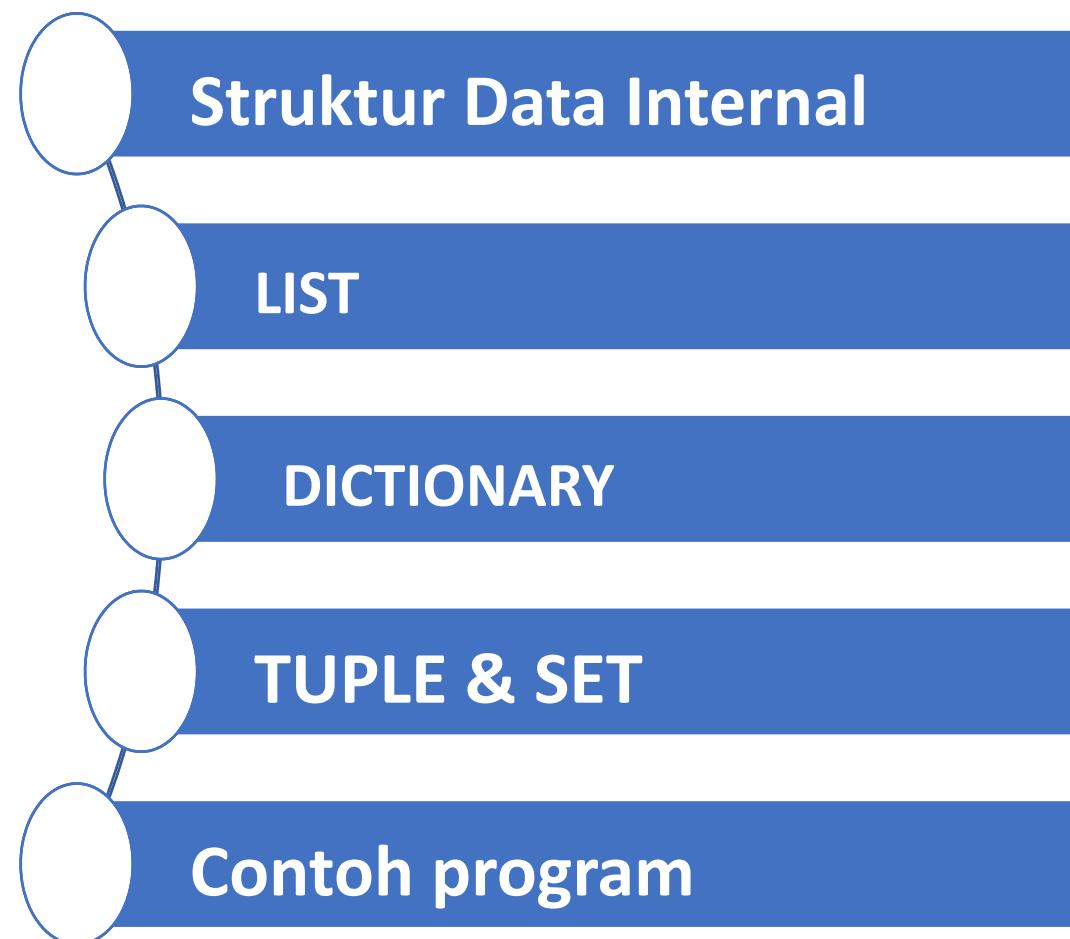


STRUKTUR DATA (PYTHON)

“Struktur Data Internal Python”

[@SUARGA] | [Pertemuan 05]

OutLine



Struktur Internal Data Python

- Python memiliki empat macam struktur data yang disajikan dalam format objek, yaitu: **List**, **Tuple**, **Dictionary** dan **Set**
- List, Tuple dan Dictionary mirip dengan larik
- List dapat dibuat langsung dengan menggunakan tanda kurung kotak []
- Tuple dapat dibuat langsung dengan menggunakan tanda kurung bulat ()
- Dictionary dapat dibuat langsung dengan menggunakan tanda kurung keriting { }

