```
            'Ubuntu',  'FreeBSD',  'NetBSD',  'OpenBSD',  'Backtrack',  'Fedora',  'Slackware']


# cara mendefinisikan tuple  sebuah_tuple = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

# cara mendefinisikan dictionary  sebuah_dictionary = {'nama':'Wiro Sableng',
                'prodi':'ilmu komputer',  'email':'wirosableng@localhost',
                'website':'http://www.sitampanggarang.com'
                }


print sebuah_list  print sebuah_tuple
print sebuah_dictionary
```

Dengan menggunakan perintah **print** maka Anda dapat mencetak isi **list**, **tuple**, atau **dictionary** yang hasil keluarannya berupa string.

<< gambar 6.1 hasil eksekusi strukdat_1.py >>

Cara Akses List, Tuple, dan Dictionary

Setelah Anda mengenal cara membuat ketiga struktur data tersebut, sekarang Anda coba cara mengakses elemen – elemen pada struktur data tersebut. Ada beberapa cara mengakses elemen pada struktur data tersebut yang biasa dilakukan misalnya mengakses salah satu elemen dengan

menggunakan indeks, *slicing* indeks, dan mengakses elemen lewat pengulangan.

Untuk akses elemen lewat indeks elemen Anda panggil nama list kemudian disusul indeks elemen yang Anda inginkan dengan diapit tanda baca "[" dan "]", misal ada sebuah list dengan nama **daftar_barang** kemudian ingin mengakses indeks ke – 10, maka pemanggilan indeks pada list tersebut adalah **daftar_barang[9].** Kenapa di pemanggilan indeks nya dari 9 ? karena indeks tersebut diawali dari 0 sehingga indeks yang diinginkan akan dikurangi 1. Begitupun dengan tuple cara akses salah satu elemenya sama dengan cara akses salah satu elemen di list. Pada dictionary, untuk mengakses salah satu elemennya Anda panggil salah satu key-nya untuk mendapatkan data yang ditunjuk key tersebut.

Slicing indeks merupakan cara untuk mengakses beberapa elemen pada list dan tuple. Cara ini tidak dapat dilakukan di dictionary. Slicing indeks dilakukan dengan memanggil list atau tuple kemudian tentukan indeks awal slicing dan batas akhirnya. Kemudian indeks tersebut dipisahkan dengan tanda ":" dan diapit oleh tanda "[" dan "]". Misal ada sebuah list **daftar_barang** kemudian ingin mengambil 10 datanya dari indeks ke – 2 maka pemanggilannya adalah **daftar_barang[1:11]**.

Berikutnya cara terakhir yang biasa dilakukan oleh untuk mengakses elemen secara keseleruhan adalah dengan melalui pengulangan **for**. Melalui cara tersebut, isi dari list, tuple, dan dictionary dapat diambil elemennya selama iterasi pada pengulangan **for**. Pada list dan tuple jika datanya diambil lewat pengulangan **for,** setiap elemen akan langsung diekstrak di setiap iterasi dan dapat digunakan untuk keperluan pemrosesan pada proses di setiap iterasi. Kalau pada dictionary di setiap iterasi pengulangan bukan elemen yang diektrak tapi key-nya. Jadi saat ingin mengambil datanya Anda harus memanggil elemen dictionary tersebut dengan key yang didapatkan disetiap iterasi. Dibawah ini adalah contoh mengakses elemen dari list, tuple dan dictionary dengan tiga cara yang biasa dipakai programmer python.

