Dictionary_VHA

November 1, 2025

1 Dictionaries

2 Introduction

A dictionary is entirely different for Python as it is not a sequence but a mapping. A mapping is the same as an object collection, but it stores objects using keys instead of relative positions. Moreover, mappings don't maintain any reliable left-to-right order; they map keys to the associated values.

3 Dictionary in Python is a collection of keys values, used to store data values like a map, which, unlike other data types which hold only a single value as an element.

In some languages it is known as map or assosiative arrays.

dict = { 'name' : 'nitish' , 'age' : 33 , 'gender' : 'male' }

Characterstics:

Mutable

Indexing has no meaning

keys can't be duplicated

keys can't be mutable items

4 We will cover the following topics in this chapter:

Creating a dictionary

Accessing the elements in a dictionary

Updating a dictionary

Deleting dictionary elements

Built-in dictionary functions

5 Creating a dictionary

Curly brackets represent dictionary entries. Key-value pairs are separated by a colon in the dictionary.

An empty dictionary without any items is written with just two curly braces, i.e.,

[3]:
```
# empty dictionary
d = {}
d
```

[3]:
```
{}
```

[4]:
```
# 1D dictionary
d1 = { 'name' : 'nitish' ,'gender' : 'male' }
d1
```

[4]:
```
{'name': 'nitish', 'gender': 'male'}
```

[5]:
```
# with mixed keys
d2 = {(1,2,3):1,'hello':'world'}
d2
```

[5]:
```
{(1, 2, 3): 1, 'hello': 'world'}
```

[6]:
```
d2 = {[1,2,3]:1,'hello':'world'}
d2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_25972\3934894962.py in <module>
----> 1 d2 = {[1,2,3]:1,'hello':'world'}
      2 d2

TypeError: unhashable type: 'list'
```

[7]:
```
# 2D dictionary -> JSON
s = {
    'name':'nitish',
     'college':'bit',
     'sem':4,
     'subjects':{
         'dsa':50,
         'maths':67,
         'english':34
     }
}
s
```

[7]:
```
{'name': 'nitish',
 'college': 'bit',
 'sem': 4,
 'subjects': {'dsa': 50, 'maths': 67, 'english': 34}}
```

[8]:
```
# using sequence and dict function
d4 = dict([('name','nitish'),('age',32),(3,3)])
d4
```

[8]:
```
{'name': 'nitish', 'age': 32, 3: 3}
```

[9]:
```
d5 = {'name':'nitish','name':'rahul'}
d5
```

[9]:
```
{'name': 'rahul'}
```

[10]:
```
# mutable items as keys
d6 = {'name':'nitish',(1,2,3):2}
print(d6)
```

```
{'name': 'nitish', (1, 2, 3): 2}
```

6 Accessing the elements in a dictionary

You use the square brackets along with the key to access the elements of a dictionary.

The get() method returns a value for the given key; it returns default value None if the key is not available.

[11]:
```
my_dict = {'name': 'Jack', 'age': 26}
[]
```

[11]:
```
[]
```

[12]:
```
my_dict = {'name': 'Jack', 'age': 26}
my_dict['age']
```

[12]:
```
26
```

[14]:
```
my_dict = {'name': 'Jack', 'age': 26}

my_dict['age']

print(my_dict.get('age'))
```

```
26
```

[15]:
```
s = {
    'name':'nitish',
```

```
    'college':'bit',
    'sem':4,
    'subjects':{
        'dsa':50,
        'maths':67,
        'english':34
    }
}
s['subjects']['maths']
```

[15]: 67

7 Updating a dictionary

Dictionary entries can be changed. We can add new items or change the values of the existing ones using the assignment operator. The value is updated if the key already exists; otherwise, a new key-value pair is added to the dictionary.