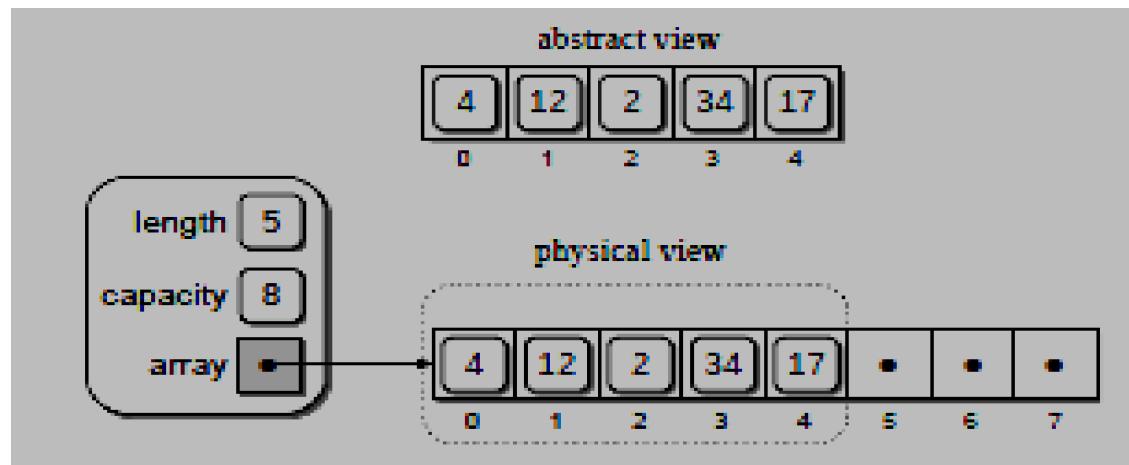
List

- Bahasa Python memiliki satu struktur data yang diberi nama “list” (sepintas telah dibicarakan pada modul-2), yang pada hakekatnya sama dengan struktur larik, namun secara internal struktur list merupakan “dynamic array”, larik yang size-nya bisa berkembang sesuai kebutuhan. Sebagai contoh pemakaian instruksi:
- `a = [0, 0, 0, 0, 0]`
- menciptakan sebuah list bernama **a** dengan lima elemen semuanya bernilai nol.

- Elemen-elemen pada python list bisa diakses dengan suatu indeks, indeks dimulai dari 0 (nol), sehingga kelima elemen list a diatas masing-masing adalah: $a[0]$, $a[1]$, ..., $a[4]$.
- python juga membolehkan indeks negatif (tidak diperkenankan oleh bahasa program lain seperti C dan Java), namun indeks -1 harus diberikan kepada elemen yang paling kanan, dengan demikian list a diatas bisa diakses melalui indeks: $a[-5]$, $a[-4]$, ..., $a[-1]$.

Abstraksi Memory LIST

- definisi struktur list seperti:
 $\text{pyList} = [4, 12, 2, 34, 17]$
- pada hakikatnya memiliki abstraksi memory sebagai berikut.



operator dan list

Operator + menyambung dua list:

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
```

Operator * akan mengulang isi list

```
>>> [0] * 4
[0, 0, 0, 0]
>>> [1, 2, 3] * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

- Operator slice (yaitu a:b) bekerja pada list

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
```

```
>>> t[1:3] #index 1 hingga sebelum index 3  
['b', 'c']
```

```
>>> t[:4] #index 0 hingga sebelum index 4  
['a', 'b', 'c', 'd']
```

```
>>> t[2:] #index 2 hingga akhir  
['c', 'd', 'e', 'f']
```

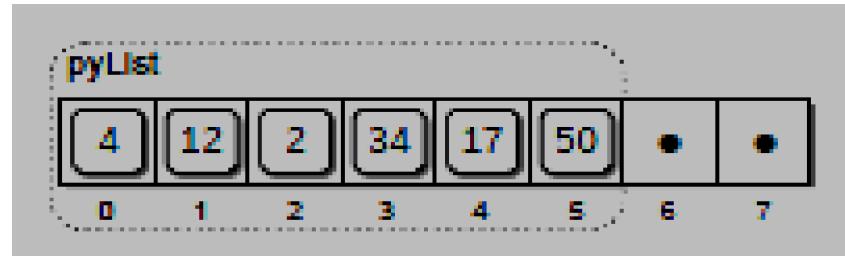
```
>>>
```

list ADT

- **append(x)**: menambahkan x diakhir list
- **extend(L)**: melakukan ekstensi list dengan menggandeng isi list L
- **insert(i,x)** : menyisipkan x dalam list pada index i
- **remove(x)** : menghapus elemen pertama dari list bernilai x, error bila x tidak ada
- **pop([i])** : menghapus elemen list pada posisi i
- **index(x)** : mencari index elemen bernilai x
- **count(x)** : mencacah elemen bernilai x
- **sort()** : men-sort elemen dalam list, ascending
- **reverse()** : mengurutkan terbalik elemen dalam list

- Contoh pemakain ADT List:

```
>>> pyList.append(50)
```



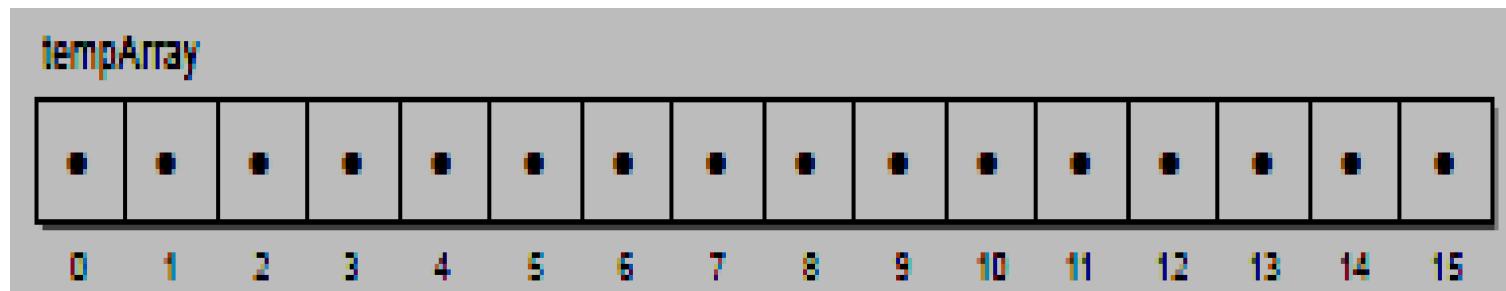
```
>>> pyList.append(18)
```

