# Python Dictionary

Setiap kunci dipisahkan dari nilainya oleh titik dua (:), item dipisahkan oleh koma, dan semuanya tertutup dalam kurung kurawal. Kamus kosong tanpa barang ditulis hanya dengan dua kurung kurawal, seperti ini: { }.
Kunci unik dalam kamus sementara nilai mungkin tidak. Nilai kamus bisa berupa tipe apa pun, namun kunci harus berupa tipe data yang tidak berubah seperti string, angka, atau tupel.

## Accessing Values in Dictionary:

Untuk mengakses elemen kamus, Anda dapat menggunakan tanda kurung siku yang sudah dikenal bersama dengan kunci untuk mendapatkan nilainya. Berikut adalah contoh sederhana -

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First' }

print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut -

```
dict['Name']:  Zara
dict['Age']:  7
```

Jika kita mencoba mengakses item data dengan sebuah kunci, yang bukan bagian dari kamus, kita mendapatkan error sebagai berikut -

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First' }

print "dict['Alice']: ", dict['Alice']
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut -

```
dict['Alice']:
Traceback (most recent call last):
  File "test.py", line 4, in <module>
      print "dict['Alice']: ", dict['Alice'];
KeyError: 'Alice'
```

## Updating Dictionary

Anda dapat memperbarui kamus dengan menambahkan entri baru atau pasangan nilai kunci, memodifikasi entri yang ada, atau menghapus entri yang ada seperti yang ditunjukkan di bawah ini dalam contoh sederhana -

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First' }

dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry


print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut -

```
dict['Age']:  8
dict['School']:  DPS School
```

## Delete Dictionary Elements

Anda dapat menghapus elemen kamus individual atau menghapus keseluruhan isi kamus. Anda juga dapat menghapus seluruh kamus dalam satu operasi.

Untuk menghapus seluruh kamus secara eksplisit, cukup gunakan del statement. Berikut adalah contoh sederhana –

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First' }

del dict['Name']; # remove entry with key 'Name'
dict.clear();     # remove all entries in dict
del dict ;        # delete entire dictionary

print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

Ini menghasilkan hasil berikut. Perhatikan bahwa pengecualian diajukan karena setelah kamus del dict tidak ada lagi -

```
dict['Age']:
Traceback (most recent call last):
  File "test.py", line 8, in <module>
                print "dict['Age']: ", dict['Age'];
TypeError: 'type' object is unsubscriptable
```

**Note:** del () metode dibahas di bagian selanjutnya.

## Properties of Dictionary Keys

**Nilai kamus tidak memiliki batasan. Mereka bisa menjadi objek Python yang sewenang-wenang, baik objek standar atau objek yang ditentukan pengguna. Namun, hal yang sama tidak berlaku untuk kunci.**

**Ada dua hal penting yang perlu diingat tentang kunci kamus –**

**Lebih dari satu entri per kunci tidak diperbolehkan. Yang berarti tidak ada kunci duplikat yang diperbolehkan. Ketika kunci duplikat ditemui selama penugasan, tugas terakhir akan menang. Sebagai contoh –**

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni' }

print "dict['Name']: ", dict['Name']
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut -

```
dict['Name']:  Manni
```

(b) Tombol harus tidak berubah. Yang berarti Anda bisa menggunakan string, angka atau tupel sebagai tombol kamus tapi sesuatu seperti ['key'] tidak diperbolehkan. Berikut adalah contoh sederhana:

```
#!/usr/bin/python

dict = {['Name']: 'Zara', 'Age': 7 }

print "dict['Name']: ", dict['Name']
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut -

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    dict = {['Name']: 'Zara', 'Age': 7 };
TypeError: list objects are unhashable
```

## Built-in Dictionary Functions  Methods

Python includes the following dictionary functions −

| SN | Function with Description |
|---|---|
| 1 | cmp(dict1, dict2) Compares elements of both dict. |
| 2 | len(dict) Gives the total length of the dictionary. This would be equal to the number of items in the dictionary. |
| 3 | str(dict) Produces a printable string representation of a dictionary |
| 4 | type(variable) Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type. |

