# LISTS

Struktur data yang paling dasar dengan Python adalah urutannya. Setiap elemen berurutan diberi nomor - posisinya atau indeksnya. Indeks pertama adalah nol, indeks kedua adalah satu, dan seterusnya.

Python memiliki enam jenis urutan built-in, namun yang paling umum adalah daftar dan tupel, yang akan kami lihat di tutorial ini.

Ada beberapa hal yang dapat Anda lakukan dengan semua tipe urutan. Operasi ini meliputi pengindeksan, pengiris, penambahan, perbanyak, dan pengecekan keanggotaan. Selain itu, Python memiliki fungsi built-in untuk menemukan panjang urutan dan untuk menemukan elemen terbesar dan terkecilnya.

Daftar Python

Daftar ini adalah datatype paling serbaguna yang tersedia dengan Python yang dapat ditulis sebagai daftar nilai yang dipisahkan koma (item) antara tanda kurung siku. Hal penting tentang daftar adalah item dalam daftar tidak perlu jenis yang sama.

Membuat daftar sesederhana memasukkan berbagai nilai yang dipisahkan koma di antara tanda kurung siku. Misalnya -

```
list1 = ['physics', 'chemistry', 1997, 2000];


list2 = [1, 2, 3, 4, 5 ];

list3 = ["a", "b", "c", "d"]
```

Serupa dengan indeks string, daftar indeks mulai dari 0, dan daftar dapat diiris, digabungkan dan seterusnya.

Mengakses Nilai dalam Daftar

Untuk mengakses nilai dalam daftar, gunakan tanda kurung siku untuk mengiris beserta indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut. Misalnya -

```
#! / Usr / bin / python
```

List1 = ’fisika’, ’kimia’, 1997, 2000];

List2 = [1, 2, 3, 4, 5, 6, 7];

Cetak "list1 [0]:", list1 [0]

Cetak "list2 [1: 5]:", list2 [1: 5]

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut -

List1 [0]: fisika

List2 [1: 5]: [2, 3, 4, 5]

Memperbarui Daftar

Anda dapat memperbarui satu atau beberapa elemen daftar dengan memberikan potongan di sisi kiri operator penugasan, dan Anda dapat menambahkan ke elemen dalam daftar dengan metode append (). Misalnya -

#! / Usr / bin / python

list = ’physics’, ’chemistry’, 1997, 2000];

print "Value available at index 2 : "

print list[2]

list[2] = 2001;

print "New value available at index 2 : "

print list[2]

Catatan: append () metode dibahas di bagian selanjutnya.

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut -

Nilai tersedia di indeks 2:

1997

Nilai baru tersedia di indeks 2:

2001

Hapus Daftar Elemen

Untuk menghapus elemen daftar, Anda dapat menggunakan salah satu pernyataan del jika Anda tahu persis elemen yang Anda hapus atau metode hapus () jika Anda tidak mengetahuinya. Misalnya -

#! / Usr / bin / python

List1 = [’fisika’, ’kimia’, 1997, 2000];

Daftar cetak1

Del list1 [2];

Cetak "Setelah menghapus nilai pada indeks 2:"

Daftar cetak1

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut -

[’Fisika’, ’kimia’, 1997, 2000]

Setelah menghapus nilai pada indeks 2:

[’Fisika’, ’kimia’, 2000]

Catatan: hapus () metode dibahas di bagian selanjutnya.

Operasi Daftar Dasar

Daftar merespons operator + dan * seperti string; Mereka berarti penggabungan dan pengulangan di sini juga, kecuali hasilnya adalah daftar baru, bukan string.