```
>>> pyList.append(64)
```

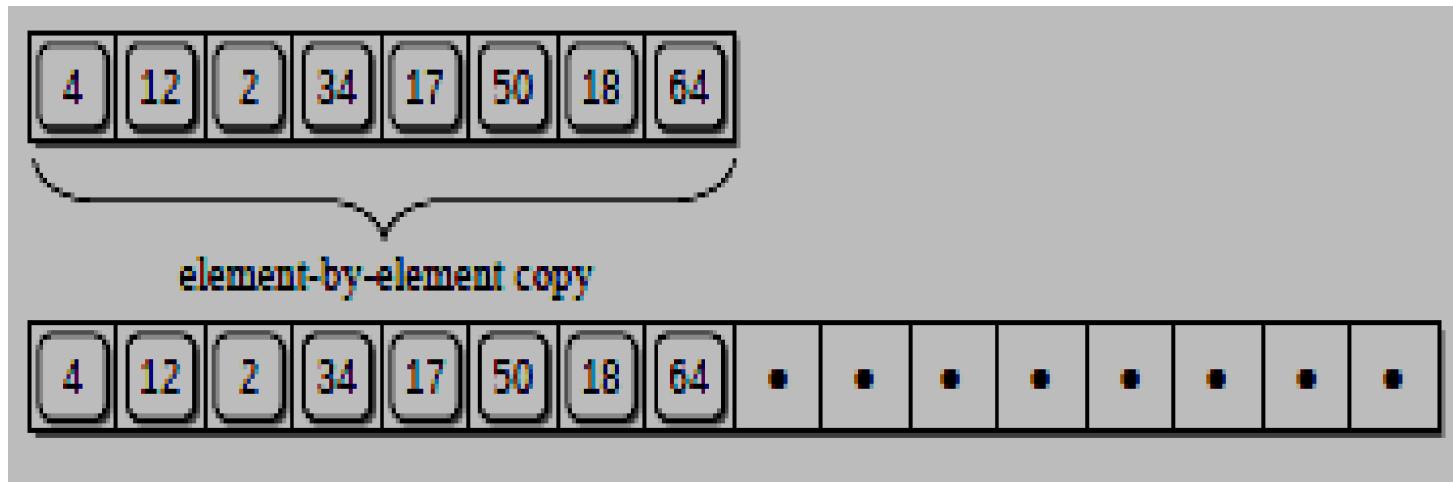
```
>>> pyList.append(6)
```



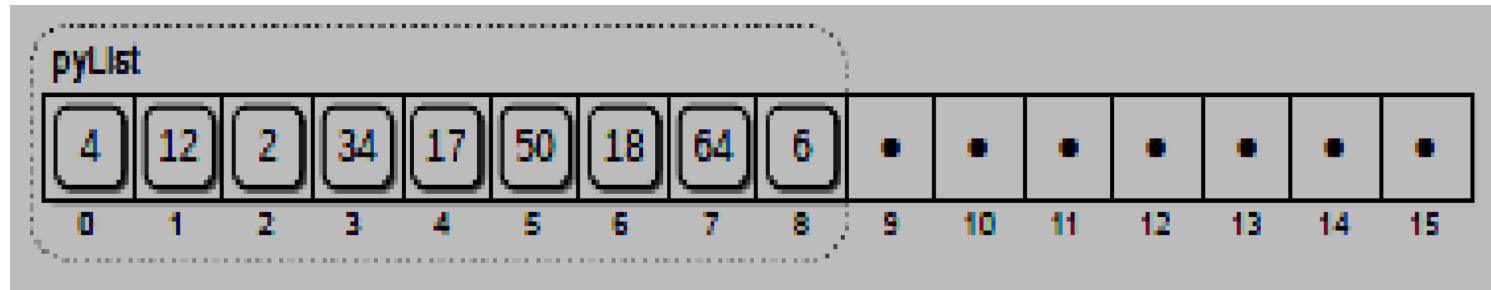
- Sebenarnya array pyList hanya tersisa dua lokasi, ketika 18 dan 64 dimasukkan maka array telah penuh.
 - Namun karena Python list adalah *array dinamis* maka secara otomatis ruang untuk data pada pyList akan diperbesar dua kali lipat oleh sistem python, ketika append(6) berlangsung, proses-nya berjalan sebagai berikut.
- (a) Sebuah array baru dengan kapasitas 2 kali dibentuk.



(b) Isi array lama di-salin ke array baru.



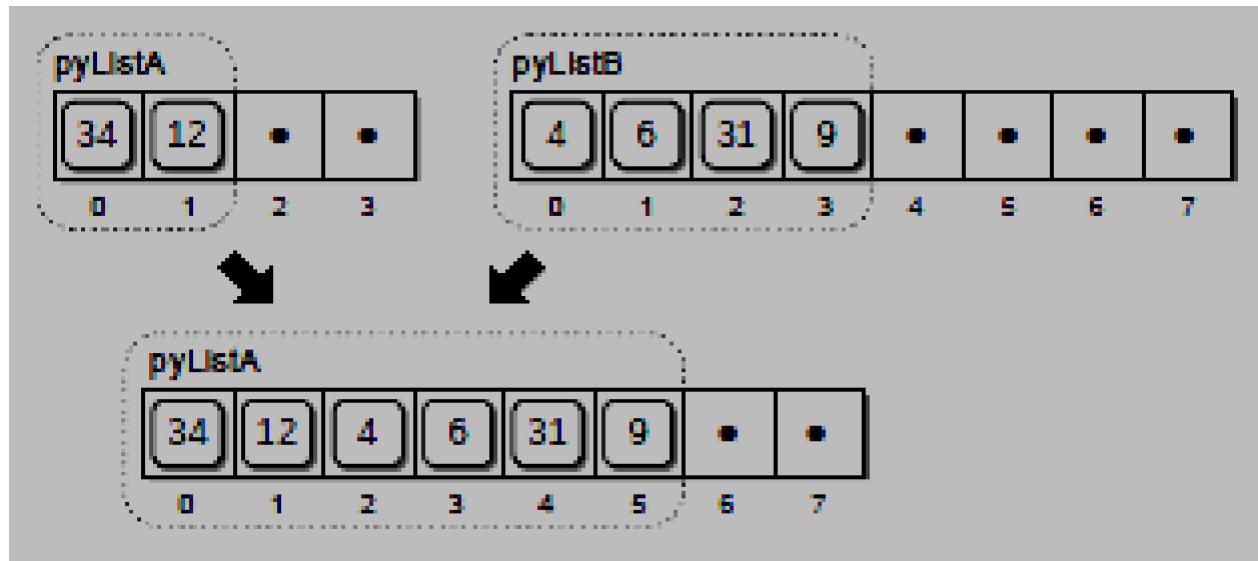
(c) nilai 6 dimasukkan ke posisi 8