8 Adding key-value pair

[17]:
```
d4 = dict([('name','nitish'),('age',32),(3,3)])
d4['gender'] = 'male'
d4
d4['weight'] = 72
d4
```

[17]: {'name': 'nitish', 'age': 32, 3: 3, 'gender': 'male', 'weight': 72}

[18]:
```
s = {
    'name':'nitish',
    'college':'bit',
    'sem':4,
    'subjects':{
        'dsa':50,
        'maths':67,
        'english':34
    }
}
s['subjects']['ds'] = 75
s
```

[18]: {'name': 'nitish',
 'college': 'bit',
 'sem': 4,
 'subjects': {'dsa': 50, 'maths': 67, 'english': 34, 'ds': 75}}

9 Deleting dictionary elements

The pop() method is used to remove an item from a dictionary. It returns the value of an item removed with the provided key. You can use popitem() to remove and return an arbitrary item (key, value) from a dictionary. The clear() method is used to remove all items at once. The del keyword can also be used to remove individual items or the entire dictionary.

pop

popitem

del

clear

[20]:
```
d = {'name': 'nitish', 'age': 32, 3: 3, 'gender': 'male', 'weight': 72}
# pop
d.pop(3)
print(d)
```

```
{'name': 'nitish', 'age': 32, 'gender': 'male', 'weight': 72}
```

[21]:
```
d = {'name': 'nitish', 'age': 32, 3: 3, 'gender': 'male', 'weight': 72}
# popitem
d.popitem()
d.popitem()
print(d)
```

```
{'name': 'nitish', 'age': 32, 3: 3}
```

[22]:
```
d = {'name': 'nitish', 'age': 32, 3: 3, 'gender': 'male', 'weight': 72}
# del
del d['name']
print(d)
```

```
{'age': 32, 3: 3, 'gender': 'male', 'weight': 72}
```

[23]:
```
d = {'name': 'nitish', 'age': 32, 3: 3, 'gender': 'male', 'weight': 72}
# clear
d.clear()
print(d)
```

```
{}
```

[24]:
```
s = {
    'name':'nitish',
     'college':'bit',
     'sem':4,
     'subjects':{
         'dsa':50,
         'maths':67,
         'english':34
```

```
    }
}
del s['subjects']['maths']
s
```

[24]:
```
{'name': 'nitish',
 'college': 'bit',
 'sem': 4,
 'subjects': {'dsa': 50, 'english': 34}}
```

10 Properties of dictionary keys

There are no restrictions on dictionary values. A Python object can either be a standard object or a user-defined object, but this is not true for dictionary keys.

The two important properties of keys are as follows:

There will be no more than one entry per key, so duplicate keys are not allowed. If duplicate keys are encountered during assignment, the last assignment wins.

A key must be immutable type, such as a dictionary key. You can use strings, numbers, or tuples, but a list, such as [‘key’], is an invalid key.

11 Editing key-value pair

[25]:
```
s = {
    'name':'nitish',
     'college':'bit',
     'sem':4,
     'subjects':{
         'dsa':50,
         'maths':67,
         'english':34
     }
}
s['subjects']['dsa'] = 80
s
```

[25]:
```
{'name': 'nitish',
 'college': 'bit',
 'sem': 4,
 'subjects': {'dsa': 80, 'maths': 67, 'english': 34}}
```

12 Dictionary Operations

Membership

Iteration

[26]:
```
s = {
    'name':'nitish',
     'college':'bit',
     'sem':4,
     'subjects':{
         'dsa':50,
         'maths':67,
         'english':34
     }
}
"name" in s
```

[26]: True

[27]:
```
s = {
    'name':'nitish',
     'college':'bit',
     'sem':4,
     'subjects':{
         'dsa':50,
         'maths':67,
         'english':34
     }
}
"nitish" in s
```

[27]: False

[28]:
```
d = {'name':'nitish','gender':'male','age':33}

for i in d:
  print(i,d[i])
```

```
name nitish
gender male
age 33
```

