MODUL
PRAKTIKUM DASAR PEMROGRAMAN
DENGAN BAHASA PYTHON

MODUL
PRAKTIKUM DASAR PEMROGRAMAN
DENGAN BAHASA PYTHON

DAFTAR ISI

Pertemuan 1: Pengantar Pemrograman dan Struktur Bahasa Python

Standar Kompetensi:
Memahami ruang lingkup dan tools yang digunakan dalam pemrograman

Sub Pokok Bahasan:
1. Ragam Bahasa Pemrograman
2. Pemrograman Visual dan Console
3. Interpreter dan Compiler
4. Terminologi Python

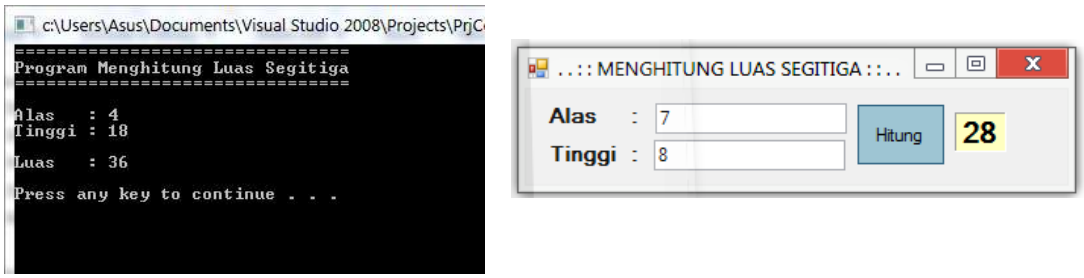
1.1. Ragam Bahasa Pemrograman.

Gambar 1.1 Beberapa bahasa pemrograman

Sebuah program dapat diartikan kumpulan instruksi-instruksi yang dibuat secara terstruktur dan logis untuk menyelesaikan permasalahan. Sebuah masalah memiliki makna keadaan yang tidak sesuai dengan kenyataan. Tanpa permasalahan maka tidak akan ada program.

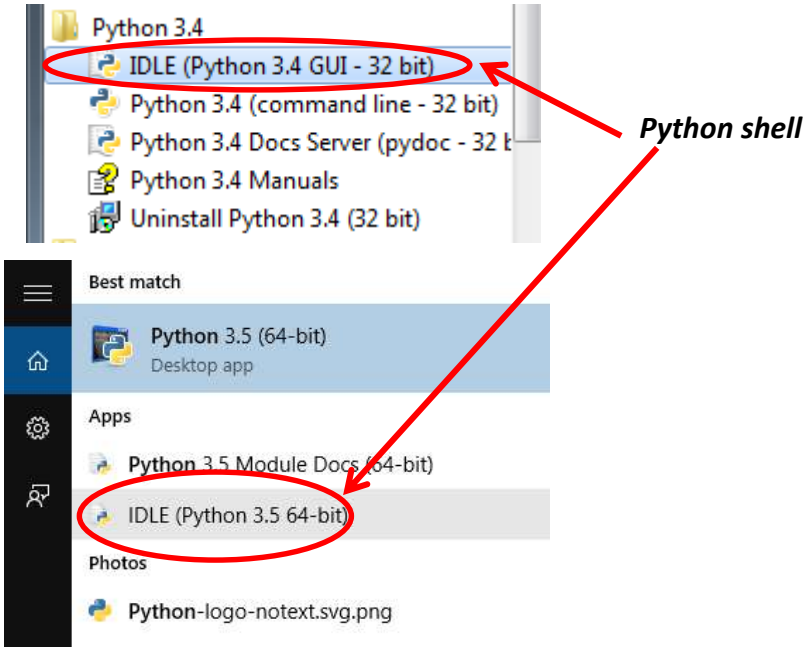
Seorang pembuat program disebut dengan programmer harus memiliki kemampuan membuat program berdasarkan ketentuan masing-masing bahasa pemrograman yang digunakan. Gambar 1.1 memperlihatkan beberapa jenis bahasa pemrograman seperti C, php, java, dan python, selain itu bahasa lainnya seperti basic, pascal, cobol, dan lain-lainnya.

1.2. Pemrograman *Visual* dan *Console*



Gambar 1.2. Tampilan Visual (sebelah kanan) dan Console (sebelah kiri)

Pada tampilan visual pada contoh dengan menampilkan textbox dan tombol yang dapat di isi dan di klik, tampilannya lebih menarik dan nyaman digunakan. Sementara tampilan Console lebih sederhana, terlihat kurang begitu menarik dan monoton.



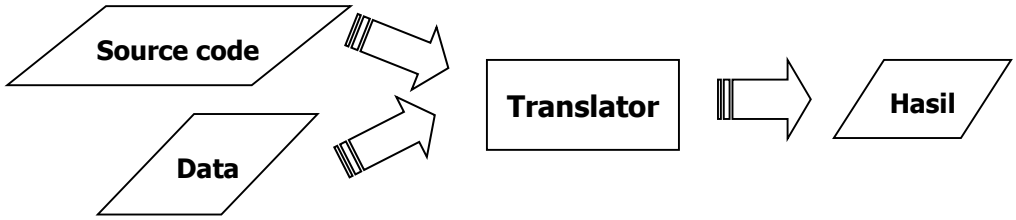
Gambar 1.3. Tools Python Shell 32 bit dan 64 bit.

Gambar 1.3 memperlihatkan tampilan IDLE (Integrated Development and Learning Environment) yaitu Python sebagai lingkungan belajar berisi tampilan GUI yang menarik, bekerja pada OS (Windows, Linux dan Mac OS X), interaktif interpreter (penterjemah) berupa kode *input/output* dan *error messages*, *multi windows*, *multiple file (grep)* berupa

search within any windows, fiture debugger (pencari kesalahan), konfigurasi/*browsers* dan dialog.

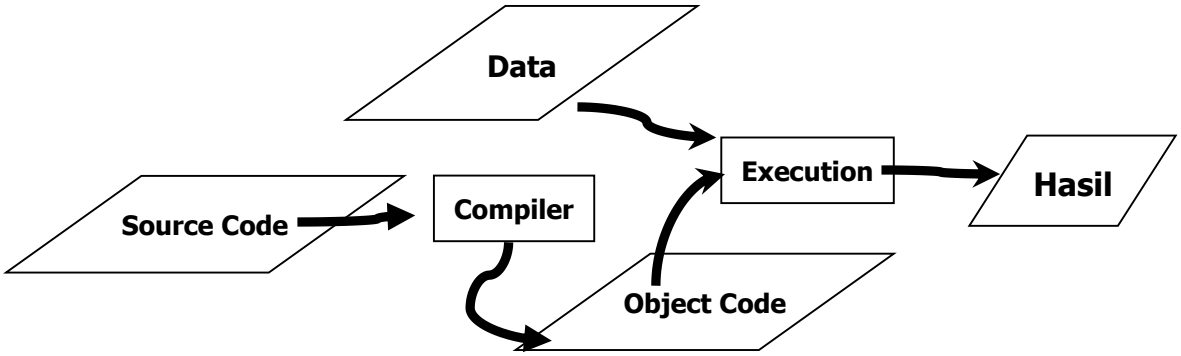
1.3. *Interpreter* dan *Compiler*

Penterjemah bahasa python menggunakan interpreter (satu per-satu pernyataan), berbeda dengan penterjemah compiler yang menterjemahkan kode program sekaligus (blok pernyataan). Interpreter: Interpreter tidak menghasilkan bentuk *object code*, tetapi hasil translasinya hanya dalam bentuk internal, dimana program induk harus selalu ada-berbeda dengan *compiler*.



Gambar 1.4 Skema proses interpreter

Compiler : *Source code* adalah bahasa tingkat tinggi, *object code* adalah bahasa mesin atau bahasa assembly. *Source code* dan data diproses secara berbeda.

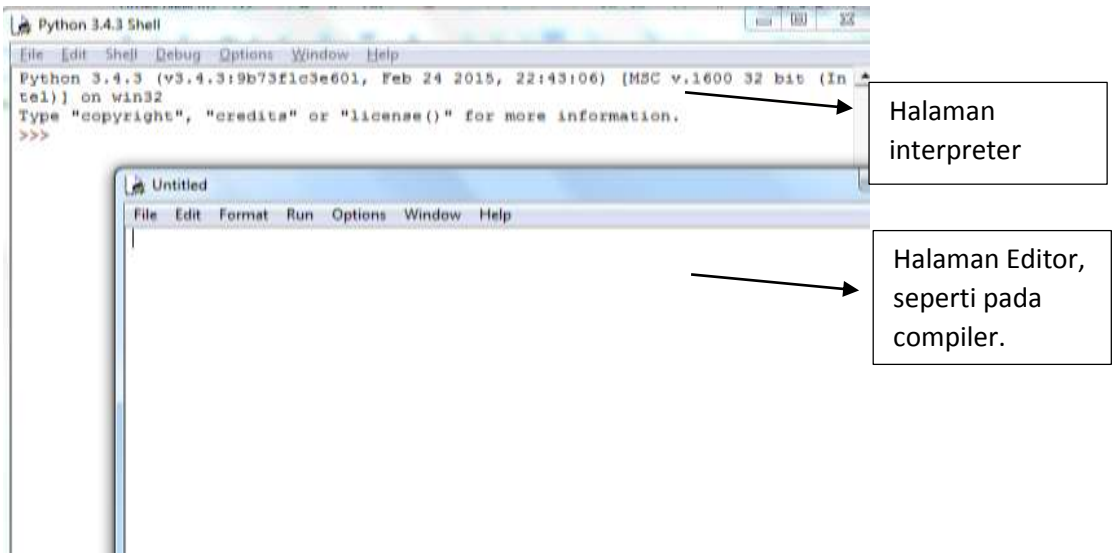


Gambar 1.5 Skema proses Compiler

1.4.Terminologi Python

Bahasa Python memiliki beberapa sintaks yang umum ada pada bahasa pemrograman lainnya seperti input/output proses, struktur seleksi, struktur pengulangan, pernyataan fungsi (sub program), dan lain sebagainya.