listing : strukdat_2.py

```
# cara mendefinisikan list  sebuah_list = ['Zorin OS',
            'Ubuntu',  'FreeBSD',  'NetBSD',  'OpenBSD',  'Backtrack',  'Fedora',  'Slackware']


# cara mendefinisikan tuple  sebuah_tuple = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

# cara mendefinisikan dictionary  sebuah_dictionary = {'nama':'Wiro Sableng',
                'prodi':'ilmu komputer',  'email':'wirosableng@localhost',
                'website':'http://www.sitampanggarang.com'
                }
```

```
# mengakses elemennya
print "mengakses salah satu elemen : "  print "------------------------------"
print sebuah_list[5]  print sebuah_tuple[8]
print sebuah_dictionary['website']

print "\n\n"

# mengakses beberapa elemen
print "mengakses beberapa elemen : "  print "------------------------------"
print sebuah_list[2:5]  print sebuah_tuple[3:6]


print "\n\n"


# mengakses elemennya dengan looping
print "mengakses semua elemen dengan looping for : "  print "------------------------------"

for sebuah in sebuah_list:  print sebuah,
print "\n"

for sebuah in sebuah_tuple:  print sebuah,
print "\n"

for sebuah in sebuah_dictionary:
    print sebuah, ':', sebuah_dictionary[sebuah],
```

Apabila Anda jalankan kode program diatas maka akan muncul output seperti pada gambar berikut ini :

<< gambar 6.2 hasil eksekusi strukdat_2.py >>

Mengubah Isi List, Tuple, dan Dictionary

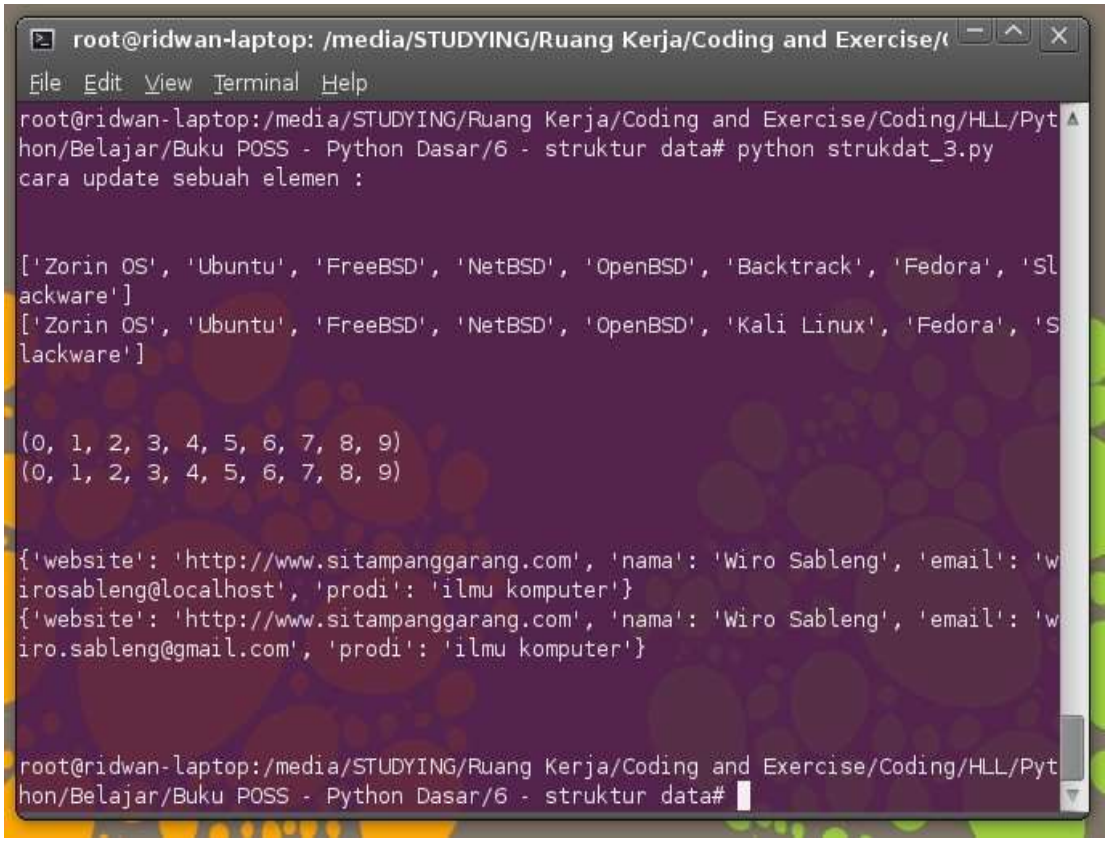
Ada waktunya Anda ingin mengubah salah satu elemen setelah mendefinisikan stuktur data. Misal ada sebuah list **daftar_barang** dan Anda ingin mengubah elemen ke-7 dengan data baru yang asalnya "kursi" menjadi "meja". Atau ada sebuah informasi dalam bentuk dictionary dengan key "nama" yang value asalnya "Son Go Ku" menjadi "Vash De Stampede". Cara mengubah data salah satu elemen di struktur data python mudah sekali. Tinggal tentukan indeks mana yang akan diubah, kemudian masukkan nilai baru kedalam indeks tersebut. Lebih jelasnya coba lihat contoh berikut :