13 Dictionary function

len

sorted

max

min

items

keys

values

update

formkeys

setdefualt

[30]:
```
d = {'name':'nitish','gender':'male','age':33}
len(d)
```

[30]: 3

[31]:
```
d = {'name':'nitish','gender':'male','age':33}
sorted(d)
```

[31]: ['age', 'gender', 'name']

[32]:
```
d = {'name':'nitish','gender':'male','age':33}
sorted(d,reverse=True)
```

[32]: ['name', 'gender', 'age']

[33]:
```
d = {'name':'nitish','gender':'male','age':33}
max(d)
```

[33]: 'name'

[34]:
```
d = {'name':'nitish','gender':'male','age':33}
min(d)
```

[34]: 'age'

[36]:
```
d = {'name':'nitish','gender':'male','age':33}
print(d.keys())
for i in d.keys():
    print(i)
```

```
dict_keys(['name', 'gender', 'age'])
name
gender
age
```

[38]:
```
d = {'name':'nitish','gender':'male','age':33}
print(d.values())
for i in d.values():
    print(i)
```

```
dict_values(['nitish', 'male', 33])
nitish
```

```
male
33
```

[40]:
```
d = {'name':'nitish','gender':'male','age':33}
print(d.items())
for i,j in d.items():
    print(i,j)
```

```
dict_items([('name', 'nitish'), ('gender', 'male'), ('age', 33)])
name nitish
gender male
age 33
```

[41]:
```
# update
d1 = {1:2,3:4,4:5}
d2 = {4:7,6:8}

d1.update(d2)
print(d1)
```

```
{1: 2, 3: 4, 4: 7, 6: 8}
```

14 setdefault()

The setdefault() method is similar to but it will set if the key is not already in dict.

Syntax: dict.setdefault(key, default=None)

Parameters

key – The key to search

Default - The value to be returned if the key cannot be found

Return value The method returns the value of the given key if it is present in the dictionary; it returns the default value if it isn't present.

[43]:
```
d = {'name':'nitish','gender':'male','age':33}
d.setdefault("age",55)
print(d)
```

```
{'name': 'nitish', 'gender': 'male', 'age': 33}
```

[44]:
```
d = {'name':'nitish','gender':'male','age':33}
d.setdefault("year",55)
print(d)
```

```
{'name': 'nitish', 'gender': 'male', 'age': 33, 'year': 55}
```

15 dict.fromkeys()

[45]:
```
x = ('key1', 'key2', 'key3')
y = 0

thisdict = dict.fromkeys(x, y)

print(thisdict)
```

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

[48]:
```
d = {'name':'nitish','gender':'male','age':33}
d=d.fromkeys("vishal",3)
print(d)
```

```
{'v': 3, 'i': 3, 's': 3, 'h': 3, 'a': 3, 'l': 3}
```

16 Dictionary Comprehension

17 What is Dictionary Comprehension?

- Just like list comprehension, dictionary comprehension provides a concise way to create dictionaries. It allows you to generate a dictionary from iterables — applying expressions for keys and values (and optionally filtering them).
- {key_expression: value_expression for item in iterable if condition}

[50]:
```
# print 1st 10 numbers and their squares
{i:i**2 for i in range(1,11)}
```

[50]:
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

[51]:
```
# using existing dict
distances = {'delhi':1000,'mumbai':2000,'bangalore':3000}
{key:value*0.62 for (key,value) in distances.items()}
```

[51]:
```
{'delhi': 620.0, 'mumbai': 1240.0, 'bangalore': 1860.0}
```

[52]:
```
# using zip
days = ["Sunday", "Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"]
temp_C = [30.5,32.6,31.8,33.4,29.8,30.2,29.9]

{i:j for (i,j) in zip(days,temp_C)}
```

[52]:
```
{'Sunday': 30.5,
 'Monday': 32.6,
 'Tuesday': 31.8,
 'Wednesday': 33.4,
 'Thursday': 29.8,
```

```
 'Friday': 30.2,
 'Saturday': 29.9}
```

