

playing with Python

Python Fundamental For Programmer



Kumpul Artikel

Kata Pengantar

Syukur alhamdulillah, ane bisa menyelesaikan **penyusunan artikel** untuk pembelajaran Python dasar yang berjudul “*Playing With Python*” dengan edisi Bahasa Indonesia.

Ebook ini free. Boleh digunakan untuk siapapun untuk mendukung gerakan Python di Indonesia.

Ebook ini bukan karya ane, ane di sini cuma *ngumpulin* artikel dari berbagai sumber yang menurut ane relevan untuk di jadikan bahan pembelajaran Python Dasar. *Setiap bab udah ane kasih link sumbernya.*

Sekali lagi ane katakan :

Ebook ini free. Boleh digunakan untuk siapapun untuk mendukung gerakan Python di Indonesia.

Ebook ini bukan karya ane, ane di sini cuma *ngumpulin* artikel dari berbagai sumber yang menurut ane relevan untuk di jadikan bahan pembelajaran Python Dasar. *Setiap bab udah ane kasih link sumbernya.*

Niat ane cuma ngebantu biar yang pade mau belajar python ada fasilitasnya, maka ane coba kumpulin artikel (bukan nulis, cuma ngumpulin abis itu di editing biar rapih ke dalam bentuk ebook) untuk berbagi ke orang yang pengen belajar python.

Ente bisa langsung ke sumbernya di sini :

<http://www.tutorialspoint.com/python/>

<http://sakti.github.io/>

dan masih ada beberapa link lain yang udah ane sertakan (tuliskan) pada tiap bab di ebook ini.

Ane sadar ane juga newbie, ane pengen ngebantu sesama temen yang mau belajar Python. Ebook ini masih banyak kekurangan mohon koreksinya kembali jika terdapat program atau beberapa link yang error atau translate yang salah. Untuk kemajuan Python di Indonesia. Salam dari ane :

Pyeople :)

Jakarta, 1 September 2014

Daftar Isi

| | |
|---|----|
| Kata Pengantar..... | 2 |
| Install Python di Windows..... | 7 |
| Pendahuluan..... | 8 |
| Tools yang dibutuhkan..... | 8 |
| Instal Python 2.7..... | 9 |
| Instal ActivePython 2.7..... | 12 |
| Python Dasar..... | 15 |
| Pendahuluan..... | 16 |
| Menggunakan Python sebagai kalkulator..... | 16 |
| Operasi aritmatika..... | 17 |
| Halo Dunia !..... | 18 |
| Komentar..... | 18 |
| Konstanta Literal..... | 19 |
| Bilangan..... | 19 |
| String..... | 19 |
| Immutable..... | 19 |
| Format String..... | 19 |
| Variabel..... | 20 |
| Nama Pengenal..... | 20 |
| Tipe Data..... | 21 |
| Obyek..... | 21 |
| Penulisan Program Python..... | 22 |
| Baris Logis dan Fisik..... | 22 |
| Indentasi..... | 23 |
| Variabel di Python..... | 24 |
| Pendahuluan..... | 25 |
| Kata Kunci di Python & Syarat Pembuatan Variabel..... | 25 |
| Tipe Data dan Variabel Pada Python..... | 27 |
| Pendahuluan..... | 28 |
| Tipe data Number..... | 29 |
| Tipe data String..... | 29 |
| Menggunakan whitespace character..... | 30 |
| Contoh Penggunaan Variabel..... | 30 |
| Operator Pada Bahasa Pemrograman Python..... | 32 |
| Pendahuluan..... | 33 |
| Operator Aritmatika..... | 33 |
| Operator Pembandingan..... | 34 |
| Operator Logika..... | 34 |
| Operator dan Ekspresi..... | 35 |
| Pendahuluan..... | 36 |
| Operator..... | 36 |
| Operasi matematika dan pengisian variabel..... | 39 |
| Urutan Evaluasi..... | 39 |
| Mengubah Urutan Evaluasi..... | 40 |
| Sifat Asosiatif..... | 40 |