A. Halaman Pengetikan Kode Program.

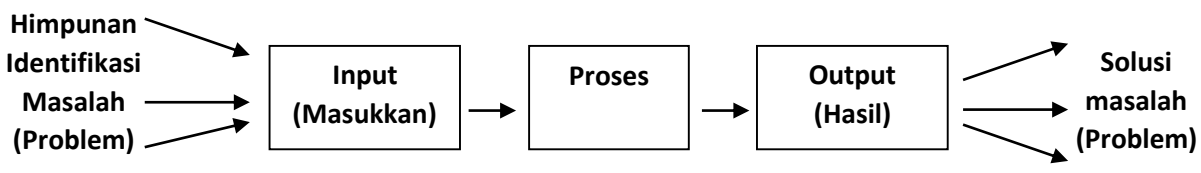


Gambar 1.6 Halaman Pengetikan Kode Program.

Gambar 1.6 menunjukkan halaman pengetikan kode program dengan satu persatu argumen/ pernyataan, atau dengan editor seperti layaknya kode pada *compiler* yang ditulis sekaligus tetapi tetap dieksekusi argumen per argumen.

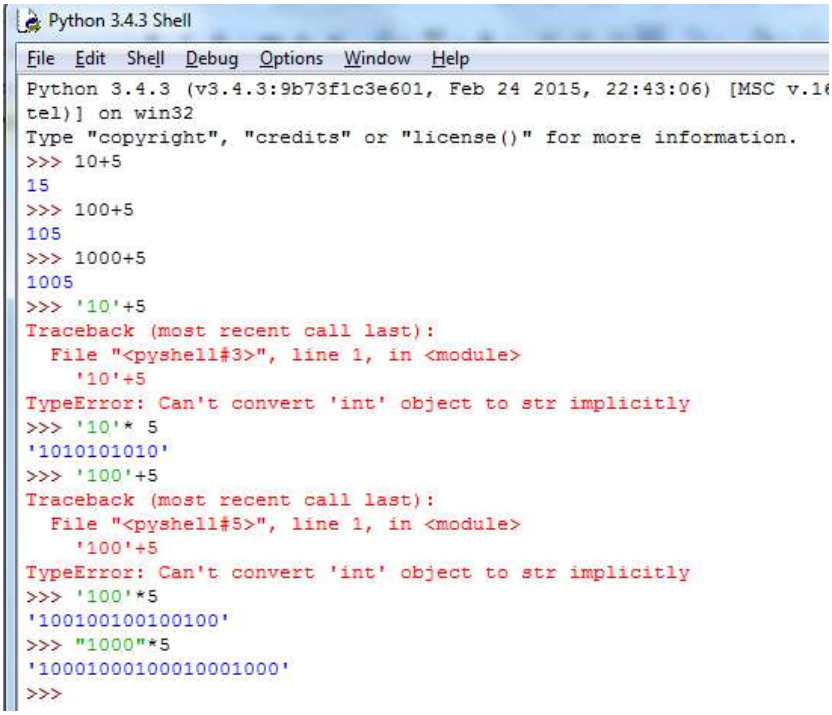
B. Nilai (*value*) dan Tipe data

Nilai (*value*) adalah hal yang paling mendasar seperti sebuah huruf, karakter khusus, atau sebuah angka yang akan dimanipulasi oleh program.



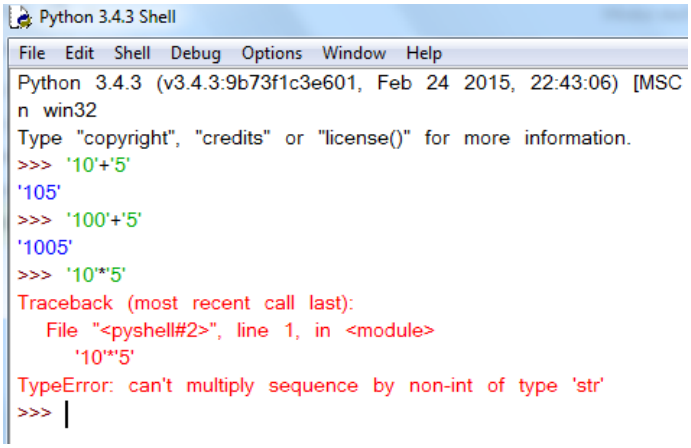
Gambar 1.7 Sistematis Pemecahan Masalah

Nilai angka seperti : angka 10, 100,1000 jika dijumlah dengan angka 5 menghasilkan angka 15,105, dan 1005. Jika di kali dengan 5 maka memberikan hasil angka 50, 500, dan 5000. Bagaimana jika angka tersebut berubah menjadi ‘10’, ‘100’, dan ‘1000’. Bagaimana jika dijumlah dengan angka 5? Bagaimana jika dikali dengan angka 5?. Perhatikan Gambar 1.8 berikut ini.



Gambar 1.8 Tampilan contoh operasi angka dan bukan angka

Selanjutnya, jika bukan angka dioperasikan sesamanya bagaimana?



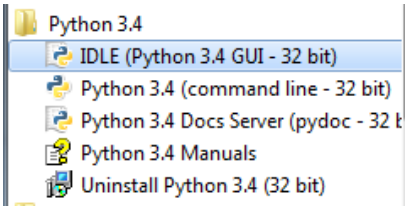
Gambar 1.9 Tampilan contoh operasi bukan angka.

Pada Gambar 1.9 menampilkan tipe data yang bukan angka jika dioperasikan sesamanya akan membuat value yang baru. Sehingga dalam membaut program data (value) selalu tergantung pada tipe data yang digunakan. Tipe data yang cocok akan membuat data tersimpan dalam memori komputer secara realtime sesaat sebelum proses selanjutnya. Tipe data yang digunakan dalam bahasa Python yaitu : tipe data sederhana seperti integer (bilangan bulat), float (bilangan berkoma), string (abjad bisa huruf, angka, atau karakter khusus). Tipe data majemuk seperti list.

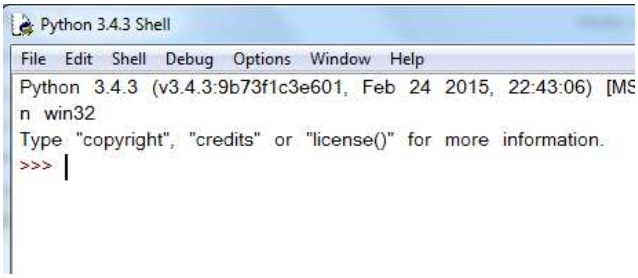
C. Mengaktifkan Halaman Bahasa Python

Berikut ini langkah-langkah untuk menjalankan dan membuka halaman bahasa Python.

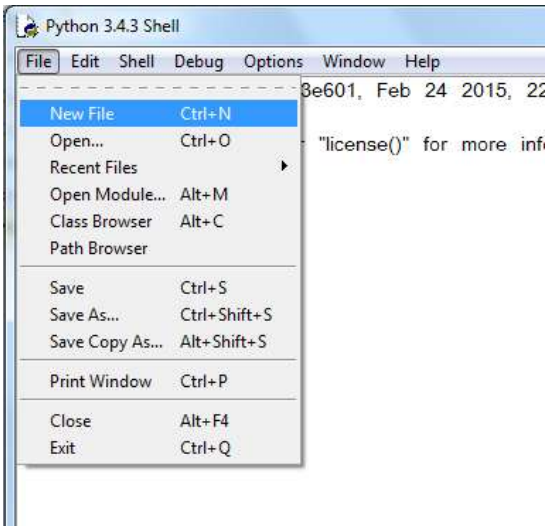
<u>Langkah 1</u>: Pilih IDLE.



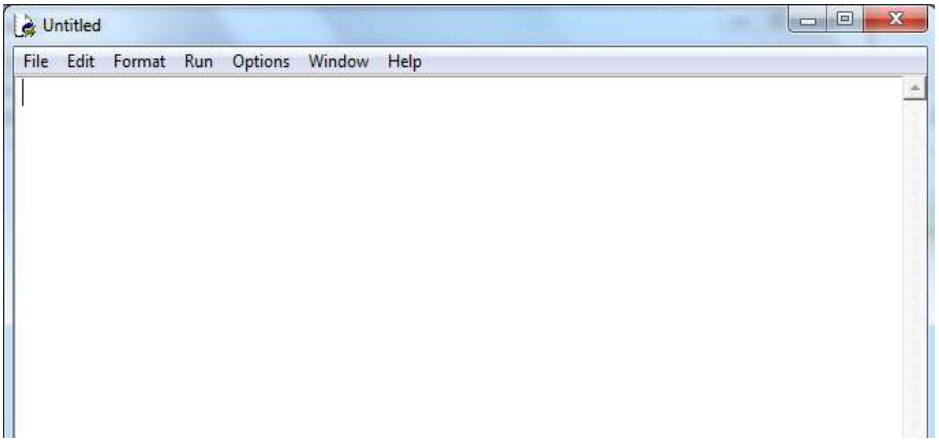
<u>Langkah 2</u>: Tampilan Halaman setelah IDLE dipilih.