```
>>> pyListA = [34, 12]  
>>> pyListB = [4, 6, 31, 9]  
>>> pyListA.extend(pyListB)  
>>> pyListA
```

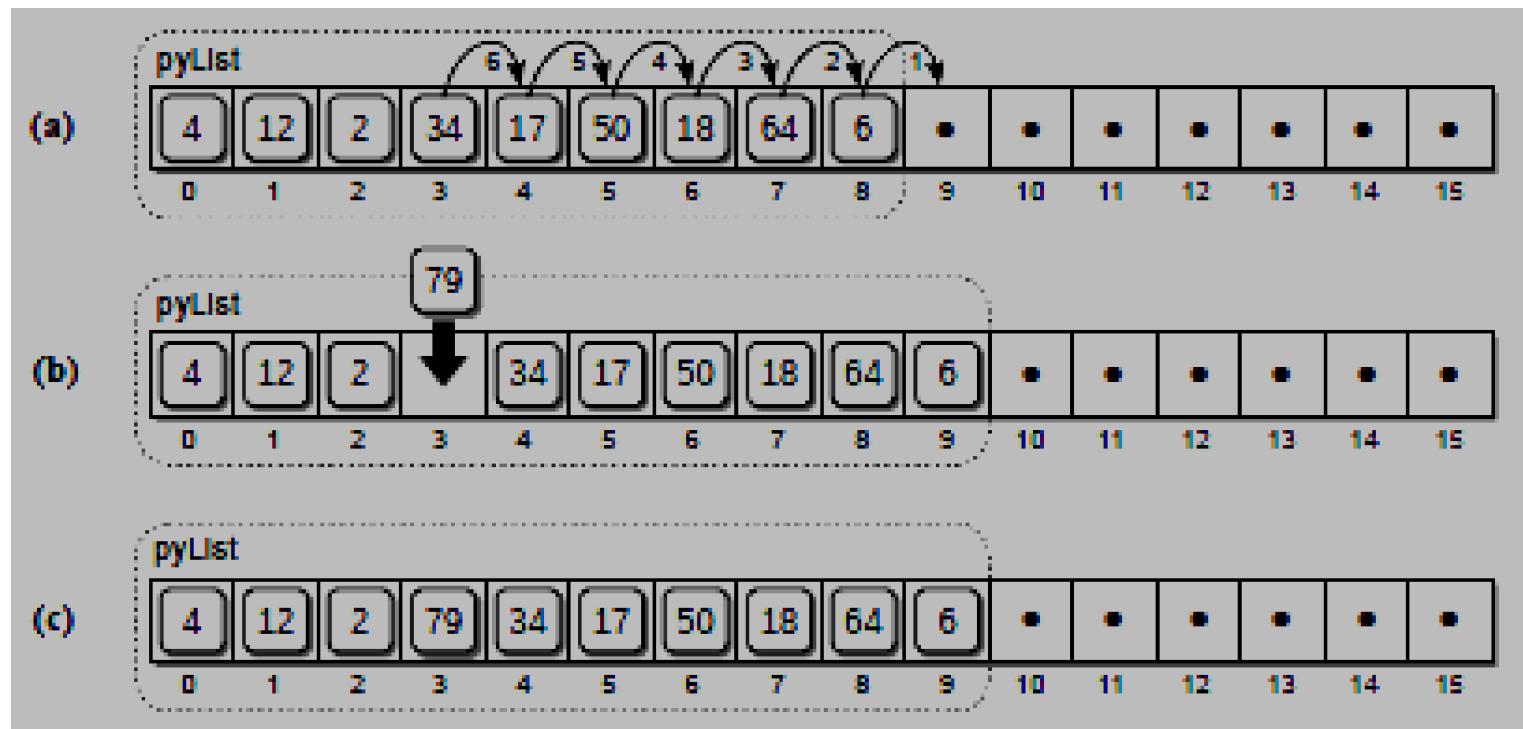
```
[34, 12, 4, 6, 31, 9]
```