listing : strukdat_3.py

```
# cara mendefinisikan list  sebuah_list = ['Zorin OS',
                'Ubuntu',  'FreeBSD',  'NetBSD',  'OpenBSD',  'Backtrack',  'Fedora',
```

```
            'Slackware']


# cara mendefinisikan tuple  sebuah_tuple = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

# cara mendefinisikan dictionary  sebuah_dictionary = {'nama':'Wiro Sableng',
                'prodi':'ilmu komputer',  'email':'wirosableng@localhost',
                'website':'http://www.sitampanggarang.com'
                }


# cara update sebuah elemen :
print "cara update sebuah elemen : "  print "\n"

print sebuah_list  sebuah_list[5] = 'Kali Linux'  print sebuah_list
print "\n"

print sebuah_tuple
# tuple tidak dapat melakukan operasi perubahan elemen :D  #sebuah_tuple[5] = 100
print sebuah_tuple  print "\n"

print sebuah_dictionary
sebuah_dictionary['email'] = 'wiro.sableng@gmail.com'  print sebuah_dictionary
print "\n\n"
```

Mudah sekali kan ? Dengan mengakses indeks tertentu pada list dan tuple serta mengakses keys tetentu pada dictionary, Anda dapat mengubah nilai pada indeks atau key tersebut dengan nilai yang baru.