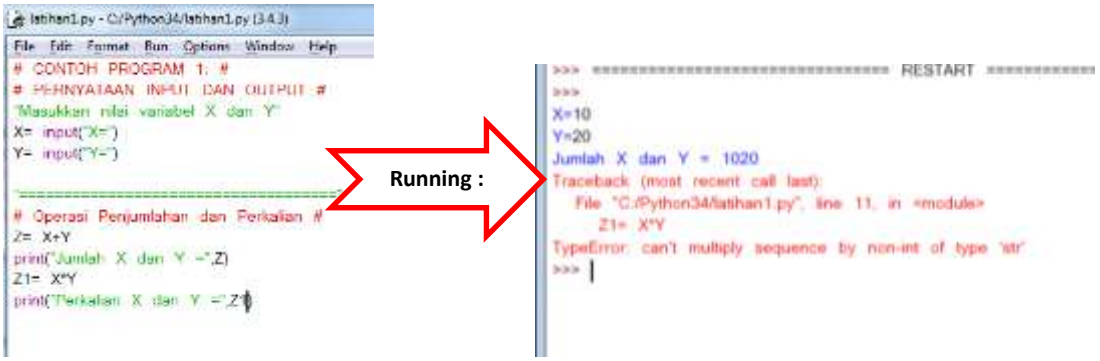
<u>Langkah 3</u>: Tampilan menu utama pada File.



<u>Langkah 3</u>: Tampilan Editor kode program yang bukan perbaris peng-eksekusian.



D. Input dan Output Bahasa Python.



Gambar 1.10 Tampilan *Error* Contoh Operasi 2 Bilangan.

Gambar 1.8 menunjukkan input bilangan pada variabel X dan Y, tetapi hasil yang diperoleh dalam bentuk tipe data string (Var = Input("….")), sehingga tidak dapat dihitung hasil perkaliannya. Untuk mengatasi persoalan tersebut gunakan **eval( )** seperti pada Gambar 1.9 berikut.

Gambar 1.11 Tampilan Operasi 2 Bilangan.

LATIHAN: Buatlah sebuah program untuk masukkan 5 buah bilangan dan menghitung rata-rata 5 buah bilangan tersebut!

Pertemuan 2 : *Identifier*, Variabel, dan Tipe Data.

Standar Kompetensi:
Memahami perintah-perintah dasar Python untuk *Identifier*, Variabel, dan Tipe Data.
Sub Pokok Bahasan:

1. Membuat variabel
2. Memberikan nilai ke dalam variabel
3. Mencetak nilai dalam variabel
4. Separator, tipe data, fungsi *type*

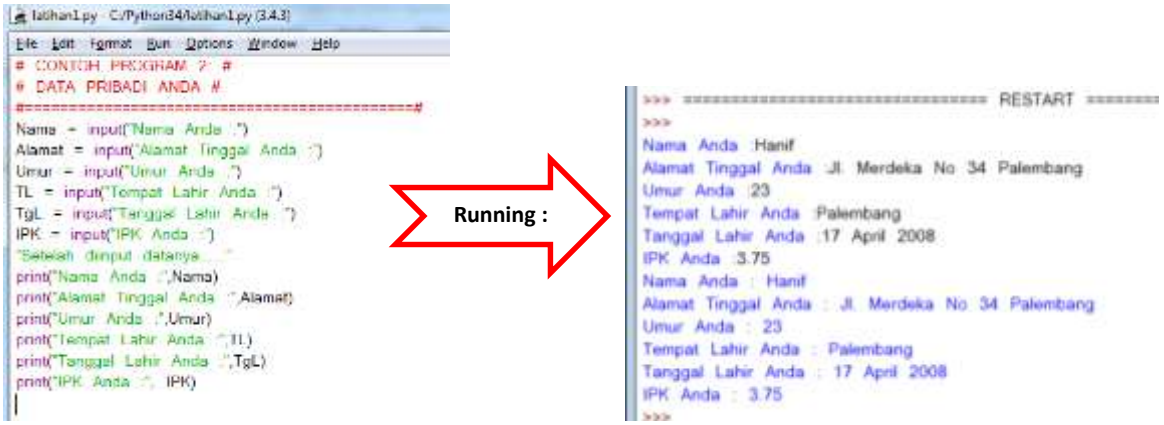
2.1 Membuat Variabel

Variabel atau peubah memiliki pengertian sembarang symbol yang dapat dimuati oleh sembarang himpunan bilangan. Dalam pengertian komputasi sebuah nama yang digunakan untuk menyimpan nilai dengan kapasitas tertentu dan alamat tertentu dalam memori komputer. Variabel merupakan pendaftaran tipe data bagi variabel, konstanta dan parameter yang digunakan sebuah program agar mempunyai alamat penyimpanan dan kapasitas data dalam memori komputer.

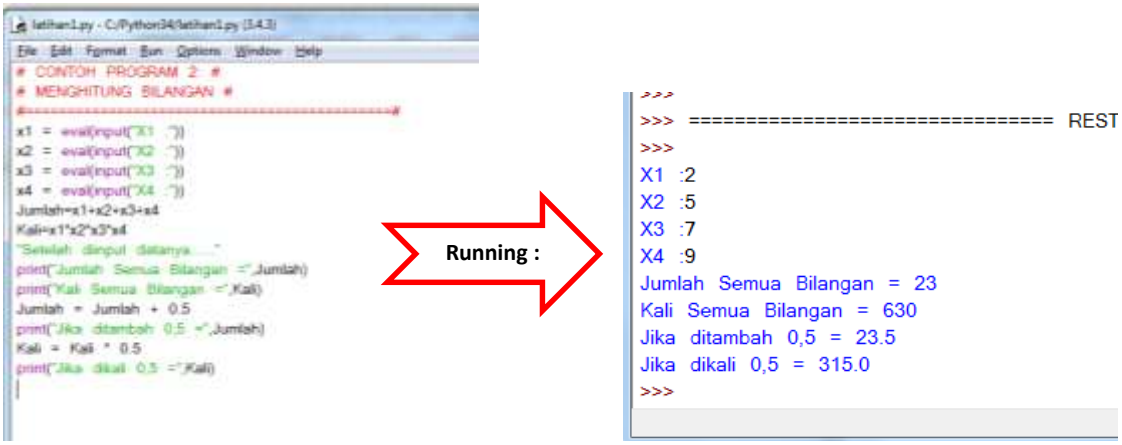
Dalam membuat variabel hindari spasi dan menggunakan karakter khusus, selain itu juga nama dalam kata cadangan Python (seperti **input**, **eval**, **if**, **elif**, **for**, **def**, dan lain-lain) tidak dapat menjadi variabel.

Penempatan Variabel pada yang semestinya.

Misalkan sebuah data pribadi berisi nama, alamat, umur, tempat lahir, tanggal lahir, indeks prestasi kumulatif akan memberikan 6 (enam) buah variabel dengan tipe datanya.



Gambar 2.1 Tampilan Contoh *Input/ Output* Tipe Data String



Gambar 2.2 Tampilan Contoh *Input/ Output* Tipe Data Bilangan

Pada Gambar 2.2 terlihat input/output pada tipe data bilangan dengan hasil yang berbeda tipe bilangannya yaitu tipe integer (bilangan bulat) atau float (bilangan berkoma).

2.2 Memberikan nilai ke dalam variabel

Lakukan inisiasi variabel atau konstanta dari permasalahan berikut! Menjumlahkan total harga pada saat konsumen membeli beberapa barang.

Langkah 1: Inisiasi Persoalan

Variabel/ konstanta input :

kode_barang, nama_barang, harga_satuan_barang,
jumlah_per_barang_beli, total_harga_per_transaksi = 0

Proses : harga_beli_per_barang = harga_satuan_barang * jumlah_per_barang_beli
total_harga_per_transaksi=harga_beli_per_barang + total_harga_per_transaksi

Output : total_harga_per_transaksi

Langkah 2: Menetapkan Tipe Data

kd_brg, nama_brg bertipe data *string*

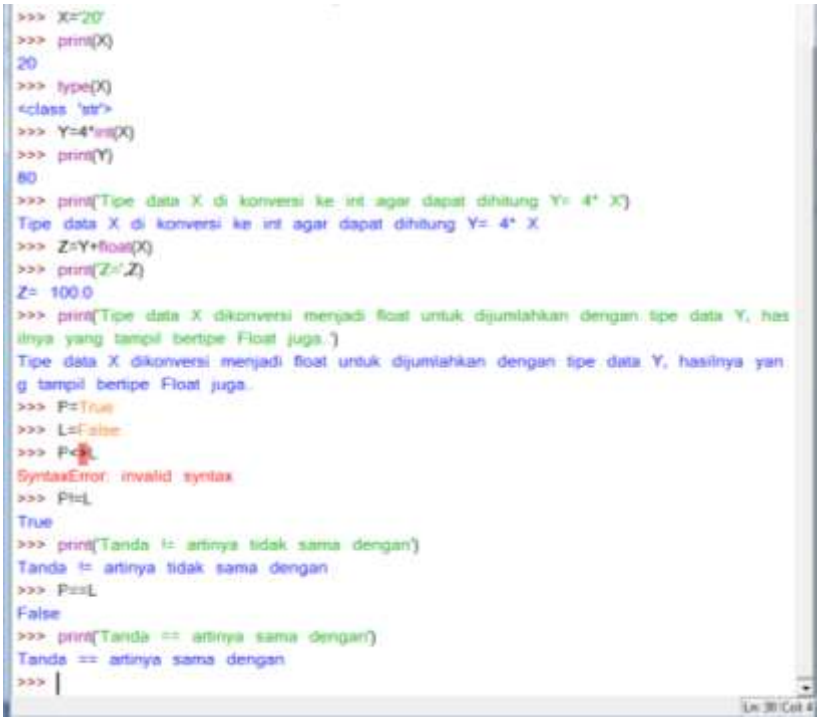
jum_brg bertipe data *integer*

harga_satuan, harga_beli, total_hrg_brg bertipe data *float*

Langkah 3 : Kode program

2.3 Mencetak nilai dalam variabel

Mencetak nilai dalam sebuah variabel menggunakan pernyataan **print,** perhatikan contoh berikut ini.