```
>>>
```



```
>>> pyList.insert(3, 79) # menyisipkan 79 pada index 3
```

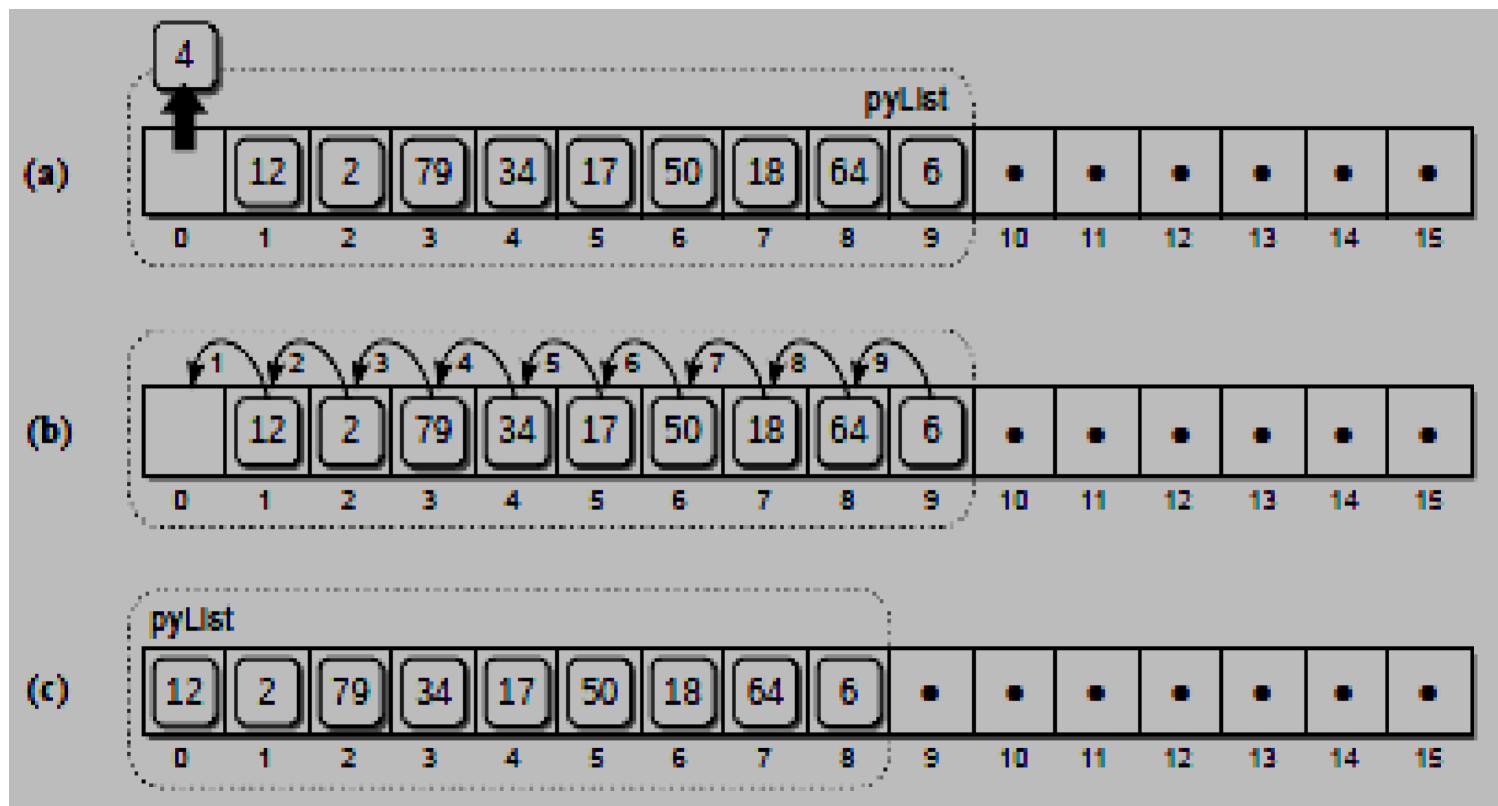
Penyisipan akan mengakibatkan penggeseran data kekanan agar data baru bisa disisipkan.