Python includes following dictionary methods −

| SN | Methods with Description |
|---|---|
| 1 | dict.clear() Removes all elements of dictionary *dict* |
| 2 | dict.copy() Returns a shallow copy of dictionary *dict* |
| 3 | dict.fromkeys() Create a new dictionary with keys from seq and values *set* to *value*. |
| 4 | dict.get(key, default=None) For *key* key, returns value or default if key not in dictionary |
| 5 | dict.has _key(key) Returns *true* if key in dictionary *dict*, *false* otherwise |
| 6 | dict.items() Returns a list of *dict*'s (key, value) tuple pairs |
| 7 | dict.keys() Returns list of dictionary dict's keys |
| 8 | dict.setdefault(key, default=None) Similar to get(), but will set dict[key]=default if *key* is not already in dict |
| 9 | dict.update(dict2) Adds dictionary *dict2*'s key-values pairs to *dict* |
| 10 | dict.values() Returns list of dictionary *dict*'s values |

## Dictionaries

### Introductions

Kami sudah mengenal daftar di bab sebelumnya. Di bab kursus Python online kami, kami akan mempresentasikan kamus dan operator dan metode pada kamus. Program atau skrip Python tanpa daftar dan kamus hampir tidak dapat dibayangkan. Seperti daftar kamus yang bisa dengan mudah diubah, bisa menyusut dan berkembang ad libitum pada saat run time. Mereka menyusut dan tumbuh tanpa perlu membuat salinan. Kamus dapat dimuat dalam daftar dan sebaliknya. Tapi apa perbedaan antara daftar dan kamus? Daftar diurutkan dari objek, sedangkan kamus tidak berurutan. Tapi perbedaan utamanya adalah item dalam kamus diakses melalui kunci dan tidak melalui posisinya. Kamus adalah array asosiatif (juga dikenal sebagai hash). Kunci kamus mana pun dikaitkan (atau dipetakan) ke sebuah nilai. Nilai kamus bisa berupa tipe data Python. Jadi kamus adalah pasangan kunci-nilai tak berurutan.

Kamus tidak mendukung urutan operasi dari jenis data urutan seperti string, tupel dan daftar. Kamus termasuk tipe pemetaan built-in. Mereka adalah satu-satunya wakil semacam ini!

Di akhir bab ini, kami akan menunjukkan bagaimana kamus dapat diubah menjadi satu daftar, berisi (kunci, nilai) -tupel atau dua daftar, yaitu satu dengan kunci dan satu dengan nilainya. Transformasi ini bisa dilakukan secara terbalik juga.

## How to create a dictionary?

Membuat kamus sama mudahnya dengan menempatkan item dalam kurung kurawal { } dipisahkan dengan koma. Item memiliki kunci dan nilai yang sesuai dinyatakan sebagai pasangan, kunci: nilai. Sementara nilai dapat berupa tipe data apa pun dan dapat diulang, kunci harus terdiri dari tipe yang tidak dapat diubah (string, number atau tupel dengan elemen yang tidak berubah) dan harus unik.

```
# 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')])
```

Seperti yang bisa Anda lihat di atas, kita juga bisa membuat kamus menggunakan fungsi built-in dict ().

## How to access elements from a dictionary?

Sementara pengindeksan digunakan dengan jenis wadah lain untuk mengakses nilai, kamus menggunakan tombol. Kunci dapat digunakan baik di dalam tanda kurung siku atau dengan metode get (). Perbedaan saat menggunakan get () adalah mengembalikan Elemen alih-alih KeyError, jika kuncinya tidak ditemukan.

```
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']
```

Saat menjalankan program, hasilnya adalah:

```
Jack
26
```

## How to change or add elements in a dictionary?

Kamus bisa berubah-ubah. Kita bisa menambahkan item baru atau mengubah nilai barang yang ada menggunakan operator penugasan.

Jika kuncinya sudah ada, nilai akan diperbarui, jika ada kunci baru: pasangan nilai ditambahkan ke kamus.

Script.py

```
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).
```

Saat menjalankan program, hasilnya adalah:
{'name': 'Jack', 'age': 27 }
{'name': 'Jack', 'age': 27, 'address': 'Downtown' }

## How to delete or remove elements from a dictionary?

Kita bisa menghapus item tertentu dalam kamus dengan menggunakan metode pop (). Metode ini menghilangkan item dengan tombol yang disediakan dan mengembalikan nilainya.
Metodenya, popitem () dapat digunakan untuk menghapus dan mengembalikan item yang sewenang-wenang (key, value) membentuk kamus. Semua item dapat dihapus sekaligus dengan menggunakan metode clear ().
Kita juga bisa menggunakan kata kunci del untuk menghapus setiap item atau keseluruhan kamus itu sendiri.

Scrip.py

```
# 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 }
{ }
```