Gambar 2.3 Tampilan Contoh Konversi Tipe Data String dan Integer

2.4 Separator, tipe data, fungsi *type*

konversi type data pada pemrograman python gunakan fungsi berikut :

str() = Untuk konversi type data ke String

int() = Untuk konversi type data ke Integer

float() = Untuk konversi type data ke Float

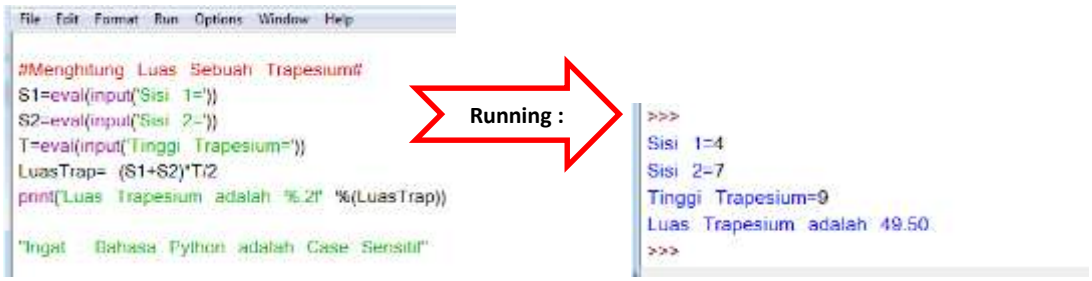
Ada dua macam variasi print :

1. Jika ada simbol, gunakan kutip dua atau gunakan backslash (\) sebelum menuliskan simbol
2. Dipisahkan dengan tanda koma
3. Diganti dengan :
    - %d : mewakili integer
    - %f : mewakili float
        - Untuk membuat n angka di belakang koma, gunakan %.nf
        - Misal untuk dua angka di belakang koma, berarti gunakan %.2f
    - %s : mewakili string

```
>>> Kode='AE107'
>>> NamaBarang='Kaca Mata'
>>> HargaSatuan=125000
>>> Stok=10
>>> print('Kode barang= %s \n Nama Barang= %s \n Harga Satuan=Rp. %d \n Stok Barang= %d' %(Kode,NamaBarang,HargaSatuan,Stok))
Kode barang= AE107
 Nama Barang= Kaca Mata
 Harga Satuan=Rp. 125000
 Stok Barang= 10
>>> HargaBarang=float(HargaSatuan)
>>> print('Harga Satuan Barang= Rp. %.3f')
Harga Satuan Barang= Rp. %.3f
>>> print('Harga Satuan Barang= Rp. %.3f' %(HargaBarang))
Harga Satuan Barang= Rp. 125000.000
>>>
```

Gambar 2.4 Tampilan Contoh print Tipe Data String, Integerm dan Float

Perhatikan Contoh Program berikut ini.



Gambar 2.5 Contoh Luas Trapesium

Perhatikan Contoh Program berikut ini.

Gambar 2.6 Contoh Menghitung Luas Tabung dengan fungsi Math

LATIHAN :

Soal 1: Menghitung rata-rata dari 3 bilangan bulat sembarang.

Soal 2: Menghitung nilai fungsi $f(x) = 2x^3 + 2x + 15/x$ jika x merupakan bilangan bulat sembarang.

Soal 3: Melakukan tukar nilai A,B,C,D menjadi B,D,A,C jika A,B,C,D merupakan bilangan desimal sembarang

Pertanyaan :

a. Tentukan input/output proses untuk masing-masing soal!
b. Buatlah *source code* yang utuh untuk menyelesaikan permasalahan masing-masing soal pada (a)!
c. Jalankan *source code* pada (b), temukan dan selesaikan kesalahan yang terjadi!

Pertemuan 3 : Pernyataan Berkondisi

Standar Kompetensi:
Memahami konsep dan penerapan pernyataan berkondisi dan pernyataan berkondisi bersarang dengan penggunaan operator.

Sub Pokok Bahasan:

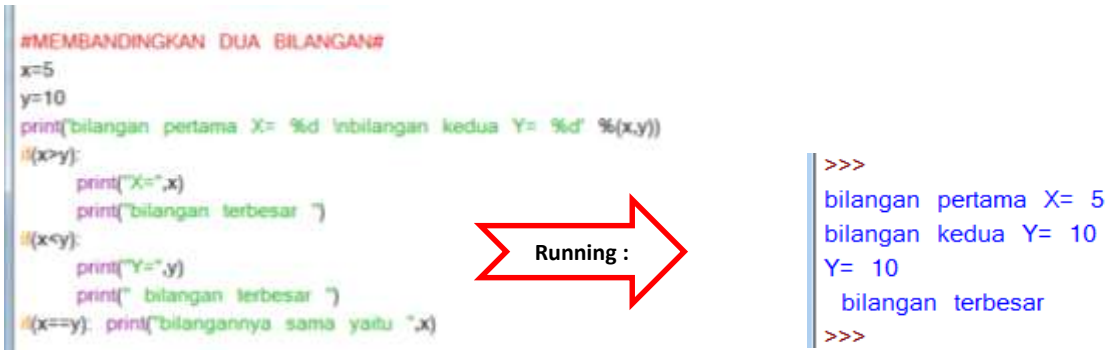
1. Pernyataan Berkondisi
2. Struktur Pernyataan Berkondisi
3. Perintah Pernyataan Berkondisi
4. Perintah Pernyataan Berkondisi Bersarang

3.1 Pernyataan Berkondisi

Pernyataan berkondisi digunakan untuk membuat alur program agar menjadi lebih efektif dan relevan terhadap permasalahan yang ada. Dalam bahasa Python pernyataan berkondisi *if* yang digunakan dengan. Pernyataan *if* juga digunakan untuk memberikan beberapa alternatif pilihan seperti bilangan genap atau ganjil, jenis kelamin perempuan atau laki-laki, golongan dan pangkat jabatan, dan lain sebagainya.

Contoh soal berkaitan dengan *If* :

Buatlah program untuk menentukan bilangan terbesar dari dua bilangan!



Gambar 3.1 Kode Program Membandingkan 2 Buah Bilangan.

Setiap pernyataan selalu berakhir dengan tanda karakter ‘;’. Setiap menggunakan **blok** pernyataan selalu mempunyai >1 pernyataan. Untuk setiap ekspresi_kondisi yang dibuat akan mempunyai nilai TRUE atau FALSE sebagai nilai akhirnya, dan bukan berupa nilai numerik. Setiap ekspresi_kondisi akan menggunakan operator logika sebagai pembanding seperti tanda ‘<’, ‘>’, ‘<=’, ‘>=’, ‘!=’, dan ‘==’.

Perhatikan bahwa dalam bahasa Python tidak memiliki blok pernyataan seperti bahasa C++, Basic, atau Pascal.

3.2 Struktur Pernyataan Berkondisi

Struktur Kondisi **if** dapat digunakan dalam beberapa bentuk sebagai berikut:

```
if (kondisi) :
        pernyataan_1
        pernyataan_2
            ::::
        pernyataan_N
else :
        pernyataan
```

Contoh permasalahan untuk menentukan Diskriminan dari Persamaan Kuadrat:

```
>>> a=3
>>> b=5
>>> c=1
>>> D=b*b-4*a*c
>>> if(D<0): print('Akar Imaginer')
elif(D>0):print('Akar Nyata Berlainan')
else: print('Akar Kembar')

Akar Nyata Berlainan
>>>
```

Gambar 3.2 Contoh Kode Program Kriteria Umur.

Pada contoh (Gambar 3.2) memperlihatkan kode program di ketik berdasarkan prompt yang aktif, ketika Enter maka kode dijalankan. Ketik **`if(D<0):print('Akar Imaginer')`** lalu tekan Enter 1 kali, kemudian ketik **`elif(D>0):print('Akar Nyata Berlainan')`** lalu tekan Enter 1 kali lagi, kemudian ketik lagi **`else:print('Akar Kembar')`** lalu tekan Enter 2 kali. Untuk kata cadangan **elif** sama dengan **else if**.

Operator Dalam Bahasa Python

Ada tiga buah **operator logika** yang umum digunakan yaitu and, or, dan not, Operator tersebut digunakan untuk membangun ekspresi dalam tipe data Boolean. Semantic (arti / maksud) dari operator-operator tersebut adalah mirip dengan artinya dalam bahasa Inggris. Seperti x > 0 and x < 10 ketika di jalankan memberikan tampilan True jika x lebih besar dari 0 pada saat yang sama x juga bernilai lebih kecil dari 10. Contoh lainnya `n % 2 == 0 or n % 3 == 0` adalah **True** jika *salah satu* dari kondisi tersebut bernilai True (jika n bisa dibagi dengan bilangan 2 atau 3). Operator **not** digunakan untuk menegasi nilai Boolean (invers nilai) seperti **not (x > y)** bernilai True jika (x > y) adalah False artinya bilangan x kurang dari y.