Sebenarnya, daftar merespons semua operasi urutan umum yang kami gunakan pada senar di bab sebelumnya.

| Python Expression | Results | Description |
|---|---|---|
| len([1, 2, 3]) | 3 | Length |
| [1, 2, 3] + [4, 5, 6] | [1, 2, 3, 4, 5, 6] | Concatenation |
| ['Hi!'] * 4 | ['Hi!', 'Hi!', 'Hi!', 'Hi!'] | Repetition |
| 3 in [1, 2, 3] | True | Membership |
| for x in [1, 2, 3]: print x, | 1 2 3 | Iteration |

Indexing, Slicing, dan Matrixes

Karena daftar adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk daftar seperti yang mereka lakukan untuk string.

Dengan asumsi masukan berikut -

L = ['spam', 'Spam', 'SPAM!']

| Python Expression | Results | Description |
|---|---|---|
| L[2] | 'SPAM!' | Offsets start at zero |
| L[-2] | 'Spam' | Negative: count from the right |
| L[1:] | ['Spam', 'SPAM!'] | Slicing fetches sections |

**Built-in List Functions & Methods:**

Python includes the following list functions –

| SN | Function with Description |
|---|---|
| 1 | cmp(list1, list2) Compares elements of both lists. |
| 2 | len(list) Gives the total length of the list. |
| 3 | max(list) Returns item from the list with max value. |
| 4 | min(list) Returns item from the list with min value. |
| 5 | list(seq) Converts a tuple into list. |

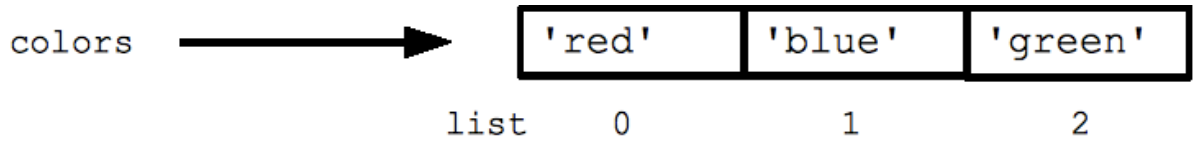
Python includes following list methods

| SN | Methods with Description |
|---|---|
| 1 | list.append(obj) Appends object obj to list |
| 2 | list.count(obj) Returns count of how many times obj occurs in list |
| 3 | list.extend(seq) Appends the contents of seq to list |
| 4 | list.index(obj) Returns the lowest index in list that obj appears |
| 5 | list.insert(index, obj) Inserts object obj into list at offset index |
| 6 | list.pop(obj=list[-1]) Removes and returns last object or obj from list |
| 7 | list.remove(obj) Removes object obj from list |
| 8 | list.reverse() Reverses objects of list in place |
| 9 | list.sort([func]) Sorts objects of list, use compare func if given |

Python memiliki tipe daftar built-in yang hebat dengan nama "daftar". Daftar literal ditulis dalam tanda kurung siku []. Daftar bekerja sama dengan senar - gunakan fungsi len () dan tanda kurung siku [] untuk mengakses data, dengan elemen pertama di indeks 0. (Lihat daftar dokumen python.org resmi).

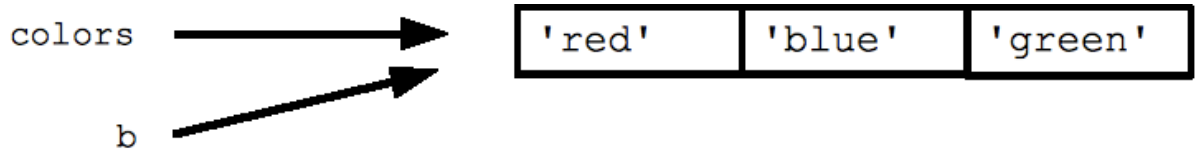
```
Warna = [’merah’, ’biru’, ’hijau’]

cetak warna [0] # # merah
cetak warna [2] # # hijau
Cetak len (warna) # # 3
```




Tugas dengan = daftar tidak membuat salinan. Sebagai gantinya, tugas membuat kedua variabel menunjuk ke satu daftar di memori.

```
B = warna # # Tidak menyalin daftar
```




"Daftar kosong" hanyalah sepasang kurung kosong []. ’+’ Bekerja untuk menambahkan dua daftar, jadi [1, 2] + [3, 4] menghasilkan [1, 2, 3, 4] (ini sama seperti + dengan string).