<< gambar 6.3 hasil eksekusi strukdat_3.py >>

Menambahkan Data pada List, Tuple, dan Dictionary

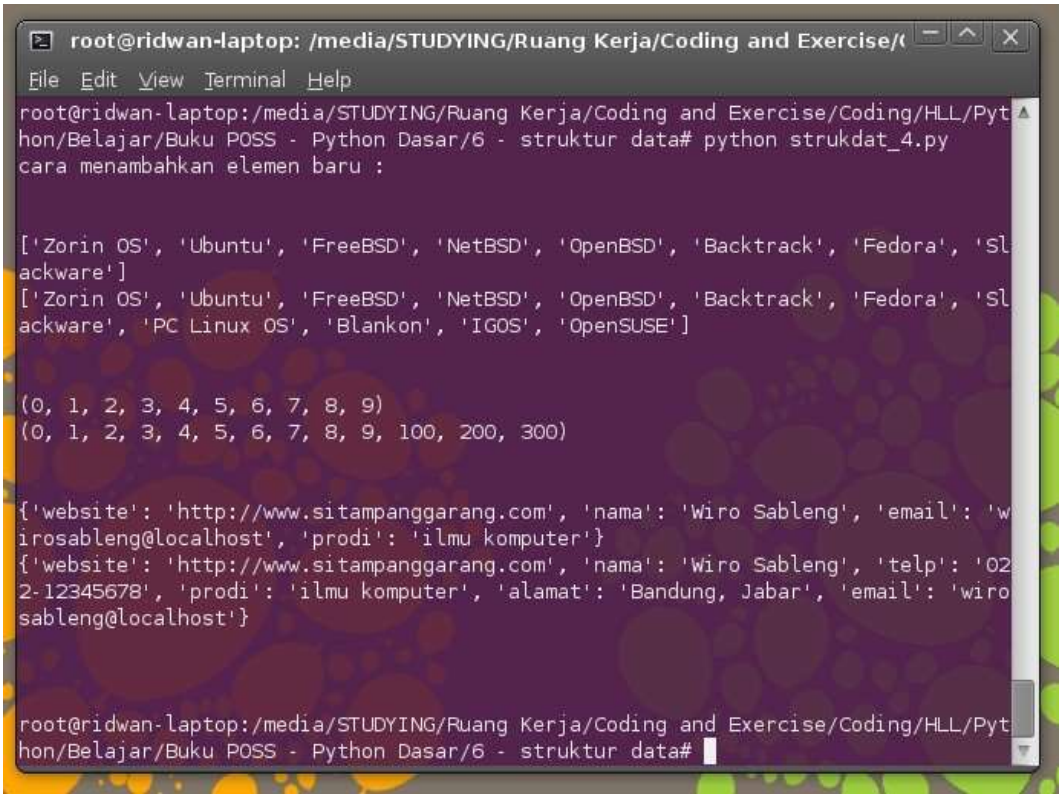
Ketiga struktur data inipun dapat ditambahkan data baru dari data semula. Pada list, digunakan tanda “+” untuk menambahkan data dari list baru ke list lama. Begitupun dengan tuple, tanda “+” digunakan untuk menambahkan data dari tuple baru ke tuple lama. Sedangkan pada dictionary digunakan method update dari dictionary yang ingin ditambahkan data baru. Kemudian dictionary semula akan memiliki data yang ditambahkan melalui method tersebut. Berikut adalah contoh menambahkan data baru pada ketiga struktur data tersebut.

listing : strukdat_4.py

```
# cara mendefinisikan list  sebuah_list = ['Zorin OS',
            'Ubuntu',  'FreeBSD',  'NetBSD',  'OpenBSD',  'Backtrack',  'Fedora',
```

```
            'Slackware']


# cara mendefinisikan tuple  sebuah_tuple = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

# cara mendefinisikan dictionary  sebuah_dictionary = {'nama':'Wiro Sableng',
                'prodi':'ilmu komputer',  'email':'wirosableng@localhost',
                'website':'http://www.sitampanggarang.com'
                }


# cara menambahkan data baru
print "cara menambahkan data baru : "  print "\n"

print sebuah_list
list_baru = sebuah_list + ['PC Linux OS', 'Blankon', 'IGOS', 'OpenSUSE']  print list_baru
print "\n"

print sebuah_tuple
tuple_baru = sebuah_tuple + (100, 200, 300)  print tuple_baru
print "\n"

print sebuah_dictionary
dictionary_baru = {'telp':'022-12345678', 'alamat':'Bandung, Jabar'}
sebuah_dictionary.update(dictionary_baru)
print sebuah_dictionary  print "\n\n"
```

Kode diatas jika dieksekusi akan muncul tampilan seperti berikut ini :



