key = i          Its occurrences = 1
```

Sorting the elements of Dictionary using Lambda

sorted( ) function can be used to sort the dictionary.

By default, the elements are sorted in ascending order.

The format of sorted function is

```
sorted(elements, key = <function> )
```

elements of the dictionary can be accessed using method d.items()

key can be assigned with lambda function which will determine whether data to be sorted using Keys or Values.

Following function will consider **keys** for sorting the elements

```
key = lambda t : t[0]
```

Following function will consider **values** for sorting the elements

```
key = lambda t : t[1]
```

Program 8: Program to sort the elements of dictionary based on key or value

```
# Sorting elements of the dictionary based on key or value
color = { 10:"Red", 35:"Green", 15:"Blue", 25:"White"}
print('Given Dictionary:',color)

# Sort the dictionary by Key i.e. )th element of the pair
c1 = sorted(color.items(), key = lambda k :k[0] )
```

```
print("Sorted Dictionary by Keys:",c1)

# Sort the dictionary by value i.e. )th element of the pair
c2 = sorted(color.items(), key = lambda k :k[1] )
print("Sorted Dictionary by Values:",c2)
```

```
Given Dictionary: {10: 'Red', 35: 'Green', 15: 'Blue', 25: 'White'}
Sorted Dictionary by Keys: [(10, 'Red'), (15, 'Blue'), (25, 'White'), (35, 'Green')]
Sorted Dictionary by Values: [(15, 'Blue'), (35, 'Green'), (10, 'Red'), (25, 'White')]
```

Converting Lists into Dictionary

If there are two lists then it is possible to convert them into dictionary.

Steps to be followed:

1. To create a zip class object by passing the two lists to zip( ) function
2. Convert zip object into dictionary by using dict( ) function

Program 9: Program to convert the elements of two lists into key-value pairs of dictionary

```
del dict
# Take two separate lists with elements
States = ['Maharashtra', 'Karnataka', 'Madhya Pradesh', 'Telangana']
Cities = ['Mumbai','Bengaluru','Bhopal','Hyderabad']

# Make Dictionary
z = zip(States,Cities)
d = dict(z)

# Display key-value pairs from dictionary d
print('{:15s} --- {:s}'.format('State','Capital'))
for i in d:
  print('{:15s} --- {:s}'.format(i,d[i]))
```

```
State           --- Capital
Maharashtra     --- Mumbai
Karnataka       --- Bengaluru
Madhya Pradesh  --- Bhopal
Telangana       --- Hyderabad
```

Converting Strings into Dictionary

When a string is given with key and value pairs separated by some delimiter (or separator) like comma, we can convert the string into dictionary.

There are three steps:

1. Split the string using separator so that we can get all pieces of string.
2. Store all the pieces into the list
3. Convert the list into dictionary

Program 10: Program to convert a string into key-value pairs and store them into dictionary

```
st = "Vijay=23,Ganesh=21,Lakshmi=29,Nikhil=35"

lst = [ ]

# Splitting the string at',' and '='
# Storing the pieces into list
for k in st.split(','):
  part = k.split('=')
  lst.append(part)

print(lst)

# Converting the list into dictionary
d = dict(lst)
print(d)

# Creating new dictionary to store the ages as an integer values
d1 = { }
for (k,v) in d.items():
  d1[k] = int(v)

# Display final dictionary
print(d1)
```

```
[['Vijay', '23'], ['Ganesh', '21'], ['Lakshmi', '29'], ['Nikhil', '35']]
{'Vijay': '23', 'Ganesh': '21', 'Lakshmi': '29', 'Nikhil': '35'}
{'Vijay': 23, 'Ganesh': 21, 'Lakshmi': 29, 'Nikhil': 35}
```

Passing Dictionaries to Function

We can pass a dictionary to a function by passing the name of the dictionary.

Program 11: Program to define a function that accepts dictionary as parameter

```
# Function that accepts dictionary object as parameter
def sample(s):
  for i, j in s.items():
```

```
    print(i,'---',j)

# Create a dictionary
d = {'a':'apple', 'b':'book', 'c':'cat'}

# pass dictionary object to function
sample(d)
```

```
a --- apple
b --- book
c --- cat
```

Ordered Dictionaries

It is a dictionary but it will keep the order of the elements.

The elements are stored and maintained in same order as they were entered into the ordered dictionary.

It can be created using the **OrderedDict( )** method of **Collection** module

Program 12: Program to create a dictionary that does not change the order of elements

```
# Create an ordered dictionary

from collections import OrderedDict

# d is ordered dictionary
d = OrderedDict( )
d[10] = 'A'
d[11] = 'B'
d[12] = 'C'
d[1] = 'D'

# Display the ordered dictionary
for i,j in d.items():
  print(i,j)
```

```
10 A
11 B
12 C
1 D
```

✓ 0s completed at 00:29