FOR dan IN

Python’s * untuk * dan * in * constructs sangat berguna, dan penggunaan pertama dari yang akan kita lihat adalah dengan daftar. * Untuk * membangun - untuk daftar var - adalah cara mudah untuk melihat setiap elemen dalam daftar (atau koleksi lainnya). Jangan menambah atau menghapus dari daftar selama iterasi.

```
kotak = [1, 4, 9, 16]
Jumlah = 0
Untuk num dalam kotak:
  Jumlah + = num
Jumlah cetak
```

Jika Anda tahu hal macam apa yang ada dalam daftar, gunakan nama variabel dalam lingkaran yang menangkap informasi seperti "num", atau "name", atau "url". Karena kode python tidak memiliki sintaks lain untuk mengingatkan Anda tentang tipe, nama variabel Anda adalah cara kunci bagi Anda untuk tetap mempertahankan apa yang sedang terjadi.

* Dalam * membangun sendiri adalah cara mudah untuk menguji apakah sebuah elemen muncul dalam daftar (atau koleksi lainnya) - nilai dalam koleksi - tes jika nilainya ada dalam koleksi, mengembalikan True / False.

```
Daftar = [’larry’, ’curly’, ’moe’]
jika ’keriting’ dalam daftar:
  Cetak ’yay’
```

The for / in constructs sangat umum digunakan pada kode Python dan bekerja pada tipe data selain list, jadi sebaiknya hafalkan sintaksnya. Anda mungkin memiliki kebiasaan dari bahasa lain di mana Anda memulai pengulangan manual melalui koleksi, dengan Python yang seharusnya Anda gunakan untuk / in.

Anda juga dapat menggunakannya untuk / dalam mengerjakan sebuah string. String bertindak seperti daftar karakternya, jadi untuk ch di s: print ch mencetak semua karakter dalam sebuah string.
Jarak

Fungsi range (n) menghasilkan angka 0, 1, ... n-1, dan range (a, b) mengembalikan a, a + 1, ... b-1 - sampai tapi tidak termasuk angka terakhir . Kombinasi fungsi for-loop dan range () memungkinkan Anda membuat numerik tradisional untuk loop:

```
# # print the numbers from 0 through 99  for i in range(100):    print i
```

Ada varian xrange () yang menghindari biaya membangun keseluruhan daftar untuk kasus sensitif kinerja (dalam Python 3000, range () akan memiliki perilaku kinerja yang baik dan Anda dapat melupakan xrange ()).

Sementara Loop

Python juga memiliki standar while-loop, dan * break * dan * continue * statements bekerja seperti di C ++ dan Java, mengubah jalannya loop terdalam. Di atas untuk / dalam loop memecahkan kasus umum iterasi pada setiap elemen dalam daftar, namun loop sementara memberi Anda kontrol penuh atas angka indeks. Berikut adalah loop sementara yang mengakses setiap elemen ke-3 dalam daftar:

```
 # # Mengakses setiap elemen ke-3 dalam daftar
I = 0
sementara i <len (a):
  cetak sebuah [i]
  i = i + 3
```

Daftar metode

Berikut adalah beberapa metode daftar umum lainnya.

List.append (elem) - menambahkan satu elemen ke akhir daftar. Kesalahan umum: tidak mengembalikan daftar baru, cukup modifikasi yang asli.

List.insert (indeks, elem) - memasukkan elemen pada indeks yang diberikan, menggeser elemen ke kanan.

List.extend (list2) menambahkan elemen dalam list2 ke akhir daftar. Menggunakan + atau + = pada daftar sama dengan menggunakan extend ().