<< gambar 6.4 hasil eksekusi strukdat_4.py >>

Menghapus Isi List, Tuple, dan Dictionary

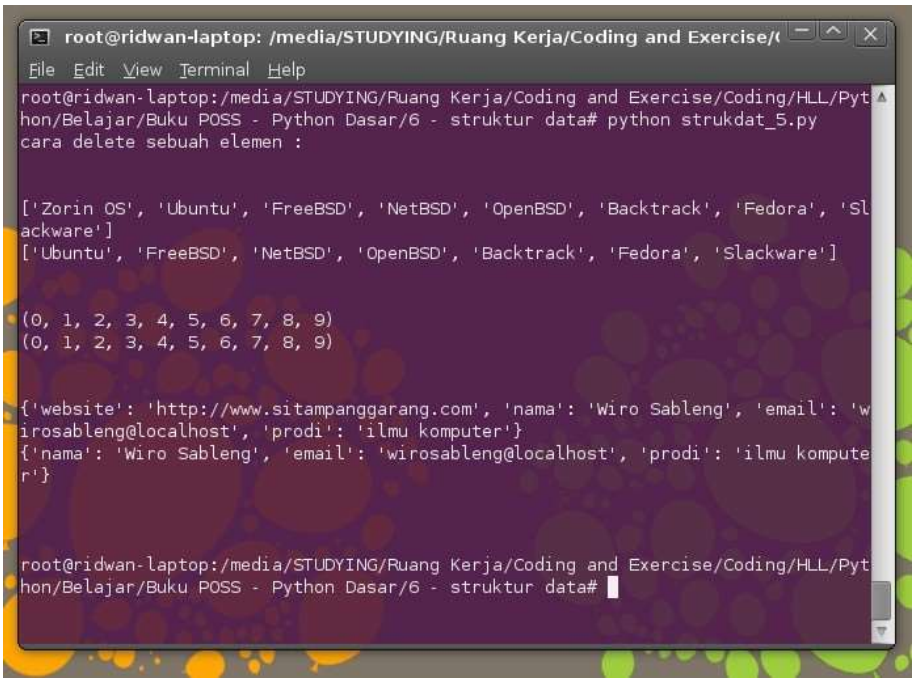
Tidak lengkap rasanya bila dalam sebuah informasi ada yang tidak dapat dihapus. Terkadang ada sebuah data yang tidak Anda butuhkan dan ingin Anda hilangkan dari kumpulan data yang dimiliki. Misal dalam sebuah list Anda ingin menghapus salah satu elemen. Atau di dictionary, Anda ingin menghilangkan salah satu key dari dictionary.tersebut. Di python sendiri penghapusan salah satu elemen dapat dilakukan di list dan dictionary. Sedangkan tuple tidak mendukung penghapusan elemen. Jika kita lakukan penghapusan pada salah satu elemen di tuple, maka akan muncul pesan error : “TypeError: 'tuple' object doesn't support item deletion”. Pada list Anda tinggal menunjuk salah satu elemennya dengan sebuah angka dari 0 sampai panjang list tersebut dikurangi satu dengan diapit tanda “[“ dan “]”. Sedangkan pada dictionary Anda tunjuk salah satu key yang akan dihapus dari dictionary. Berikut adalah contoh penghapusan salah satu elemen pada list dan dictionary.

listing : strukdat_5.py

```
# cara mendefinisikan list  sebuah_list = ['Zorin OS',
```

```
            'Ubuntu',  'FreeBSD',  'NetBSD',  'OpenBSD',  'Backtrack',  'Fedora',  'Slackware']


# cara mendefinisikan tuple  sebuah_tuple = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

# cara mendefinisikan dictionary  sebuah_dictionary = {'nama':'Wiro Sableng',
                'prodi':'ilmu komputer',  'email':'wirosableng@localhost',
                'website':'http://www.sitampanggarang.com'
                }


# cara delete sebuah elemen :
print "cara delete sebuah elemen : "  print "\n"

print sebuah_list  del sebuah_list[0]  print sebuah_list  print "\n"

print sebuah_tuple
# tuple tidak mendukung proses penghapusan elemen :D.(coba hilangkan tanda '#' disampingnya)
#del sebuah_tuple[8]
print sebuah_tuple  print "\n"

print sebuah_dictionary
del sebuah_dictionary['website']  print sebuah_dictionary
print "\n\n"
```

Kode diatas jika dieksekusi akan muncul tampilan seperti berikut :



