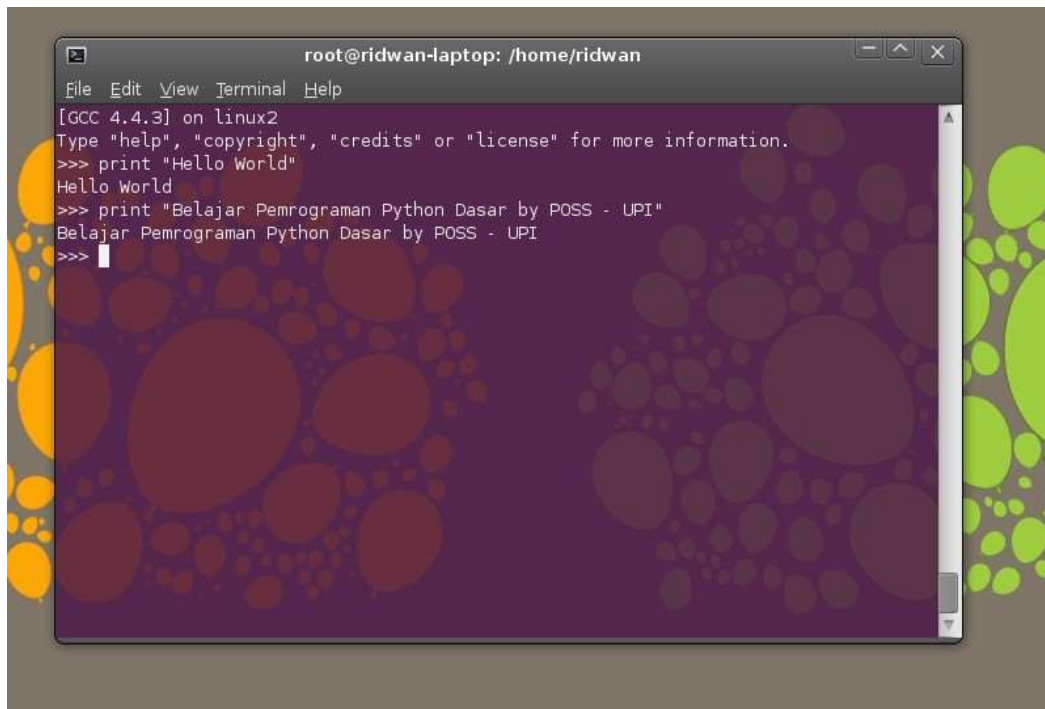


2. Cara Menggunakan Python

Menggunakan Python Interpreter Prompt dan Teks Editor

Untuk menggunakan Python, kita bisa memilih dua cara yang umum digunakan, yaitu lewat *Python Interpreter Prompt* dan mengeksekusi *file* Python dari *command line*. Anda bisa menggunakan *interpreter prompt* untuk sekedar menulis program kecil, atau menguji coba modul – modul yang dimiliki Python. Untuk menggunakannya jika sistem operasi Anda sudah di-*install* Python, coba panggil perintah **python** melalui *command line*. Dan jika pengaturan *path*-nya sudah sesuai, akan muncul gambar seperti berikut:

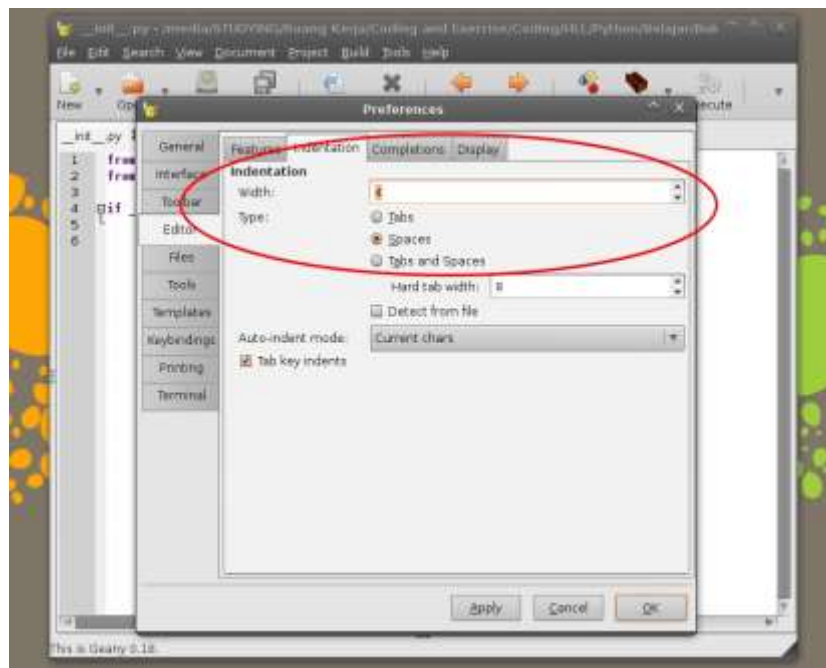


<< gambar 2.1 python interpreter prompt >>

Untuk keluar dari mode *interpreter prompt*, gunakan kombinasi `ctrl+d` atau `ctrl+z` kemudian tekan *enter*.

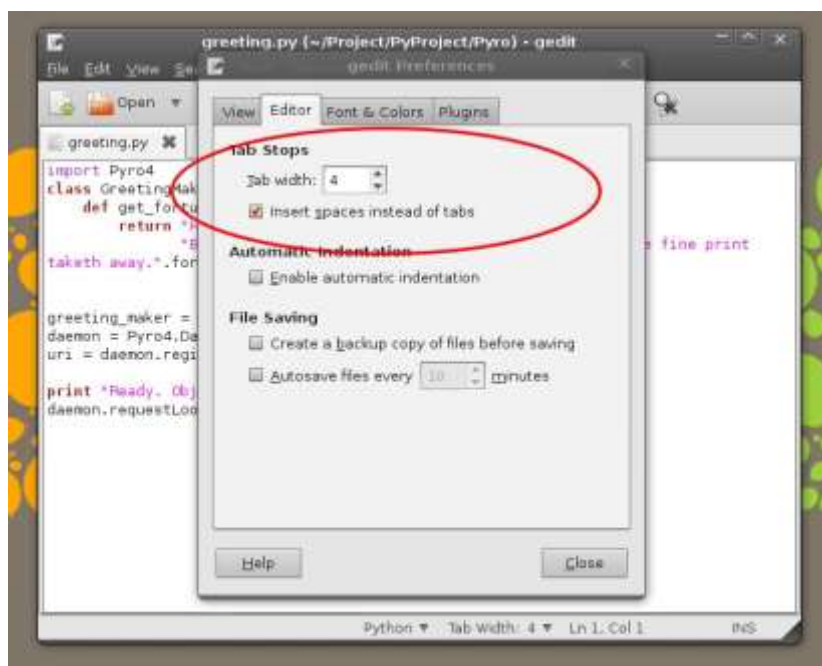
Lalu jika ingin menggunakan Python tanpa melalui *interpreter prompt*, Anda bisa menggunakan *text editor* yang Anda sering gunakan dan simpan *file* Python dengan ekstensi *file* *.py. Kemudian atur jarak tab pada *text editor* tersebut sebanyak empat satuan dan isi tab tersebut diganti dengan spasi atau memilih menu “*replace by space*”. Berikut adalah beberapa contoh *text editor* yang bisa digunakan untuk membuat program Python beserta menu untuk pengaturan tab.

Tampilan antarmuka Geany dan tempat pengaturan indentasinya :