[53]:
```
# using if condition
products = {'phone':10,'laptop':0,'charger':32,'tablet':0}

{key:value for (key,value) in products.items() if value>0}
```

[53]:
```
{'phone': 10, 'charger': 32}
```

[54]:
```
# Nested Comprehension
# print tables of number from 2 to 4
{i:{j:i*j for j in range(1,11)} for i in range(2,5)}
```

[54]:
```
{2: {1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18, 10: 20},
 3: {1: 3, 2: 6, 3: 9, 4: 12, 5: 15, 6: 18, 7: 21, 8: 24, 9: 27, 10: 30},
 4: {1: 4, 2: 8, 3: 12, 4: 16, 5: 20, 6: 24, 7: 28, 8: 32, 9: 36, 10: 40}}
```

18 Key with maximum unique values

Given a dictionary with values list, extract key whose value has most unique values.

Example 1:

Input:

test_dict = {“CampusX” : [5, 7, 9, 4, 0], “is” : [6, 7, 4, 3, 3], “Best” : [9, 9, 6, 5, 5]}

Output:

CampusX

Example 2:

Input:

test_dict = {“CampusX” : [5, 7, 7, 7, 7], “is” : [6, 7, 7, 7], “Best” : [9, 9, 6, 5, 5]}

Output:

Best

[55]:
```
# write your code here
test_dict = {"CampusX" : [5, 7, 7, 7, 7], "is" : [6, 7, 7, 7], "Best" : [9, 9,␣
 ↪6, 5, 5]}

max_val = 0
max_key = ''
for i in test_dict:
  if max_val < len(set(test_dict[i])):
    max_val = len(set(test_dict[i]))
    max_key = i
```

```
print(max_key)
```

```
Best
```

19 Replace words from Dictionary. Given String, replace it's words from lookup dictionary.

Example 1:

Input:

test_str = 'CampusX best for DS students.'

repl_dict = {"best" : "is the best channel", "DS" : "Data-Science"}

Output:

CampusX is the best channel for Data-Science students.

Example 2:

Input:

test_str = 'CampusX best for DS students.'

repl_dict = {"good" : "is the best channel", "ds" : "Data-Science"}

Output:

CampusX best for DS students.

[56]:
```
# write your code here
test_str = 'CampusX best for DS students.'
repl_dict = {"best" : "is the best channel", "DS" : "Data-Science"}

res = []
for i in test_str.split():
  if i in repl_dict:
    res.append(repl_dict[i])
  else:
    res.append(i)

print(" ".join(res))
```

```
CampusX is the best channel for Data-Science students.
```

20 Convert List to List of dictionaries. Given list values and keys list, convert these values to key value pairs in form of list of dictionaries.

Example 1:

Input:

test_list = [“DataScience”, 3, “is”, 8]

key_list = [“name”, “id”]

Output:

[{‘name’: ‘DataScience’, ‘id’: 3}, {‘name’: ‘is’, ‘id’: 8}]

Example 2:

Input:

test_list = [“CampusX”, 10]

key_list = [“name”, “id”]

Output:

[{‘name’: ‘CampusX’, ‘id’: 10}]

[57]:
```
# write your code here
test_list = ["CampusX", 10]
key_list = ["name", "id"]

n = len(test_list)

res = []

for i in range(0,n,2):
  res.append({key_list[0]: test_list[i],key_list[1]:test_list[i+1]})

print(res)
```

```
[{'name': 'CampusX', 'id': 10}]
```

21 Convert a list of Tuples into Dictionary.

Example 1:

Input:

[(“akash”, 10), (“gaurav”, 12), (“anand”, 14), (“suraj”, 20), (“akhil”, 25), (“ashish”, 30)]

Output:

{‘akash’: [10], ‘gaurav’: [12], ‘anand’: [14], ‘suraj’: [20], ‘akhil’: [25], ‘ashish’: [30]}