<< gambar 6.5 hasil eksekusi strukdat_5.py >>

Menghapus List, Tuple, dan Dictionary

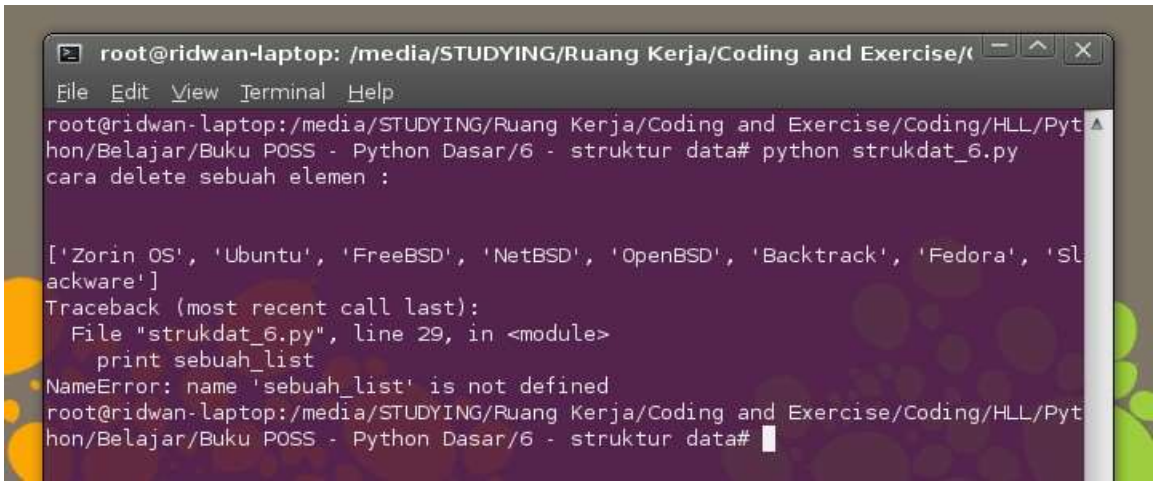
Pada contoh sebelumnya Anda hanya menghapus salah satu elemen. Lalu bagaimanakah jika ingin menghapus keseluruhan struktur data sehingga struktur data tersebut terhapus dari memory seluruhnya ?. Di Python dengan menggunakan perintah **del** pada sebuah struktur data maka struktur data tersebut akan dihapus sepenuhnya dari memory. Hal ini berlaku juga bagi variabel dan objek yang didefinisikan oleh programmer. Dengan hilangnya dari memory maka struktur data yang telah dihapus tidak dapat digunakan lagi oleh program yang Anda bangun.

list : strukdat_6.py

```
# cara mendefinisikan list  sebuah_list = ['Zorin OS',
            'Ubuntu',  'FreeBSD',  'NetBSD',  'OpenBSD',  'Backtrack',  'Fedora',  'Slackware']


# cara mendefinisikan tuple
```

```
sebuah_tuple = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

# cara mendefinisikan dictionary  sebuah_dictionary = {'nama':'Wiro Sableng',
                'prodi':'ilmu komputer',  'email':'wirosableng@localhost',
                'website':'http://www.sitampanggarang.com'
                }


# cara update sebuah elemen :
print "cara delete sebuah elemen : "  print "\n"

print sebuah_list  del sebuah_list  #print sebuah_list  print "\n"

print sebuah_tuple  del sebuah_tuple  #print sebuah_tuple  print "\n"

print sebuah_dictionary  del sebuah_dictionary  #print sebuah_dictionary  print "\n\n"
```

Cobalah hapus tanda “#” pada baris perintah print di bawah perintah del. Cobalah satu persatu dan amatilah apa yang terjadi. Jika kita coba pada print sebuah_list yang berada dibawah del sebuah_list. Maka akan muncul pesan error : “NameError : name 'sebuah_list' is not defined”



<< gambar 6.6 hasil eksekusi strukdat_6.py >>

Menggunakan Built-in Function pada List, Tuple, dan Dictionary

Apakah fitur – fitur manipulasi list, tuple, dan dictionary hanya terbatas pada hapus, tambah dan ubah ?. Python menyediakan beberapa fitur dasar lainnya yang dapat digunakan untuk proses mencari nilai maksimum dan minimum, menghitung panjang, membandingkan dua buah struktur data yang sejenis, bahkan mengubah struktur data dari list ke tuple atau sebaliknya.