```
>>> pyList.pop(0) # ambil elemen pertama
```

4

Setelah elemen pertama di keluarkan maka elemen lain digeser ke kiri secara otomatis.



```
>>> pyList.pop()    # ambil elemen terakhir  
6  
>>> a=[66.6, 333, 1, 1234.5, 333, 66.6]  
>>> a.count(333), a.count(66.6), a.count('x')  
(2, 2, 0)  
>>> a.remove(333)  
>>> a  
[66.6, 1, 1234.5, 333, 66.6]  
>>> a.sort()  
>>> a  
[1, 66.6, 66.6, 333, 1234.5]  
>>> a.reverse()  
>>> a  
[1234.5, 333, 66.6, 66.6, 1]  
>>>
```

```
>>> t = ['a', 'b', 'c']
>>> del t[1]
>>> print t
['a', 'c']
```

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> del t[1:5]
>>> print t
['a', 'f']
```

```
>>> s = 'spam'
>>> t = list(s)
>>> print t
['s', 'p', 'a', 'm']
```

```
>>> s = 'pining for the fjords'  
>>> t = s.split()  
>>> print t  
['pining', 'for', 'the', 'fjords']
```

```
>>> t = ['pining', 'for', 'the', 'fjords']  
>>> delimiter = ' '  
>>> delimiter.join(t)  
'pining for the fjords'
```

```
>>> s = 'spam-spam-spam'  
>>> delimiter = '-'  
>>> s.split(delimiter)  
['spam', 'spam', 'spam']
```

Dictionary

- Struktur lain dalam Python yang mirip dengan list adalah “dictionary”. Pada struktur dictionary, ada dua jenis elemen yaitu: key dan value. Key merupakan index untuk membaca value, susunan elemennya sebagai berikut,
- `dict = {kunci : value, kunci : value, }`

misalnya:

- `eng2sp = {'one' : 'uno', 'two' : 'dos', 'three' : 'tres'}`

- Dictionary kosong dapat diciptakan dengan keyword dict().
- Perhatikan contoh prosedur/fungsi berikut, dimana akan dibentuk suatu dictionary d yang berisi frekuensi huruf dalam kalimat s,

```
def histogram(s):  
    d = dict()  
    for c in s:  
        if c not in d:  
            d[c]=1  
        else:  
            d[c] += 1  
    return d
```

```
h=histogram('dinosaurus')
print(h)
```

```
{'a': 1, 'd': 1, 'i': 1, 'o': 1, 'n': 1, 's': 2,
'r': 1, 'u': 2}
```

Fungsi histogram diatas membentuk suatu dictionary dimana key adalah huruf-huruf dari “dinosaurus” dan value-nya adalah frekuensi huruf-nya, misalnya

```
>>> print(h['a'])
```

```
1
```

```
>>> print(h['s'])
```

```
2
```

- Kapan struktur dictionary perlu digunakan?
 - ketika diperlukan assosiasi antara kunci (key) dan nilai (value)
 - ketika suatu nilai perlu diakses cepat berdasarkan kunci
 - ketika data sering berubah (mutable)

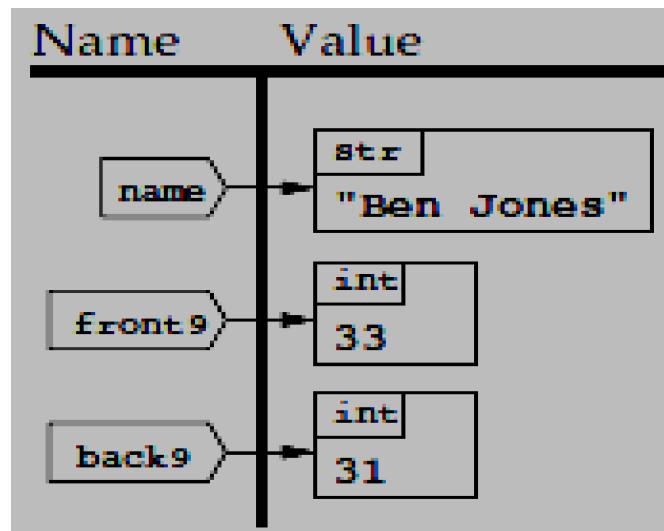
dictionary ADT

- **Beberapa method/fungsi untuk dictionary adalah:**
- **has_key(k)** : memeriksa apakah kunci k ada dalam dictionary
- **fromkeys(S,v)** : bentuk dict baru mulai dari kunci S bernilai v
- **clear()** : menghapus semua isi dictionary
- **keys()** : menampilkan semua kunci dalam dictionary
- **values()** : menampilkan semua nilai-nya
- **items()** : menampilkan semua pasangan key dan nilai
- **get(k)** : memperoleh nilai suatu kunci k, None bila tidak ada
- **pop(k)** : mengambil item dgn kunci k, dan memberikan nilainya
- **popitem()** : mengambil pasangan kunci dan nilai dari dict
- **update(d2)** : menyambung dictionary dengan dictionary d2

Abstraksi Memory dictionary

- Gambaran struktur data dari dictionary adalah sebagai berikut.