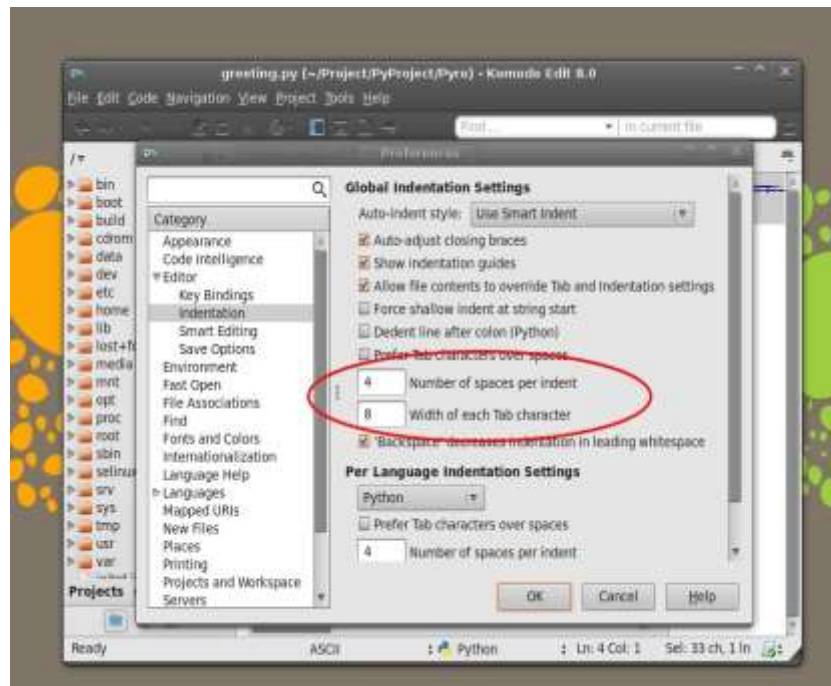
<< gambar 2.2 Geany dan pengaturan tab-nya>>

Tampilan antarmuka GEdit dan tempat pengaturan indentasinya :



<< gambar 2.3 GEdit dan pengaturan tab-nya>>

Tampilan antarmuka Komodo Edit dan tempat pengaturan indentasinya :



<< gambar 2.4 KomodoEdit dan pengaturan tab-nya >>

Jika Anda belum pernah belajar pemrograman Anda bisa gunakan GEdit atau Geany untuk membuat file Python. Jika sudah pernah belajar pemrograman, Anda bisa pilih *text editor* manapun sesuai selera.

Mencetak Informasi dengan Function “print”

Output digunakan pada program untuk memberikan *feedback* dan keadaan sebuah program, misal hasil perhitungan, pertanyaan, daftar pengguna, dan grafik.. Tapi dalam sebuah program konsol, *output* biasanya berupa teks yang dicetak dengan menggunakan *function* tertentu pada sebuah bahasa pemrograman. Di Python untuk mencetak teks ke layar digunakanlah *function* **print**. *Function* **print** ini akan mencetak *string* yang diinginkan. Ada banyak cara dalam menggunakan **print**, berikut adalah contoh penggunaan **print** dalam sebuah program konsol :