List.index (elem) - mencari elemen yang diberikan dari awal daftar dan mengembalikan indeksnya. Melempar ValueError jika elemen tidak muncul (gunakan "in" untuk memeriksa tanpa ValueError).

List.remove (elem) - mencari instance pertama dari elemen yang diberikan dan menghapusnya (melempar ValueError jika tidak ada)

List.sort () - menyusun daftar di tempat (tidak mengembalikannya). (Fungsi yang diurutkan () yang ditunjukkan di bawah ini lebih diutamakan.)

List.reverse () - membalik daftar di tempat (tidak mengembalikannya)

List.pop (index) - menghapus dan mengembalikan elemen pada indeks yang diberikan. Mengembalikan elemen paling kanan jika indeks dihilangkan (kira-kira kebalikan dari append ()).

Perhatikan bahwa ini adalah * metode * pada daftar objek, sedangkan len () adalah fungsi yang mengambil daftar (atau string atau apapun) sebagai argumen.

```
Daftar = ['larry', 'curly', 'moe']
 List.append ('shemp') # # append elem di akhir
 List.insert (0, 'xxx') # # masukkan elem pada indeks 0
 list.extend (['yyy', 'zzz']) # # tambahkan daftar elems at end
 daftar cetak # # ['xxx', 'larry', 'curly', 'moe', 'shemp', 'yyy', 'zzz']
 Print list.index ('keriting') # # 2

 List.remove ('curly') # # cari dan hapus elemen itu
 List.pop (1) # # menghapus dan mengembalikan 'larry'
 daftar cetak # # ['xxx', 'moe', 'shemp', 'yyy', 'zzz']
```

Kesalahan umum: perhatikan bahwa metode di atas tidak * mengembalikan * daftar yang dimodifikasi, mereka hanya memodifikasi daftar aslinya.

```
 Daftar = [1, 2, 3]
 Print list.append (4) # # TIDAK, tidak bekerja, append () return Tidak ada
  # # Pola yang benar:
 List.append (4)
 Daftar cetak # # [1, 2, 3, 4]
```

st Build Up

Salah satu pola yang umum adalah dengan memulai daftar daftar kosong [], lalu gunakan append () atau extend () untuk menambahkan elemen ke dalamnya:

```
 List = [] # # Mulai sebagai daftar kosong
 List.append ('a') # # Gunakan append () untuk menambahkan elemen
 List.append ('b')
```

Daftar irisan

Slice bekerja pada daftar seperti halnya senar, dan juga dapat digunakan untuk mengubah sub-bagian daftar.

```
 Daftar = ['a', 'b', 'c', 'd']
 Daftar cetak [1: -1] # # ['b', 'c']
 Daftar [0: 2] = 'z' # # ganti ['a', 'b'] dengan ['z']
 Daftar cetak # # ['z', 'c', 'd']
```

Tipe data daftar memiliki beberapa metode lagi. Berikut adalah semua metode daftar objek:

List.append (x)

Tambahkan item ke bagian akhir daftar. Setara dengan [len (a):] = [x].

list.extend (iterable)

Perluas daftar dengan menambahkan semua item dari iterable. Setara dengan [len (a):] = iterable.

list.insert (i, x)

Masukkan item pada posisi tertentu. Argumen pertama adalah indeks dari elemen yang sebelum dimasukkan, jadi a.insert (0, x) memasukkan di bagian depan daftar, dan a.insert (len (a), x) setara dengan a.append ( x).

List.remove (x)

Hapus item pertama dari daftar yang nilainya x. Ini adalah kesalahan jika tidak ada item seperti itu.

List.pop ([i])

Hapus item pada posisi yang diberikan dalam daftar, dan kembalikan. Jika tidak ada indeks yang ditentukan, a.pop () menghapus dan mengembalikan item terakhir dalam daftar. (Tanda kurung siku di sekitar i pada tanda tangan metode menunjukkan bahwa parameternya adalah opsional, bukankah Anda harus mengetikkan tanda kurung siku pada posisi itu. Anda akan sering melihat notasi ini di Referensi Perpustakaan Python.)