`d = { 'name' : 'Ben Jones', 'front9' : 33, 'back9' : 31 }`



Tuple

- Satu struktur dasar yang memiliki basis array yang lain adalah “tuple”, mirip list tetapi isi-nya tidak bisa di-ubah (immutable) sehingga cocok digunakan sebagai daftar key dalam dictionary.
- Membuat tuple menggunakan kurung biasa (), misalnya sebagai berikut:
- `t = ('a', 'b', 'c', 'd', 'e')`
- elemen-nya bisa diakses memakai indeks, misalnya:

```
print(t[0])
```

```
'a'
```

Karena bersifat immutable maka elemen-nya tidak bisa diubah, misalnya:

```
t[0] = 'A'
```

```
TypeError : object doesn't support item assignment
```

tuple ADT

- ADT dari tuple antara lain memiliki fungsi:
- **count()** : menghitung banyaknya data dalam tuple
- **index(v)** : mencari index dari data bernilai v, error bila v tidak ada
- **len()** : panjang dari tuple
- **iter(x)** : menciptakan objek yang bisa di-iterasi dari tuple
- **in()** : memeriksa apakah suatu nilai ada dalam tuple
- Selain itu objek tuple bisa dijumlahkan (+), dikalikan dengan skalar (*), dan membandingkan dua tuple dengan operasi logik ($>$, $<$, $==$, $>=$, $<=$, $!=$).

Struktur Set

- Set atau himpunan adalah wadah untuk koleksi data yang tidak boleh berulang, atau tidak boleh ada duplikasi nilai, semua elemen dalam set harus unik. Data dalam set berada diantara tanda kurung { .. }

Berikut ini adalah contoh menciptakan set:

```
>>> a=[1, 2, 3]      #buat sebuah list a
>>> s=set(a)        #kemudian dijadikan set s
>>> a
[1, 2, 3]
>>> s
{1, 2, 3}
```

ADT dari Set

Fungsi-fungsi atau operasi pada ADT set adalah:

- **union** : operasi union (gabung) dua himpunan (|)
- **intersection** : operasi intersection (nilai sama) pada dua himpunan (&)
- **difference** : mencari elemen beda dari dua set (-)
- **symmetric difference** : elemen berada di set 1 atau di set 2 tapi tidak kedua-dua-nya (^)

Contoh operasi Set

```
>>> set1={1,2,3}
>>> set2={3,4,5}
>>> a = set1 | set2    # union
>>> a
{1, 2, 3, 4, 5}
>>> b = set1 & set2    # intersection
>>> b
{3}
>>> c = set1 - set2    # difference
>>> c
{1, 2}
>>> d = set1 ^ set2    # symmetric difference
>>> d
{1, 2, 4, 5}
>>>
```

Beberapa Aplikasi

- **listArray.py** : membuat Array memakai List
- **MadLib.py** : aplikasi permainan kata memakai List
- **histoKata.py** : membuat histogram kata, memakai dictionary

listArray.py

```
#listArray.py => membuat array berbasis python List
#1.menciptakan array berisi 0
def ciptaArray(A,n):
    #n = jumlah elemen
    for i in range(n):
        A.append(0)

#2.mengisi array
def isiArray(A,n):
    for i in range(n):
        A[i] = int(input('isi: '))

#3.baca isi array
def bacaArray(A):
    r = input('0-Semua 1-satu data : ')
    r = int(r)
    if r < 1:
        print(A)
    else:
        x = int(input('posisi : '))
        print(A[x-1])
```

```
#4.cari data
def cariArray(x,A):
    n = len(A)
    ketemu = False
    for i in range(n):
        if (x == A[i]):
            print('Ketemu di posisi ',i+1,'\n')
            ketemu = True
    if (not ketemu):
        print(x,' tidak ada dalam array \n')

#5.hapus elemen
def hapus(x,A):
    n = len(A)
    ketemu = False
    for i in range(n):
        if (x == A[i]):
            print('Ketemu di posisi ',i+1,'\n')
            ketemu = True
            A[i] = ''
    if (not ketemu):
        print(x,' tidak ada dalam array \n')
```

list_Array_test.py

```
#list_Array_test.py
from listArray import *

def main():
    A = []
    ciptaArray(A,7)
    isiArray(A,7)
    print("Isi Larik : ")
    bacaArray(A)
    print("Mencari angka dalam larik")
    z = int(input("Data akan dicari: "))
    cariArray(z,A)
    z = int(input('Data yang akan dihapus: '))
    print("Menghapus elemen %d" % z)
    hapus(z,A)
    print("Larik setelah hapus data:")
    bacaArray(A)

main()
```