listing : pakai_python_1.py

```
# mencetak sebuah kalimat
```

```

print "Aku sedang belajar bahasa pemrograman python"

# mencetak angka print 6666

# mencetak variabel
sebuah_variabel = "Life is never float" print sebuah_variabel

# mencetak langsung sebuah operasi bilangan panjang = 10
lebar = 5
print (2 * panjang) + (2 * lebar)

# mencetak sebuah variabel dan string dengan menggunakan koma
nama = "Mario"

# dicetak diakhir
print "Nama saya adalah ", nama

# dicetak ditengah
print "Saya memainkan game Super ", nama, " bersama adik saya"

# dicetak diawal
print nama, " adalah karakter utama dalam permainan Super ", nama

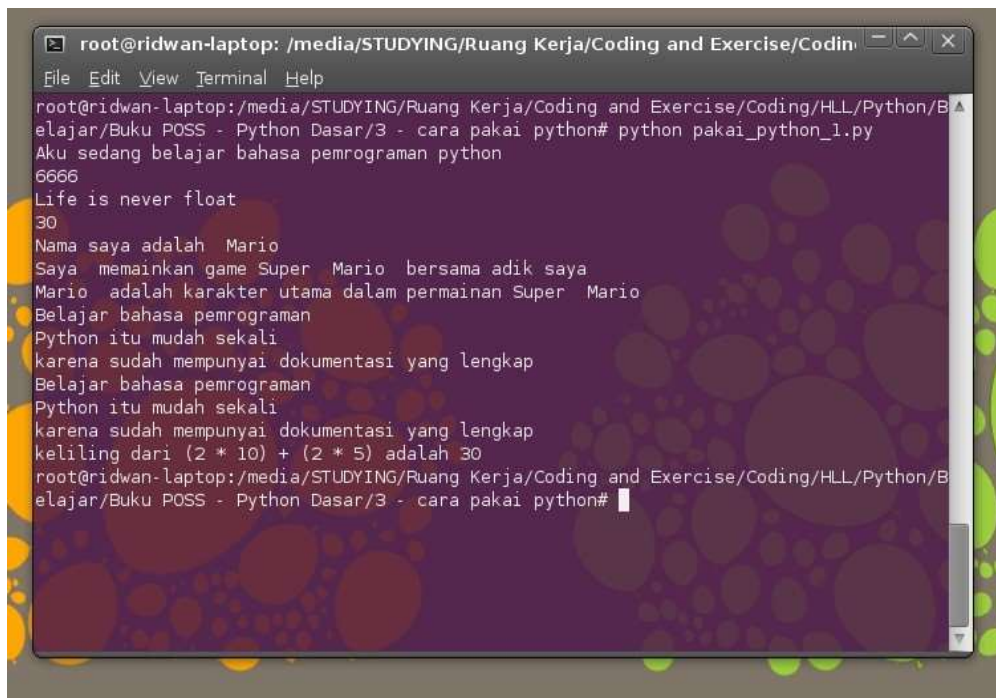
# mencetak banyak baris
print "Belajar bahasa pemrograman" print "Python itu mudah
sekali"
print "karena sudah mempunyai dokumentasi yang lengkap"

# mencetak banyak baris dalam satu kali print
print "Belajar bahasa pemrograman \nPython itu mudah sekali          \nkarena sudah mempunyai
dokumentasi yang lengkap"

# mencetak variabel pada string dengan format string panjang = 10
lebar = 5
keliling = (2 * panjang) + (2 * lebar)
print "keliling dari (2 * %d) + (2 * %d) adalah %d " % (panjang, lebar, keliling)

```

Jika kode diatas dieksekusi, akan tampil *output* seperti berikut :



```

root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Codin
File Edit View Terminal Help
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/3 - cara pakai python# python pakai_python_1.py
Aku sedang belajar bahasa pemrograman python
6666
Life is never float
30
Nama saya adalah Mario
Saya memainkan game Super Mario bersama adik saya
Mario adalah karakter utama dalam permainan Super Mario
Belajar bahasa pemrograman
Python itu mudah sekali
karena sudah mempunyai dokumentasi yang lengkap
Belajar bahasa pemrograman
Python itu mudah sekali
karena sudah mempunyai dokumentasi yang lengkap
keliling dari (2 * 10) + (2 * 5) adalah 30
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/3 - cara pakai python#

```

<< gambar 2.5 hasil eksekusi pakai_python_1.py >>

Menerima Masukan Data dengan Function “raw_input” dan “input”

Selain *output* tentu saja dalam membuat sebuah program Anda membutuhkan fitur untuk meminta *input* dari *user*. Fitur tersebut berguna untuk menciptakan interaksi antara *user* dan program yang Anda bangun. Di Python, untuk menerima *input* ada beberapa cara yang biasa digunakan :

raw_input, *function* ini berguna untuk menerima *input* dari *user* yang akan selalu dikonversi kedalam *string*. Misal Anda memberikan *input* berupa “Belajar Python”. Maka data tersebut akan ditampung sebagai *string* utuh. Kemudian pada **raw_input**, terdapat satu parameter yang akan dijadikan pertanyaan atau perintah tertulis saat meminta *input*. Jika Anda ingin memberikan *input* berupa angka, saat memasukkan angka tersebut tidak boleh lebih dari satu angka. Hal ini disebabkan karena ketika menggunakan **raw_input**, sekalipun yang diberikan adalah angka tetap akan dianggap *string*. Apabila Anda memberikan *input* satu angka kepada **raw_input**, Anda harus mengkonversinya dengan *function* **int**, **float**, **long**, atau beberapa *function* konversi ke angka lainnya sesuai dengan kebutuhan yang diinginkan. Coba perhatikan kode dibawah ini :