3.3 Perintah Pernyataan Berkondisi

Struktur Kondisi **if** dapat digunakan dalam beberapa bentuk sebagai berikut:

```
if (kondisi1) :
        pernyataan_1
        pernyataan_2
              ::::
        pernyataan_N
elif(kondisi2):
        pernyataan_1
        pernyataan_2
              ::::
        pernyataan_N
```

Contoh permasalahan untuk menentukan Kriteria Umur :

Buatlah program untuk menyeleksi kriteria umur jika!

umur<=5 maka Kriterianya Balita

5<umur<=13 maka Kriterianya Anak-anak

13<umur<=25 maka Kriterianya Remaja

25<umur<=35 maka Kriterianya Dewasa

35<umur<=55 maka Kriterianya Orang Tua

umur>55 maka Kriterianya Lansia

Source Code Penyelesaian:

Gambar 3.3 Contoh Kode Program Kriteria Umur.

3.4 Perintah Pernyataan Berkondisi Bersarang

Struktur Kondisi **if** dapat digunakan dalam beberapa bentuk sebagai berikut:

```
if (kondisi1) :
        pernyataan_1
        pernyataan_2
        ::::
        if(kondisi2):
                pernyataan_1
                ::::
                elif(kondisi3):
                        pernyataan_1
                        pernyataan_2
                        ::::
                        pernyataan_N
                pernyataan_2
                ::::
                pernyataan_M
        ::::
        pernyataan_K
```

Source Code Contoh if Bersarang:

```
#PERINTAH IF BERSARANG#
Nama= input('Nama Karyawan:')
NIK= input('NIK Karyawan:')
Status= input('Status (Staff/Honor):')
Gol= input('Golongan (A/B/C):')
if(Status=='Staff'):
        Gaji=1000000
        if(Gol=='A'):
                Bonus=200000
        elif(Gol=='B'):
                Bonus=400000
        elif(Gol=='B'):
                Bonus=550000
elif(Status=='Honor'):
        Gaji=750000
        if(Gol=='A'):
                Bonus=150000
        elif(Gol=='B'):
                Bonus=275000
        elif(Gol=='B'):
                Bonus=480000
Gatot = Gaji + Bonus
print('Gaji =Rp.',Gaji)
print('Bonus =Rp.',Bonus)
print('Gaji Total =Rp.',Gatot)
```


```
>>>
Nama Karyawan:SAMSIAH
NIK Karyawan:201589
Status (Staff/Honor):Staff
Golongan (A/B/C):B
Gaji =Rp. 1000000
Bonus =Rp. 400000
Gaji Total =Rp. 1400000
>>>
```

Gambar 3.4 Contoh kode program if bersarang

LATIHAN

1. Buatlah program dalam struktur control *If* untuk menyeleksi kriteria nilai jika diketahui informasi sebagai berikut!

   Nilai>=88 kriteria A

   77<=Nilai<88 kriteria B

   60<=Nilai<77 kriteria C

   45<=Nilai<60 kriteria D

   Nilai<45 kriteria E

Pertemuan 4 : Pernyataan Perulangan

Standar Kompetensi:
Memahami konsep dan penerapan perulangan dengan menggunakan **for**.

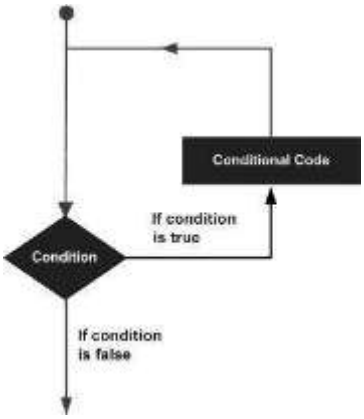
Sub Pokok Bahasan:

1. Struktur Perulangan For
2. Range
3. Perintah perulangan dengan for
4. Kombinasi For Dengan Pemilihan

4.1 Struktur Pengulangan *For*

Diperlukan perulangan dalam menulis suatu baris program, tujuan utamanya adalah agar tidak menuliskan perintah tersebut secara berulang-ulang, hanya perlu menentukan variabel-variabel yang mengalami perubahan kemudian memasukan data untuk di iterasi dengan menggunakan perintah Python, dalam Modul ini hanya membahas perintah **for**.

Perulangan maksudnya adalah mengeksekusi perintah tertentu secara berulang-ulang dan jumlah perulangan dapat diatur sendiri sesuai dengan keinginan. Perintah for dalam python mempunyai ciri khas tersendiri dibandingkan dengan bahasa pemrograman lain. Tidak hanya mengulang bilangan-bilangan sebuah ekspresi aritmatik, atau memberikan keleluasaan dalam mendefinisikan iterasi perulangan dan menghentikan perulangan padasaat kondisi tertentu. Dalam python, statemen **for** bekerja mengulang berbagai macam tipe data yang sekuensial seperti *List*, *String*, dan *Tuple*. Bentuk perulangan **for** akan dijalankan selama kondisi bernilai TRUE dan akan keluar dari perulangan jika kondisi FALSE yang dapat digambarkan menggunakan *flowchart* berikut ini (Gambar 4.1).



Gambar 4.1. Struktur Perulangan *FOR*

For Tunggal

Struktur penulisan :

```
forvariabel in iterable :
     pernyataan 1
     pernyataan 2
     dan seterusnya
```

4.2 Range

- `range(nilai_awal,nilai_akhir,pencacah)`
- `range(nilai_awal,nilai_akhir)`
- `range(nilai_akhir)`

`Iterable dapat diisi dengan variabel, list, dan range`

Contoh penggunaan `range(nilai_awal,nilai_akhir,pencacah):`

Berikut ini contoh penggunaan range untuk menampilkan bilangan dari 1 – 100 dengan penambahan/pencacah 1 dengan menambahkan **`end=' '`** agar bilangan tampil secara horizontal tidak pindah baris ke bawah

```
for i in range(1,101,1):
    print(i, end=' ')
```

Setelah perintah diatas dijalankan (*run*) maka akan tampil bilangan seperti berikut ini :

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71 72 73 74 75
76 77 78 79 80 81 82 83 84 85 86 87
88 89 90 91 92 93 94 95 96 97 98 99
100
```

Pencacah bilangan untuk bentuk perulangan **for** tidak harus selalu 1, boleh saja dengan bilangan yang lain, seperti contoh berikut ini dimana pencacah bilangan 2 yang menampilkan bilangan dari 1-100 dengan penambahan 2.

```
for i in range(1,101,2):
    print(i, end=' ')
```

Setelah perintah diatas dijalankan (*run*) maka akan tampil bilangan seperti berikut ini :

```
1 3 5 7 9 11 13 15 17 19 21 23 25 27
29 31 33 35 37 39 41 43 45 47 49 51
53 55 57 59 61 63 65 67 69 71 73 75
77 79 81 83 85 87 89 91 93 95 97 99
```

Coba perhatikan bilangan yang ditampilkan! Antara bilangan yang satu dengan bilangan yang berikutnya selisihnya adalah 2 sehingga bilangan yang ditampilkan akan berhenti sampai bilangan 99 bukan 100.

Contoh penggunaan `range(nilai_awal,nilai_akhir):`

Jika pada contoh sebelumnya nilai awal dan nilai akhir telah ditentukan secara langsung pada stuktur **for,** maka pada contoh berikut ini akan memasukkan nilai awal dan nilai akhir secara tidak langsung yaitu dengan menggunakan suatu variabel yang nilainya diinput pada saat programnya dijalankan (*run*). Contoh berikut ini juga tidak menuliskan secara langsung pencacahnya. Jika tidak ditentukan secara langsung pencacahnya maka secara *default* pencacahnya akan bernilai 1.

Berikut ini contoh penggunaan **range** untuk menampilkan dari bilangan tertentu sampai bilangan tertentu dan menghitung banyaknya bilangan serta menghitung jumlah seluruh bilangan yang ada dengan menambahkan **`end=' '`** agar bilangan tampil secara horizontal tidak pindah baris ke bawah. Dari bilangan tertentu sampai bilangan tertentu diinput terlebih dahulu pada saat program dijalankan dengan menggunakan variabel awal dan akhir. Selanjutnya nilai awal dan akhir tadi akan ditampilkan ke layar dengan sebelumnya juga dihitung berapa banyak bilangan yang tampil serta berapa jumlah seluruh bilangan yang ada. Varibel akhir harus ditambah 1 supaya nilai akhir yang diinginkan terpenuhi.

```
awal=int(input('Set Nilai Awal = '))
akhir=int(input('Set Nilai Akhir = '))

count=0
summ=0

print('Bilangan antara %d dan %d' %(awal,akhir))

for i in range(awal,akhir+1):
    print(i, end=' ')
    count=count+1
    summ=summ+i

print('Bilangan di atas ada %d bilangan' %count)
print('Jumlah semua bilangan adalah %d' %summ)
```

Setelah perintah diatas dijalankan (*run*) maka akan tampil bilangan seperti berikut ini :

```
Set Nilai Awal = 10
Set Nilai Akhir = 67
Bilangan antara 10 dan 67
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 2
5 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 5
6 57 58 59 60 61 62 63 64 65 66 67 Bilangan di
atas ada 58 bilangan
Jumlah semua bilangan adalah 2233
```

Coba perhatikan bilangan yang ditampilkan! Antara bilangan yang satu dengan bilangan yang berikutnya selisihnya adalah 1 walaupun tidak dituliskan secara eksplisit karena

merupakan nilai *default*. Dengan adanya penambahan 1 pada variabel akhir maka bilangan yang ditampilkan akan berhenti sampai bilangan 67 jika tidak ditambah dengan 1 maka akan ditampilkan hanya sampai 66.

LATIHAN.

Buatlah program untuk menampilkan bilangan dari bilangan tertentu sampai bilangan tertentu dengan pencacah 3 dan menghitung hasil perkalian untuk semua bilangan yang ada!

Contoh penggunaan `range(nilai_akhir):`

Jika pada contoh sebelumnya nilai awal dan nilai akhir harus telah ditentukan terlebih dahulu secara langsung pada stuktur **for,** maka pada contoh berikut ini hanya memasukkan nilai akhir saja secara langsung maupun secara tidak langsung dengan menggunakan variabel. Dengan hanya memasukkan nilai akhir saja maka secara *default* nilai awal akan bernilai nol dan pencacahnya akan bernilai 1.

Berikut ini contoh penggunaan range untuk menampilkan bilangan dari 0 – 100 dengan menambahkan **`end=' '`** agar bilangan tampil secara horizontal tidak pindah baris ke bawah

```
for i in range(101):
    print(i,end='  ')
```

Perlu diingat bahwa nilai akhir menggunakan operator < bukan ≤ sehingga untuk menampilkan sampai angka 100 nilai akhir harus kita buat menjadi 101

Setelah perintah diatas dijalankan (*run*) maka akan tampil bilangan seperti berikut ini :

```
0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22
  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42
  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100
```

Berikut ini contoh penggunaan range untuk menampilkan sampai bilangan tertentu dengan menggunakan variabel nilai akhir dengan menambahkan **end=' '** agar bilangan tampil secara horizontal tidak pindah baris ke bawah.