Example 2:

Input:

[(‘A’, 1), (‘B’, 2), (‘C’, 3)]

Output:

{‘A’: [1], ‘B’: [2], ‘C’: [3]}

[58]:
```
# write your code here
L1 = [("akash", 10), ("gaurav", 12), ("anand", 14), ("suraj", 20), ("akhil",␣
↪25), ("ashish", 30)]

L = [('A', 1), ('B', 2), ('C', 3)]

d = {}

for i,j in L:
  d[i] = [j]

print(d)
```

```
{'A': [1], 'B': [2], 'C': [3]}
```

22 Sort Dictionary key and values List.

Example 1:

Input:

{‘c’: [3], ‘b’: [12, 10], ‘a’: [19, 4]}

Output:

{‘a’: [4, 19], ‘b’: [10, 12], ‘c’: [3]}

Example 2:

Input:

{‘c’: [10, 34, 3]}

Output:

{‘c’: [3, 10, 34]}

[1]:
```
# write your code here

d = {'c': [3], 'b': [12, 10], 'a': [19, 4]}

res = {}

for i in sorted(d):
  res[i] = sorted(d[i])

print(res)
```

```
{'a': [4, 19], 'b': [10, 12], 'c': [3]}
```

23 write a python program to Find how many words start with certain letter in a string

[67]:
```
s="acd bcd avf ght dfg bgh kju avf ghv bvf ujh kju bgh avf nhj bgh bvf cdf avg␣
 ↪sde kju gty frt der cdf xsd zse"
d={}
for i in s.split():
    d[i[0]]=d.get(i[0],0)+1
print(d)
```

```
{'a': 5, 'b': 6, 'g': 3, 'd': 2, 'k': 3, 'u': 1, 'n': 1, 'c': 2, 's': 1, 'f': 1,
'x': 1, 'z': 1}
```

24 The code you've provided aims to create a dictionary where the keys are the first letters of the words in the string s, and the values are lists containing the words that start with that particular letter.

[73]:
```
s="acd bcd avf ght dfg bgh kju avf ghv bvf ujh kju bgh avf nhj bgh bvf cdf avg␣
 ↪sde kju gty frt der cdf xsd zse"
d={}
for i in s.split():
    if i[0] not in d.keys():
        d[i[0]]=[]
        d[i[0]].append(i)
    else:
        d[i[0]].append(i)
print(d)
```

```
{'a': ['acd', 'avf', 'avf', 'avf', 'avg'], 'b': ['bcd', 'bgh', 'bvf', 'bgh',
'bgh', 'bvf'], 'g': ['ght', 'ghv', 'gty'], 'd': ['dfg', 'der'], 'k': ['kju',
'kju', 'kju'], 'u': ['ujh'], 'n': ['nhj'], 'c': ['cdf', 'cdf'], 's': ['sde'],
'f': ['frt'], 'x': ['xsd'], 'z': ['zse']}
```

25 I have a string of words. I would like to sort the list by the length of each word so that the longest word is at the top

[85]:
```
s="nbhg mjknh kiuytgft nbh nbhgtygnm mnjku lkmnhg nbh nb ghyuytredf bghyyt␣
 ↪kmnjkmnjkmn bghbghb vfgbvc hgv bvg"
d={}
for i in s.split():
    d[i]=len(i)
#print(d)
l1=d.values()
l2=d.keys()
```

```
#print(l1)
#print(l2)
ans=[i[1] for i in sorted(list(zip(l1,l2)),reverse=True)]
print(ans)
```

```
['kmnjkmnjkmn', 'ghyuytredf', 'nbhgtygnm', 'kiuytgft', 'bghbghb', 'vfgbvc',
'lkmnhg', 'bghyyt', 'mnjku', 'mjknh', 'nbhg', 'nbh', 'hgv', 'bvg', 'nb']
```