List.clear ()

Hapus semua item dari daftar. Setara dengan del a [:].

List.index (x [, start [, end]])

Kembalikan indeks berbasis nol dalam daftar item pertama yang nilainya x. Meningkatkan ValueError jika tidak ada item seperti itu.

Argumen dan argumen opsional dimulai dengan interpretasi seperti notasi irisan dan digunakan untuk membatasi pencarian ke urutan berikutnya dari daftar. Indeks yang dikembalikan dihitung relatif terhadap awal urutan penuh daripada argumen awal.

List.count (x)

Kembalikan berapa kali x muncul dalam daftar.

List.sort (key = None, reverse = False)

Urutkan item daftar di tempat (argumen dapat digunakan untuk kustomisasi sortir, lihat diurutkan () untuk penjelasan mereka).

List.reverse ()

Membalikkan unsur daftar di tempat.

List.copy ()

Kembalikan salinan daftar yang dangkal. Setara dengan [:].

Contoh yang menggunakan sebagian besar metode daftar:

```
»> fruits = ’orange’, ’apple’, ’pear’, ’banana’, ’kiwi’, ’apple’, ’banana’]
»> fruits.count(’apple’)
2
»> fruits.count(’tangerine’)
0
»> fruits.index(’banana’)
3
»> fruits.index(’banana’, 4)  # Find next banana starting a position 4
6
»> fruits.reverse()
»> fruits
[’banana’, ’apple’, ’kiwi’, ’banana’, ’pear’, ’apple’, ’orange’]
»> fruits.append(’grape’)
»> fruits
[’banana’, ’apple’, ’kiwi’, ’banana’, ’pear’, ’apple’, ’orange’, ’grape’]
»> fruits.sort()
»> fruits
[’apple’, ’apple’, ’banana’, ’banana’, ’grape’, ’kiwi’, ’orange’, ’pear’]
»> fruits.pop()
’pear’
```

mungkin telah memperhatikan bahwa metode seperti insert, remove atau sortir yang hanya memodifikasi daftar tidak memiliki nilai pengembalian tercetak - mereka mengembalikan default None. [1] Ini adalah prinsip desain untuk semua struktur data yang bisa berubah dengan Python.

```
»> stack = [3, 4, 5]
»> stack.append (6)
»> stack.append (7)
»> susun
[3, 4, 5, 6, 7]
»> stack.pop ()
7
»> susun
[3, 4, 5, 6]
»> stack.pop ()
6
»> stack.pop ()
5
»> susun
[3, 4]
```

### 5.1.2. Menggunakan Daftar sebagai Antrian

Hal ini juga memungkinkan untuk menggunakan daftar sebagai antrian, di mana elemen pertama yang ditambahkan adalah elemen pertama yang diambil ("first-in, first-out"); Namun, daftar tidak efisien untuk tujuan ini. Sementara menambahkan dan muncul dari akhir daftar dengan cepat, melakukan sisipan atau muncul dari awal daftar lambat (karena semua elemen lainnya harus digeser oleh satu).

Untuk menerapkan antrean, gunakan collections.deque yang dirancang agar cepat ditambahkan dan muncul dari kedua ujungnya. Sebagai contoh:
»>

```
»> dari koleksi import deque
»> antrian = deque (["Eric", "John", "Michael"])
»> queue.append ("Terry") # Terry tiba
»> queue.append ("Graham") # Graham tiba
»> queue.popleft () # Yang pertama tiba sekarang pergi
'Eric'
»> queue.popleft () # Yang kedua tiba sekarang pergi
'John'
»> antrian # Sisa antrian sesuai urutan kedatangan
deque (['Michael', 'Terry', 'Graham'])
```