```
nilaiakhir = int(input('Input Sampai BIlangan : '))
for i in range(nilaiakhir+1):
    print(i,end='  ')
```

Perlu diingat bahwa nilai akhir harus ditambah dengan 1 supaya memenuhi kondisi nilai akhir yang ingin ditampilkan.

Setelah perintah di atas dijalankan (*run*) maka akan tampil bilangan seperti berikut ini :

```
Input Sampai BIlangan : 70
0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22
  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42
  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
  63  64  65  66  67  68  69  70
```

4.3 Perintah Perulangan Dengan *For* Bertingkat

Secara umum bentuk *for* bertingkat sama saja seperti *for* tunggal, hal saja di dalam bentuk *for* tunggal terdapat lagi bentuk *for* tunggal yang lain.

Struktur Penulisan :

```
for variabel1 in iterable :
    for variabel2 in iterable :
        ….
        for variabelN in iterable :
            pernyataan 1
            pernyataan 2
            dan seterusnya
```

Range :

- range(nilai_awal,nilai_akhir,pencacah)
- range(nilai_awal,nilai_akhir)
- range(nilai_akhir)

Iterable dapat diisi dengan variabel, **list**, dan range

Contoh penggunaan **For Bertingkat** (2 tingkat):

Contoh berikut ini akan menampilkan bilangan dalam bentuk baris dan kolom, dimana untuk baris dinyatakan dengan x dan untuk kolom dinyatakan dengan y, lalu selanjutnya antara baris dan kolom nilainya dilakukan perkalian.

```
for x in range(1, 3):
    for y in range(1, 3):
        print (x, y, x*y)
```

Setelah perintah diatas dijalankan (*run*) maka akan tampil bilangan seperti berikut ini :

```
1 1 1
1 2 2
2 1 2
2 2 4
```

Lebih jauh for bertingkat dapat digunakan untuk menyelesaikan masalah-masalah perhitungan diantaranya perhitungan matriks, perhitungan total pembelian (kuitansi) maupun soal yang lainnya. Berikut ini contoh soal untuk menghitung rata-rata dari M orang mahasiswa dari N buah mata kuliah yang diambil. Untuk menyelesaikan soal seperti ini maka banyaknya mahasiswa (M) diset sebagai baris (for level1) dan banyaknya mata kuliah (N) yang diambil sebagai kolom (for level 2). Adapun perintahnya adalah sebagai berikut:

```
banyakMhs = int(input('Banyak Mahasiswa : '))
for M in range (banyakMhs):
    print('Mahasiswa ke-',M+1)
    banyakMK  = int(input('Banyak Matakuliah yang diambil : '))
    totalnilai=0
    for N in range(banyakMK):
        nilai=int(input('Input Nilai ke %d :' %(N+1)))
        totalnilai=totalnilai+nilai
    rerata=totalnilai/banyakMK
    print('Rata-Rata : ',rerata)
```

Setelah perintah diatas dijalankan (*run*) maka akan tampil bilangan seperti berikut ini :

```
Banyak Mahasiswa : 2
Mahasiswa ke- 1
Banyak Matakuliah yang diambil : 2
Input Nilai ke 1 :75
Input Nilai ke 2 :80
Rata-Rata :  77.5
Mahasiswa ke- 2
Banyak Matakuliah yang diambil : 2
Input Nilai ke 1 :60
Input Nilai ke 2 :90
Rata-Rata :  75.0
```

LATIHAN

Buatlah program untuk menghitung penjumlahan 2 buah matriks!

4.4 Kombinasi *For* Dengan Pemilihan

Penggunaan bentuk looping for dapat diperluas dengan bentuk lain yang sudah ada di python diantaranya adalah dengan mengkombinasikan bentuk perulangan for dengan bentuk pemilihan if dalam memecahkan kasus/permasalahan

Contoh penggunaan for dengan bentuk pemilihan tunggal:

Berikut ini contoh menampilkan bilangan tertentu dari daftar bilangan yang telah ada dimana program akan keluar dari perulangan jika bilangan itu adalah bilangan ganjil. Untuk soal seperti ini diperlukan bentuk for untuk memeriksa suatu bilangan dari daftar bilangan yang ada. Selanjutnya selama proses pemeriksaan diperlukan bentuk pemilihan if untuk menentukan apakah bilangan tersebut memenuhi kriteria yang telah ditentukan. Adapun perintahnya adalah sebagai berikut:

```
for i in [12, 16, 17, 24, 29]:
    if i % 2 == 1: #Jika bilangan ganjil
        break      #... Langsung keluar dari perulangan
    print(i)
print("selesai")
```

Setelah perintah diatas dijalankan (*run*) maka akan tampil bilangan seperti berikut ini :

```
12
16
```

Contoh penggunaan for dengan bentuk pemilihan if else:

Berikut ini contoh menampilkan jenis bilangan (genap atau ganjil) dari bilangan tertentu sampai bilangan tertentu dimana untuk nilai awal dan nilai akhir ditentukan ketika program dijalankan. Untuk soal seperti ini diperlukan bentuk **for** untuk menampilkan semua bilangan yang ada. Selanjutnya selama proses untuk menampilkan bilangan diperlukan bentuk pemilihan **if else** untuk menentukan apakah bilangan tersebut memenuhi kriteria yang telah ditentukan. Adapun perintahnya adalah sebagai berikut:

```
nilaiawal  = int(input('Input Nilai Awal : '))
nilaiakhir = int(input('Input Nilai Akhir: '))
for i in range (nilaiawal,nilaiakhir+1):
    if i%2==0:
        print(i,' adalah bilangan genap')
    else:
        print(i,' adalah bilangan ganjil')
```

Setelah perintah diatas dijalankan (*run*) maka akan tampil bilangan seperti berikut ini :

```
Input Nilai Awal : 5
Input Nilai Akhir: 10
5  adalah bilangan ganjil
6  adalah bilangan genap
7  adalah bilangan ganjil
8  adalah bilangan genap
9  adalah bilangan ganjil
10  adalah bilangan genap
```

Contoh penggunaan for dengan bentuk pemilihan if else bertingkat:

Berikut ini contoh menampilkan jenis bilangan (positif, negatif atau nol) untuk N buah bilangan yang diinput ketika program dijalankan dimana hanya diinput nilai akhir saja yang menyatakan banyaknya bilangan. Untuk soal seperti ini diperlukan bentuk **for** untuk

menerima input bilangan sebanyak N buah bilangan. Selanjutnya selama proses untuk menentukan jenis bilangan (positif, negatif atau nol) diperlukan bentuk pemilihan **if else bertingkat** untuk menentukan apakah bilangan tersebut memenuhi kriteria yang telah ditentukan. Digunakan bentuk **if else bertingkat** karena kondisi yang akan diperiksa lebih dari 2. Adapun perintahnya adalah sebagai berikut:

```
banyakbil = int(input('Input Banyak Bilangan : '))
for i in range(banyakbil+1):
    bil=int(input('Input Bilangan : '))
    if bil>0:
        print(bil,' adalah jenis bilangan positif')
    elif bil== 0:
        print(bil,' adalah bilangan nol')
    else:
        print(bil,' adalah jenis bilangan negatif')
```

Setelah perintah diatas dijalankan (*run*) maka akan tampil bilangan seperti berikut ini :

```
Input Banyak Bilangan : 10
Input Bilangan : 5
5  adalah jenis bilangan positif
Input Bilangan : 4
4  adalah jenis bilangan positif
Input Bilangan : 0
0  adalah bilangan nol
Input Bilangan : -2
-2  adalah jenis bilangan negatif
Input Bilangan : 4
4  adalah jenis bilangan positif
Input Bilangan : 0
0  adalah bilangan nol
Input Bilangan : -3
-3  adalah jenis bilangan negatif
Input Bilangan : 4
4  adalah jenis bilangan positif
Input Bilangan : 1
1  adalah jenis bilangan positif
Input Bilangan : -7
-7  adalah jenis bilangan negatif
Input Bilangan : 9
9  adalah jenis bilangan positif
```

Contoh penggunaan for dengan bentuk pemilihan if else:

Berikut ini contoh menampilkan bentuk bilangan tertentu dimana program akan menampilkan bilangan sesuai dengan baris dan kolomnya, dan hanya akan tampil jika

nilai baris dan kolomnya sama. Banyaknya baris ataupun kolom akan ditentukan pada saat program dijalankan. Untuk soal seperti ini diperlukan bentuk for bertingkat untuk menampilkan bilangan dalam bentuk baris dan kolom. Dikarenakan bilangan hanya akan tampil jika nilai baris dan kolomnya sama maka hanya perlu satu inputan saja untuk menyatakan baris/kolom yang akan digunakan untuk perulangan. Selanjutnya selama proses perulangan diperlukan bentuk pemilihan **if else** untuk menentukan apakah nilai baris dan kolomnya sama, jika nilai baris dan kolomnya tidak sama maka bilangan tidak ditampilkan (digantikan dengan spasi). Bilangan yang ditampilkan bentuknya akan menjadi diagonal. Adapun perintahnya adalah sebagai berikut:

```
bil = int(input('Input Bilangan : '))
for i in range(1,bil+1):
    for j in range(1,bil+1):
        if i==j:
            print(i,end='   ')
        else:
            print(end='   ')
    print()
```

Setelah perintah diatas dijalankan (*run*) maka akan tampil bilangan seperti berikut ini :

```
Input Bilangan : 10
1
   2
      3
         4
            5
               6
                  7
                     8
                        9
                           10
```

LATIHAN

1. Buatlah program untuk menghitung total pembelian N buah barang dengan harga dan jumlah tertentu

2. Buatlah program untuk menghitung banyaknya bilangan genap ataupun bilangan ganjil dari N buah bilangan yang diinput secara acak

3. Buatlah program untuk menampilkan bilangan kelipatan 3 diantara bilangan tertentu

4. Buatlah program untuk menampilkan bentuk bilangan seperti berikut ini:
   Bilangan : <input suatu bilangan (misal:5)>
   1
   12
   123
   1234
   12345

5. Buatlah program untuk menghitung dan menampilkan bentuk berikut ini:

   No. Nama Mhs N.Tugas N.Kuis N.UTS N.UAS NilaiAkhir

   1
   2
   ..
   N

Pertemuan 5 : Fungsi

Standar Kompetensi:
Memahami konsep dan penerapan fungsi

Sub Pokok Bahasan:
1. Struktur fungsi
2. Pembuatan dan pemanggilan fungsi
3. Fungsi dengan parameter
4. Variabel lokal dan global

5.1 Pengertian Fungsi

Untuk menyelesaikan masalah yang kompleks menggunakan perangkat lunak, kasus utama harus dipecah-pecah menjadi kasus yang lebih kecil. Kemudian kita berkonsentrasi untuk mencari pemecahan yang terbaik dari masing-masing bagian kecil ini. Masing-masing bagian diselesaikan dengan menggunakan algoritma sebaik mungkin. Bagian-bagian kecil ini pada akhirnya bisa dapat digabungkan untuk memberikan jawaban yang optimal terhadap masalah yang ada.

Fungsi dipakai untuk mengumpulkan beberapa perintah yang sering dipakai dalam sebuah program. Dengan memakai fungsi, program yang dibuat menjadi lebih terstruktur. Lebih mudah diikuti oleh orang lain yang membaca program dibuat. Paling penting adalah mempersingkat waktu yang diperlukan untuk mengembangkan suatu perangkat lunak. Karena perangkat lunak yang dibuat, bisa jadi memakai komponen-komponen yang sama.

Seperti layaknya sebuah bahasa pemrograman, Python juga memberikan fasilitas pembuatan fungsi yang sangat bagus. Konsep fungsi dalam Python sama dengan bahasa pemrograman C/C++. Python menganggap fungsi dan prosedur adalah sesuatu yang sama, dalam artian cara mendeklarasikan fungsi dan prosedur adalah sama. Hanya bedanya, kalau fungsi mengembalikan suatu nilai setelah proses sedangkan prosedur tidak.

Fungsi (*Function*) adalah suatu program terpisah dalam blok sendiri yang berfungsi sebagai sub-program (modul program) yang merupakan sebuah program kecil untuk memproses sebagian dari pekerjaan program utama.

Kategori Fungsi :

1. *Standard Library Function*

   Fungsi-fungsi yang telah disediakan oleh Interpreter Python dalam file-file atau library-nya.

2. *User Defined Function*

   Fungsi yang dibuat sendiri. Function ini memiliki nama tertentu yang unik dalam program, letaknya terpisah dari program utama, dan bisa dijadikan satu ke dalam suatu library

Mendeklarasikan dan Memanggil Fungsi

- Statemen def digunakan untuk mendeklarasikan fungsi.
- Sedangkan statemen return digunakan untuk mengembalikan suatu nilai kepada bagian program yang memanggil fungsi

5.2 Struktur Fungsi

Bentuk umum untuk mendeklarasikan fungsi adalah sebagai berikut :

```
def  namaFungsi(daftar-parameter) :
        pernyataan
        ...
            return [ekspresi]
```

- Sebuah fungsi diawali dengan statemen **def** kemudian diikuti oleh sebuah nama_fungsi nya. Pernyataan def dipakai untuk mendeklarasikan fungsi.
- Sebuah fungsi dapat memiliki daftar argumen (parameter) ataupun tidak.
- Tanda titik dua (:)
  menandakan awal pendefinisian tubuh dari fungsi yang terdiri dari statemen-statemen
- Statemen return
  menandakan akhir dari pemanggilan fungsi dan akan mengirimkan suatu nilai balik kepada program yang memanggil fungsi tersebut. Pernyataan return dipakai untuk mengembalikan suatu nilai kepada bagian program yang memanggil fungsi.
  Statemen return bersifat opsional, artinya jika sebuah fungsi tidak memiliki statemen return, maka sebuah fungsi tidak akan mengembalikan suatu nilai apapun

5.3 Pembuatan dan pemanggilan fungsi

Memanggil Fungsi

```
namaFungsi (daftar-parameter)
```

Dalam deklarasi fungsi, juga bisa menambahkan komentar-komentar yang memberi penjelasan mengenai fungsi yang dibuat. Secara umum memang bisa menambahkan komentar-komentar di sembarang tempat dalam program yang dibuat. Baris-baris komentar diawali dengan karakter pagar (#). Semua karakter yang mengikuti tanda ini sampai akhir baris dianggap sebagai komentar dan tidak akan mempengaruhi jalannya progam. Akan tetapi terdapat satu gaya pemberian komentar dalam Python yang disebut dengan ***docstring***. Biasanya dipakai untuk memberi penjelasan mengenai fungsi atau objek. *Docstring* diapit dengan tanda petik ganda, komentar jenis ini hanya boleh diberikan tepat satu baris dibawah deklarasi fungsi atau objek yang akan ditunjukkan pada pembahasan selanjutnya. *Docstring* sangat bermanfaat ketika kita ingin mendokumentasikan semua fungsi dan kelas yang telah kita buat. Karena ada beberapa perangkat lunak yang mampu membuat dokumentasi berdasarkan *docstring* yang ada dalam *source code*.

Contoh penggunaan fungsi tanpa parameter tanpa nilai kembalian :

Berikut ini adalah contoh fungsi untuk menampilkan kalimat “Hallo, Selamat Belajar Python”.

```
#deklarasi fungsi
def kalimat():
    "menampilkan kalimat Hallo, Selamat Belajar Python"
    print('Hallo, Selamat Belajar Python')

#Program Utama
#memanggil fungsi
kalimat()
```

Gambar 5.1 Contoh Fungsi Tanpa Parameter

Perintah diatas dapat dimaknai sebagai berikut, Pernyataan **def** mendefinisikan sebuah fungsi dengan nama **kalimat**. Tidak ada paramater yang akan dilewatkan ke dalam fungsi sehingga di dalam tanda kurung tidak ada yang perlu dituliskan Baris deklarasi fungsi ini diakhiri dengan titik dua (:). Tanda ini memberitahukan pada interpreter Python bahwa baris ini masih berlanjut pada baris-baris berikutnya. Dalam deklarasi diatas terdapat penggunaan komentar yang ditandai dengan tanda pagar (#) yaitu tulisan **#deklarasi fungsi** dan tulisan **#Program Utama** serta tulisan **#memanggil fungsi** dengan maksud untuk memberi keterangan ataupun memperjelas maksud dari kode-kode yang digunakan. Pada perintah diatas juga terdapat **docstring** yaitu tulisan "menampilkan kalimat Hallo, Selamat Belajar Python" yang diapit dengan tanda petik ganda. Digunakan untuk memberi penjelasan mengenai fungsi dengan nama kalimat diatas. Fungsi diatas tidak memiliki nilai kembalian dengan demikian tidak perlu menggunakan pernyataan **return**

Setelah perintah di atas dijalankan (*run*) maka akan tampil seperti berikut ini :

```
Hallo, Selamat Belajar Python
```

Contoh penggunaan fungsi tanpa parameter dengan nilai kembalian :

Berikut ini adalah contoh fungsi untuk menghitung penjumlahan 1 dengan 2!

```
#fungsi hitung
def hitung():
    return 1+2

#panggil fungsi coba
tampil=hitung()
print('Hasil Perhitungan : ',tampil)
```

Gambar 5.2 Contoh Fungsi Penjumlahan 2 Buah Bilangan

Pada perintah diatas setelah proses penjumlahan maka hasilnya akan dikembalikan ke fungsi yang memanggil dengan menggunakan pernyataan **return,** sehubungan ada nilai yang dikembalikan maka diperlukan variabel penampung yang dalam hal ini menggunakan variabel **tampil** baru selanjutnya hasilnya ditampilkan.

Setelah perintah diatas dijalankan (*run*) maka akan tampil seperti berikut ini :

```
Hasil Perhitungan :  3
```

5.4 Fungsi dengan parameter

Contoh penggunaan fungsi dengan parameter tanpa nilai kembalian :

Berikut ini adalah contoh fungsi untuk menginput nama dan kota kelahiran!