listing : pakai_python_2.py

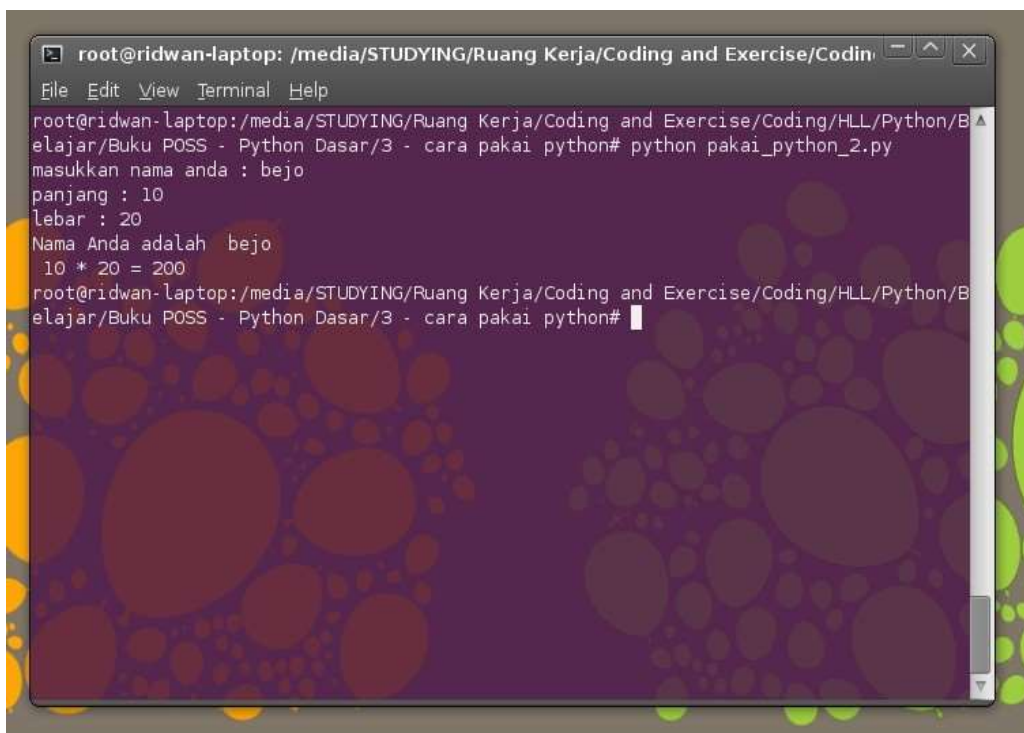
```

nama = raw_input('masukkan nama anda : ') panjang = raw_input("panjang : ")
lebar = raw_input("y : ")

```

```
print "Nama Anda adalah ", nama
luas = int(panjang) * int(lebar)
print " %d * %d = %d" % (int(panjang), int(lebar), luas)
```

Jika kode diatas dieksekusi, maka akan muncul *output* seperti berikut :



```
root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Codin
File Edit View Terminal Help
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/3 - cara pakai python# python pakai_python_2.py
masukkan nama anda : bejo
panjang : 10
lebar : 20
Nama Anda adalah bejo
10 * 20 = 200
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/3 - cara pakai python#
```

<< gambar 2.6 hasil eksekusi pakai_python_2.py >>

input, function ini digunakan untuk menerima *input* sesuai dengan data yang diberikan oleh *user*. Tidak seperti **raw_input** yang menerima *input* dan dianggap *string*. Saat memberikan *input* kepada **raw_input**, Anda tidak perlu menggunakan aturan penulisan untuk tipe data tertentu. Sedangkan di **input** Anda harus mengikuti aturan penulisan untuk memasukkan *input* dari tipe data tertentu. Sebagai contoh dibawah terdapat beberapa contoh aturan penulisan saat akan memberikan data dengan tipe data tertentu kepada **input**.