Untuk mencari nilai maksimum pada list, tuple, atau dictionary digunakan function **max()**, sedangkan untuk mencari nilai minimum digunakan function **min()**. Untuk perbandingan dua buah struktur data sejenis, misal list dengan list, digunakanlah function **cmp()**. Function **cmp()** ini akan menghasilkan tiga nilai yaitu -1 jika list pertama kurang dari list kedua, 0 jika kedua list sama, dan 1 jika list pertama lebih besar dari list kedua. Kemudian untuk mencari jumlah elemen yang berada pada struktur data tersebut digunakan function **len()**. Dan terdapat juga untuk konversi tipe struktur data. Tapi fitur ini hanya dapat digunakan pada list dan tuple. Dictionary tidak mendukung proses konversi. Jadi hanya pengubahan dari list ke tuple dan sebaliknya. Untuk pengubahan dari list ke tuple digunakan function **tuple()** sedangkan untuk pengubahan dari tuple ke list digunakan function **list().**

Agar lebih paham cobalah sedikit source code tentang penggunaan built-in function yang digunakan untuk manipulasi list, tuple, dan dictionary.

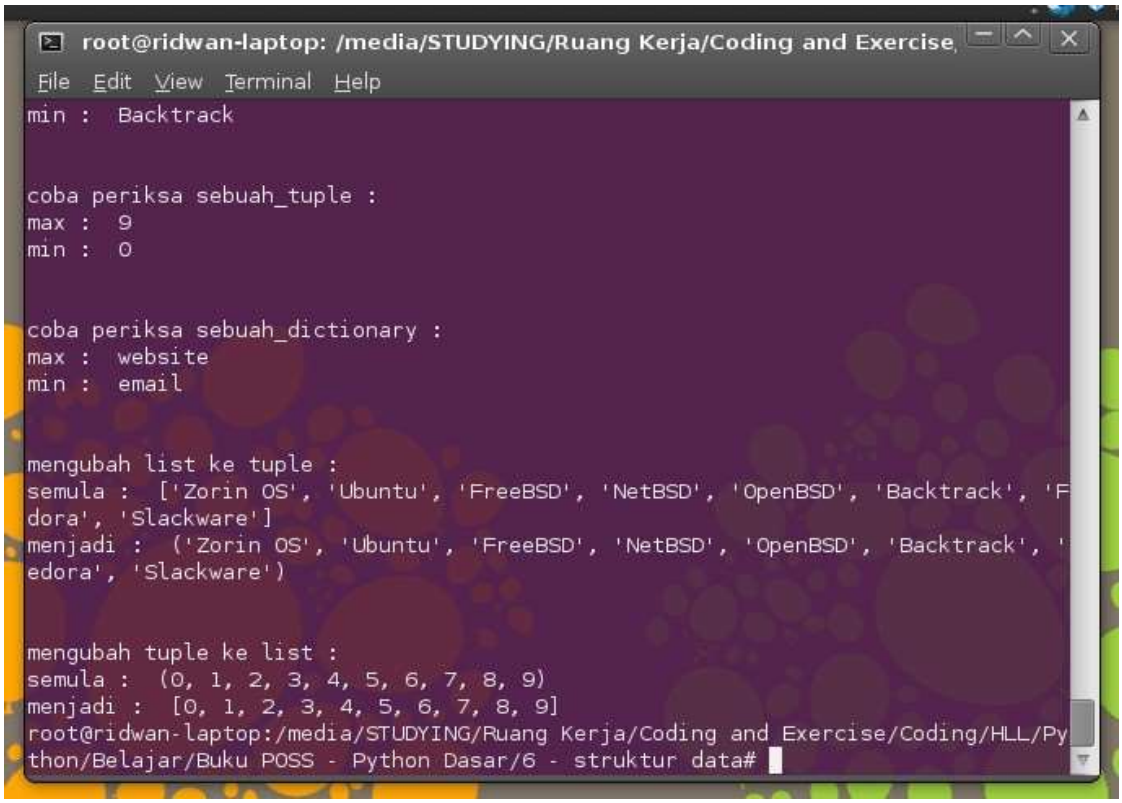
listing : strukdat_7.py

```
# cara mendefinisikan list  sebuah_list = ['Zorin OS',
            'Ubuntu',  'FreeBSD',  'NetBSD',  'OpenBSD',  'Backtrack',  'Fedora',  'Slackware']


# cara mendefinisikan tuple  sebuah_tuple = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

# cara mendefinisikan dictionary  sebuah_dictionary = {'nama':'Wiro Sableng',
                'prodi':'ilmu komputer',  'email':'wirosableng@localhost',
                'website':'http://www.sitampanggarang.com'
                }
```

```
# menambahkan elemen baru
print "menambahkan elemen baru : \n"  print sebuah_list
list_baru = sebuah_list + ['PC Linux OS', 'Blankon', 'IGOS', 'OpenSUSE']  print list_baru
print "\n"

print sebuah_tuple  tuple_baru = sebuah_tuple  print tuple_baru
print "\n"

print sebuah_dictionary
dictionary_baru = {'telp':'022-12345678', 'alamat':'Bandung, Jabar'}  print sebuah_dictionary
print "\n\n"