```
#deklarasi fungsi
def datadiri(nama,kota):
    "menampilkan nama dan kota"
    print('Nama Anda : ', nama)
    print('Dari Kota : ', kota)

#Program Utama
nama=input('Masukkan nama Anda : ')
kota=input('Masukkan kota asal : ')

#memanggil fungsi
datadiri(nama,kota)
```

Gambar 5.3 Contoh Fungsi Dengan Parameter

Pada perintah diatas paramater-parameter yang akan dilewatkan ke dalam fungsi didaftarkan dalam tanda kurung yaitu parameter nama dan kota. Masing-masing paramater dipisahkan oleh koma (,) . Setelah dilewatkan selanjutnya di dalam fungsi akan ditampilkan berdasarkan nilai yang dikirim

Setelah perintah diatas dijalankan (*run*) maka akan tampil seperti berikut ini :

```
Masukkan nama Anda : saya
Masukkan kota asal : palembang
Nama Anda :  saya
Dari Kota :  palembang
```

Contoh penggunaan fungsi dengan parameter dengan nilai kembalian :

Berikut ini adalah contoh fungsi perkalian dua bilangan bulat!

```
#deklarasi fungsi
def perkalian(bil1, bil2):
    "menghitung perkalian 2 bilangan bulat"
    hasil = bil1 * bil2
    return hasil

#Program Utama
bil1=int(input('Input bil 1 : '))
bil2=int(input('Input bil 2 : '))
#memanggil fungsi
print('Hasil Perkalian ',bil1,' dengan ',bil2,' adalah ', perkalian(bil1,bil2))
```

Gambar 5.4 Contoh-1 Fungsi Perkalian 2 Buah Bilangan

Pernyataan **def** mendefinisikan sebuah fungsi dengan nama perkalian. Paramater-parameter yang akan dilewatkan ke dalam fungsi didaftarkan dalam tanda kurung yaitu bil1 dan bil2. Masing-masing paramater dipisahkan oleh koma (,). Baris deklarasi fungsi ini diakhiri dengan titik dua (:). Tanda ini memberitahukan pada interpreter Python bahwa baris ini masih berlanjut pada baris-baris berikutnya.

Perhatikan dua baris pernyataan terindentasi yang mengikutinya. Dalam Python semua pernyataan yang diindentasi dalam satu tingkatan indentasi adalah pernyataan-pernyataan yang satu derajat. Artinya semua pernyataan tersebut akan dieksekusi sesuai dengan urutan penulisannya. Untuk memanggil fungsi yang telah dibuat adalah dengan cara menyebutkan nama fungsi yang bersangkutan beserta daftar parameter yang sebenarnya. Dalam deklarasi fungsi, Anda juga bisa menambahkan komentar-komentar yang memberi penjelasan mengenai fungsi yang dibuat. Secara umum kita memang bisa menambahkan komentar-komentar di sembarang tempat dalam program yang kita buat. Baris-baris komentar diawali dengan karakter pagar (#). Semua karakter yang mengikuti tanda ini sampai akhir baris dianggap sebagai komentar dan tidak akan mempengaruhi jalannya progam.

Setelah perintah diatas dijalankan (*run*) maka akan tampil seperti berikut ini :

```
Input bil 1 : 5
Input bil 2 : 7
Hasil Perkalian  5  dengan  7  adalah  35
```

Melewatkan Argumen dengan Kata Kunci

Kalau kita perhatikan kembali fungsi perkalian sebelumnya, proses penyalinan ke variabel lokal sesuai dengan urutan deklarasi fungsi yang kita panggil. Jika fungsi perkalian kita

panggil dengan memberi pernyataan perkalian(bil1,bil2), maka nilai 5 akan disalin ke variabel bil1 dan nilai 7 ke variabel bil2. Kadang-kadang ini agak menyulitkan jika kita membuat fungsi dengan jumlah variabel yang cukup banyak, sementara urutannya harus tepat. Solusinya adalah dengan menyebutkan kata-kunci (*keyword*) yang kita pakai pada saat mendefinisikan fungsi.

Kita ubah sedikit program perkalian kita agar pembahasan di bagian ini lebih jelas. Perhatikan program di bawah.

```
def perkalian(bil1, bil2):
    "menghitung perkalian 2 bilangan bulat"
    hasil = bil1 * bil2
    return hasil

#Program Utama
bil1=int(input('Input bil 1 : '))
bil2=int(input('Input bil 2 : '))
#memanggil fungsi
print('Hasil Perkalian ',bil1,' dengan ',bil2,' adalah ',
      perkalian(bil2=bil1,bil1=bil2))
```

Gambar 5.5 Contoh-2 Fungsi Perkalian 2 Buah Bilangan

Dengan menyebutkan kata kunci yang kita buat saat mendeklarasikan program kita dapat mengubah urutan penyalinan argumen. Akan tetapi Anda harus berhati-hati ketika menyebutkan kata-kunci, karena tidak boleh ada duplikasi. Panggil fungsi perkalian dengan pernyataan perkalian(7,bil1=5), maka Anda akan mendapatkan pesan kesalahan sbb.:

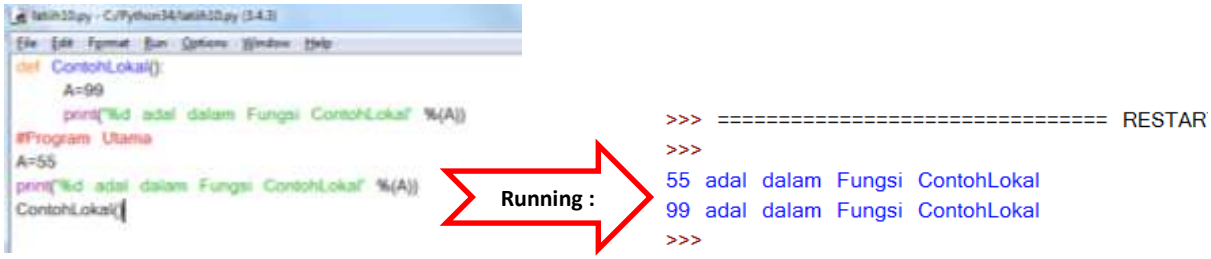
```
Traceback (most recent call last):
  File "F:/python/fungsi.py", line 25, in <module>
    perkalian(5,bil1=bil1))
TypeError: perkalian() got multiple values for argument 'bil1'
```

Hasil ini menunjukkan pada kita bahwa **bil1** sudah dipakai. Dengan melihat pada definisi fungsi yang telah dibuat, parameter pertama adalah **bil1** dan kedua adalah **bil2**. Jadi ketika kita panggil dengan menyebutkan parameter kedua sebagai **bil1** juga akan terjadi kesalahan.

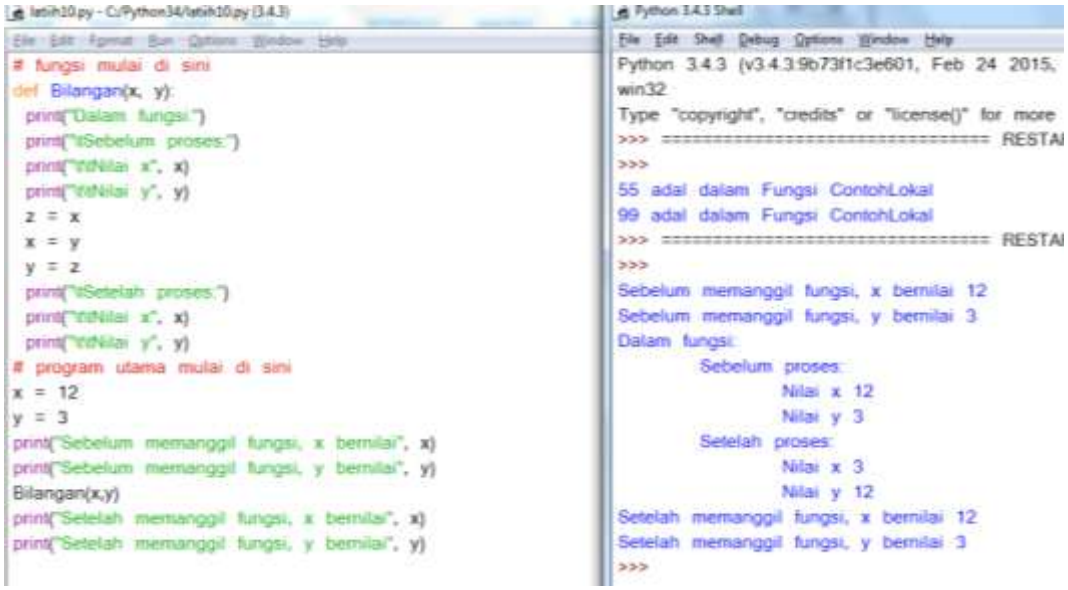
5.5 Variabel lokal dan global

Variabel local dan global

- Variabel disebut local ketika variabel tersebut didefinisikan didalam sebuah fungsi (def). Artinya, variabel tersebut hanya dapat di gunakan dalam cakupan fungsi tersebut saja.
- Variabel disebut global jika sebuah variabel didefinisikan diluar fungsi. Artinya, variabel tersebut dapat digunakan oleh fungsi lain atau pun program utamanya



Gambar 5.6 Contoh Fungsi Dengan Lokal Variabel



Gambar 5.7 Pemanggilan Fungsi Lokal Variabel dan Global Variabel

==========|||==========