Hasil test:

```
*** Remote Interpreter Reinitialized ***
isi: 4
isi: 2
isi: 5
isi: 6
isi: 9
isi: 8
isi: 7
Isi Larik :
0-Semua 1-satu data : 0
[4, 2, 5, 6, 9, 8, 7]
Mencari angka dalam larik
Data akan dicari: 7
Ketemu di posisi 7

Data yang akan dihapus:5
Menghapus elemen 5
Ketemu di posisi 3

Larik setelah hapus data:
0-Semua 1-satu data : 0
[4, 2, '', 6, 9, 8, 7]
>>>
```

```
# MadLib.py
import random
namesList = [ 'Weird Al Yankovic', 'The Teenage Mutant Ninja Turtles', \
              'Supergirl', 'The Stay Puft Marshmallow Man', 'Shrek', \
              'Sherlock Holmes', 'The Beatles', 'Powerpuff Girl', \
              'The Pillsbury Doughboy']

while True:
    # Choose a random index into the namesList
    nameIndex = random.randrange(0, 9)
    name = namesList[nameIndex] # Use the index to choose a random name
    verb = input('Enter a verb: ')
    adjective = input('Enter an adjective: ')
    noun = input('Enter a noun: ')
    sentence = name + ' ' + verb + \
               ' through the forest, hoping to escape the ' + \
               adjective + ' ' + noun + '.'

    print()
    print(sentence)
    print()
    # See if the user wants to quit or continue
    answer = input('Type "q" to quit, or Enter to continue: ')
    if answer == 'q':
        break

print('Bye')
```

Hasil test

```
Enter a verb: running
```

```
Enter an adjective: slowly
```

```
Enter a noun: lion
```

```
The Beatles running through the forest, hoping to escape  
the slowly lion.
```

```
Type "q" to quit, or Enter to continue:
```

```
Enter a verb: jumping
```

```
Enter an adjective: faster
```

```
Enter a noun: monkey
```

```
Shrek jumping through the forest, hoping to escape the  
faster monkey.
```

```
Type "q" to quit, or Enter to continue: q
```

```
Bye
```

```
>>>
```

histoKata.py

```
#histoKata.py - histogram kata dari file teks
import string
def proses_file(filename):
    # hist adalah dictionary yang menampung kata
    hist = {}
    try:
        fp = open(filename)

        for line in fp:
            proses_line(line, hist)
    return hist
except IOError as ioerr:
    print('File tidak bisa dibuka ...' + str(ioerr))
return (None)
```

```
def proses_line(line, hist):
    # memisah setiap baris menjadi beberapa kata
    # memasukkan setiap kata ke dalam hist{}
    # Apabila ada hyphens maka ganti dengan spasi
    line = line.replace('-', ' ')
    for word in line.split():
        # mengabaikan semua tanda baca (punctuation)
        # dan mengubah ke huruf kecil (lowercase)
        word = word.strip(string.punctuation + \
                           string.whitespace)
        word = word.lower()
        # memasukkan kata ke histogram
        hist[word] = hist.get(word, 0) + 1

def daftar_kata(hist):
    # membuat daftar kata dan frekuensinya
    # ber-urut mulai dari yang tertinggi (descending)
    t = []
    for key, value in hist.items():
        t.append((value, key))

    t.sort()
    t.reverse()
    return t
```

```
def cetak_daftar_kata(hist, num=10):
    t = daftar_kata(hist)
    print ('Kata yang paling sering muncul adalah:')
    for freq, word in t[:num]:
        print (word, '\t\t', freq)

def total_kata(hist):
    # memberikan jumlah kata yang ada dalam file
    return sum(hist.values())

def different_words(hist):
    # memberikan banyaknya kata yang berbeda."""
    return len(hist)

# program utama
def main():
    path = "D:\\USER\\Python\\jaringan.txt"
    hist = proses_file(path)
    print ('Jumlah kata dalam file: : ', total_kata(hist))
    print ('Banyaknya kata berbeda: ', different_words(hist))

    t = daftar_kata(hist)
    print ('20 Kata dengan frekuensi tertinggi:')
    for freq, word in t[0:20]:
        print ('%12s %4d' % (word, freq))

main()
```

Hasil test

```
===== RESTART: D:\USER\Python\histoKata.py ===
Jumlah kata dalam file: : 555
Banyaknya kata berbeda: 255
20 Kata dengan frekuensi tertinggi:
jaringan      20
komputer      15
    dan        14
    yang        13
    data        13
    untuk       10
dengan         10
    dapat       10
    satu        9
    pada        9
    dari        9
    dalam       9
informasi      8
    atau        8
    lain        7
secara         6
    maka        6
    ada         6
    ú           5
suatu          5
```