# membandingkan yang lama dengan yang baru
print "membandingkan yang lama dengan yang baru : \n"
print "sebuah_list banding list_baru : ", cmp(sebuah_list, list_baru)
print "sebuah_tuple banding tuple_baru : ", cmp(sebuah_tuple, tuple_baru)
print "sebuah_dictionary banding dictionary_baru : ", cmp(sebuah_dictionary, dictionary_baru)

print "\n\n"

# mengetahui panjang list, tuple, dan dictionary
print "mengetahui panjang list, tuple, dan dictionary : \n"  print "panjang sebuah_list : ",
len(sebuah_list)
print "panjang sebuah_tuple : ", len(sebuah_tuple)
print "panjang sebuah_dictionary : ", len(sebuah_dictionary)

print "\n\n"

# mengubah list, tuple, dictionary menjadi string
print "mengubah list, tuple, dictionary menjadi string : \n"
print str(sebuah_list), ' memiliki panjang karakter : ', len(str(sebuah_list))  print
str(sebuah_tuple), ' memiliki panjang karakter : ', len(str(sebuah_tuple))
print str(sebuah_dictionary), ' memiliki panjang karakter : ', len(str(sebuah_dictionary))

# mencari nilai max dan min
print "mencari nilai max dan min : \n"  print "coba periksa sebuah_list :"  print "max : ",
max(sebuah_list)
print "min : ", min(sebuah_list)  print "\n"
print "coba periksa sebuah_tuple :"  print "max : ", max(sebuah_tuple)  print "min : ",
min(sebuah_tuple)  print "\n"
print "coba periksa sebuah_dictionary :"
```

```
print "max : ", max(sebuah_dictionary)  print "min : ", min(sebuah_dictionary)  print "\n"

# mengubah list ke tuple dan sebaliknya  print "mengubah list ke tuple : "
print "semula : ", sebuah_list
print "menjadi : ", tuple(sebuah_list)  print "\n"
print "mengubah tuple ke list : "  print "semula : ", sebuah_tuple
print "menjadi : ", list(sebuah_tuple)
```

Dengan adanya *built-in function* tersebut pekerjaan Anda sudah dimudahkan oleh Python dalam memanipulasi struktur data yang telah Anda definisikan sebelumnya. Berikut adalah salah satu contoh hasil operasi dengan *built function* dari kode diatas.



<< gambar 6.7 hasil eksekusi strukdat_7.py >>