listing : pakai_python_3.py

```

# meminta input boolean : coba masukkan True variabel_bool = input('masukkan data
boolean : ') print "isi variabel_bool : ", variabel_bool

# meminta input float : coba masukkan 3.14 variabel_float = input('masukkan data float : ')
print "isi variabel_float : ", variabel_float

# meminta input string : coba masukkan “lagi belajar python” variabel_string =
input('masukkan data string : ')
print "isi variabel_string : ", variabel_string

# meminta input octal : coba masukkan 010 variabel_octal = input('masukkan data octal : ')
print "isi variabel_octal : ", variabel_octal

# meminta input hexa : coba masukkan 0x114 variabel_hexa = input('masukkan data hexa :
') print "isi variabel_hexa : ", variabel_hexa

# meminta input long : coba masukkan 123456789123456789L variabel_long =
input('masukkan data long : ')
print "isi variabel_long : ", variabel_long

# meminta input dictionary : coba masukkan {'nama':'luffy', 'hobi':'makan', 'asal':'east blue'}
variabel_dict = input('masukkan data dictionary : ')
print "isi variabel_dict : ", variabel_dict

# meminta input list : coba masukkan [1, 2, 3, 4, 5] variabel_list = input('masukkan data list
: ')
print "isi variabel_list : ", variabel_list

# meminta input tuple : coba masukkan (1, 2, 3, 4, 5) variabel_tuple = input('masukkan
data tuple : ')
print "isi variabel_tuple : ", variabel_tuple

```

Jika kode diatas dieksekusi, akan muncul *output* seperti berikut :



```

root@ridwan-laptop: /media/STUDYING/Ruang Kerja/Coding and Exercise/Codin
File Edit View Terminal Help
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/3 - cara pakai python# python pakai_python_3.py
masukkan data boolean : True
isi variabel_bool : True
masukkan data float : 3.14
isi variabel_float : 3.14
masukkan data string : "lagi belajar python"
isi variabel_string : lagi belajar python
masukkan data octal : 010
isi variabel_octal : 8
masukkan data hexa : 0x114
isi variabel_hexa : 276
masukkan data long : 123456789123456789L
isi variabel_long : 123456789123456789
masukkan data dictionary : {'nama':'luffy', 'hobi':'makan', 'asal':'east blue'}
isi variabel_dict : {'hobi': 'makan', 'nama': 'luffy', 'asal': 'east blue'}
masukkan data list : [1, 2, 3, 4, 5]
isi variabel_list : [1, 2, 3, 4, 5]
masukkan data tuple : (1,2,3,4,5)
isi variabel_tuple : (1, 2, 3, 4, 5)
root@ridwan-laptop:/media/STUDYING/Ruang Kerja/Coding and Exercise/Coding/HLL/Python/B
elajar/Buku POSS - Python Dasar/3 - cara pakai python#

```

<< gambar 2.7 hasil eksekusi pakai_python_3.py >>

Contoh diatas memperlihatkan sebuah perbedaan penggunaan **raw_input** dengan **input**. Data yang didapat dari **raw_input** harus dikonversikan dengan *built-in function* untuk tipe data tertentu. Sedangkan data yang didapat dari **input** tidak perlu dikonversikan, tapi saat memasukkan data harus mengikuti aturan penulisan untuk tipe data tertentu.

Hal Lain yang Harus Diingat dalam Penggunaan Python

Selain itu terdapat beberapa karakter khusus yang dinamakan *escape character*. Berikut adalah daftar beberapa *escape character* yang terdapat di Python :

<i>Escape Character</i>	Heksadesimal	Keterangan
\a	0x07	bel
\b	0x08	<i>backspace</i>
\f	0x0c	<i>formfeed</i>
\e	0x1b	<i>escape</i>
\n	0x0a	<i>newline</i>
\t	0x09	<i>tab</i>
\v	0x0b	<i>Vertical tab</i>
\r	0x0d	<i>Carriage return</i>
\nnn		Notasi oktal, dimana n

<i>Escape Character</i>	Heksadesimal	Keterangan
<code>\xnn</code>		merupakan rentang angka dari 0 sampai 7 Notasi heksadesimal, dimana n merupakan rentang dari 0..9, a..f, atau A..F

Pada kode diatas *listing* pakai_python_1.py, terdapat sebuah simbol `%d` di dalam perintah **print**. Simbol tersebut dinamakan *string formatter* yang berfungsi untuk mencetak data sesuai dengan format yang diinginkan pada *string* yang disisipi simbol tersebut. Berikut adalah daftar beberapa *string formatter* yang disediakan Python:

Simbol	Keterangan
<code>%c</code>	Mencetak karakter
<code>%s</code>	Mencetak data dari jenis apapun menjadi string
<code>%i, %d</code>	Mencetak angka desimal bertanda
<code>%u</code>	Mencetak angka desimal tak bertanda
<code>%o</code>	Mencetak angka oktal
<code>%x, %X</code>	Mencetak angka heksa dengan huruf kecil, Mencetak angka heksa dengan huruf besar
<code>%f</code>	Mencetak angka real berkoma
<code>%e, %E</code>	Mencetak tanda eksponensial dengan huruf kecil, mencetak tanda eksponensial dengan huruf besar
<code>%g, %G</code>	Fungsi hampir sama dengan <code>%f</code> dan <code>%e</code> hanya saja pencetakan angka di belakang koma lebih pendek, pencetakan tanda eksponensial menggunakan huruf besar

Kemudian tak lupa kalau di Python sendiri saat sedang menggunakan *interpreter prompt*, Anda bisa menggunakan *function* **help()** untuk melihat struktur sebuah objek atau perintah – perintah di Python. Misal Anda bisa melihat bantuan tentang perintah **print** maka Anda harus mengetikkan:

```
>> help('print')
```

The ``print`` statement

```
print_stmt ::= "print" ([expression ("," expression)* [","]]
| ">>" expression [("," expression)+ [","]])
```

``print`` evaluates each expression in turn and writes the resulting object to standard output (see below). If an object is not a string, it is first converted to a string using the rules for string conversions. The (resulting or original) string is then written. A space is written before each object is (converted and) written, unless the output system believes it is positioned at the beginning of a line. This is the case (1) when no characters have yet been written to standard output, (2) when the last character written to standard output is a whitespace character except ``'``, or (3) when the last write operation on standard output was not a ``print`` statement. (In some cases it may be functional to write an empty string to standard output for this reason.)

Note: Objects which act like file objects but which are not the built-in file objects often do not properly emulate this aspect of the file object's behavior, so it is best not to rely on this.

A ``\n`` character is written at the end, unless the ``print`` statement ends with a comma. This is the only action if the statement contains just the keyword ``print``.

Standard output is defined as the file object named ``stdout`` in the built-in module ``sys``. If no such object exists, or if it does not have a ``write()`` method, a ``RuntimeError`` exception is raised.

``print`` also has an extended form, defined by the second portion of the syntax described above. This form is sometimes referred to as "``print`` chevron." In this form, the first expression after the ``>>`` must evaluate to a "file-like" object, specifically an object that has a ``write()`` method as described above. With this extended form, the subsequent expressions are printed to this file object. If the first expression evaluates to ``None``, then ``sys.stdout`` is used as the file for output.

(END)

Untuk keluar dari mode bantuan tersebut tekan tombol "q". Sekarang kita coba lihat bantuan mengenai struktur data list:

```
>> help('list')

Help on class list in module _builtin_:

class list(object)
| list() -> new empty list
| list(iterable) -> new list initialized from iterable's items
|
| Methods defined here:
|
|_add_(...)
|     x._add_(y) <==> x+y
|
|_contains_(...)
|     x._contains_(y) <==> y in x
|
|_delitem_(...)
|     x._delitem_(y) <==> del x[y]
|
|_delslice_(...)
|     x._delslice_(i, j) <==> del x[i:j]
|
|     Use of negative indices is not supported.
|_eq_(...)
|_[:]
```

Dengan demikian sekalipun tidak ada koneksi internet, Anda tetap bisa terus membuat program Python dengan dibantu **help()** yang sudah disediakan oleh Python.

Tipe data yang terdapat pada kode – kode diatas akan dijelaskan lebih lanjut pada bab berikutnya.