| | |
|---|----|
| Penggunaan If Pada Python..... | 41 |
| Pendahuluan..... | 42 |
| Menggunakan IF Tunggal..... | 42 |
| Menggunakan IF ELSE..... | 42 |
| Menggunakan ELIF..... | 42 |
| Alur Kontrol..... | 44 |
| Pendahuluan..... | 45 |
| Statemen If..... | 45 |
| Statemen While..... | 46 |
| Perulangan For (For Loop)..... | 47 |
| Statemen Break..... | 48 |
| Statemen Continue..... | 49 |
| Struktur Data Python..... | 50 |
| Pendahuluan..... | 51 |
| List..... | 51 |
| Pengenalan singkat obyek dan class..... | 51 |
| Tuple..... | 52 |
| Dictionary..... | 53 |
| Sequence..... | 54 |
| Set..... | 55 |
| Referensi..... | 55 |
| String..... | 56 |
| List di Python..... | 57 |
| Pendahuluan..... | 58 |
| List Python..... | 58 |
| Mengakses Nilai di dalam List..... | 58 |
| Memperbarui List..... | 59 |
| Menghapus Elemen List..... | 59 |
| Operasi Dasar List..... | 60 |
| Pengeindeksan, pengirisan dan matrixes..... | 60 |
| Fungsi Built-in List..... | 61 |
| Fungsi cmp..... | 61 |
| Fungsi len..... | 61 |
| Fungsi max..... | 62 |
| Fungsi min..... | 62 |
| Fungsi list..... | 63 |
| Metode Built-in List..... | 64 |
| Metode append..... | 64 |
| Metode count..... | 64 |
| Metode extend..... | 65 |
| Metode index..... | 65 |
| Metode insert..... | 66 |
| Metode pop..... | 66 |
| Metode remove..... | 67 |
| Metode reverse..... | 67 |
| Metode sort..... | 68 |
| Tuple..... | 69 |

| | |
|---|-----|
| Pendahuluan..... | 70 |
| Mengakses Nilai di dalam Tuple..... | 70 |
| Mengupdate Tuple..... | 71 |
| Menghapus Tuple..... | 71 |
| Operasi Dasar Tuple..... | 72 |
| Pengindeksan dan Pemotongan..... | 72 |
| Fungsi Built-In Tuple..... | 73 |
| Fungsi cmp..... | 73 |
| Fungsi len..... | 74 |
| Fungsi max..... | 74 |
| Fungsi min..... | 75 |
| Fungsi tuple..... | 75 |
| Dictionary..... | 77 |
| Pendahuluan..... | 78 |
| Mengakses Nilai di Dictionary..... | 78 |
| Mengupdate Dictionary..... | 79 |
| Menghapus Elemen-Elemen Dictionary..... | 79 |
| Key Dictionary..... | 80 |
| Fungsi Built-In Dictionary..... | 81 |
| Fungsi cmp..... | 81 |
| Fungsi len..... | 82 |
| Fungsi str..... | 82 |
| Fungsi type..... | 83 |
| Method Built-In Dictionary..... | 84 |
| Method clear..... | 84 |
| Method copy..... | 85 |
| Method fromkeys..... | 85 |
| Method get..... | 86 |
| Method has_key..... | 86 |
| Method value..... | 87 |
| Library Standar..... | 88 |
| Module getpass..... | 89 |
| Modul random..... | 89 |
| Modul datetime..... | 90 |
| Modul math..... | 91 |
| Modul sys..... | 92 |
| PYMOTW (Python Module of The Week)..... | 93 |
| Fungsi (Function)..... | 94 |
| Pendahuluan..... | 95 |
| Parameter Fungsi..... | 95 |
| Variabel Lokal..... | 96 |
| Penggunaan Statemen Global..... | 97 |
| Nilai Argumen Default..... | 98 |
| Keyword Argumen..... | 98 |
| Parameter VarArgs..... | 99 |
| Statemen Return..... | 99 |
| Doc String..... | 100 |

| | |
|---|-----|
| Modul..... | 101 |
| Pendahuluan..... | 102 |
| Byte-compiled (file .pyc)..... | 102 |
| Statement from ... import..... | 102 |
| Nama Modul..... | 103 |
| Fungsi dir..... | 103 |
| Package..... | 103 |
| Input dan Output..... | 105 |
| Pendahuluan..... | 106 |
| Input dari Pengguna..... | 106 |
| File..... | 106 |
| Pickle..... | 107 |
| Eksepsi..... | 108 |
| Pendahuluan..... | 109 |
| Syntax Error..... | 109 |
| Exception..... | 109 |
| Penanganan Exception..... | 110 |
| Mengeluarkan Exception..... | 110 |
| Try ... Finally..... | 111 |
| Statemen with..... | 111 |
| Object-Oriented Programming..... | 113 |
| Pendahuluan..... | 114 |
| this : self..... | 114 |
| Class..... | 114 |
| Method Obyek..... | 115 |
| Method init..... | 115 |
| Variabel Class dan Variabel Obyek (Instance)..... | 116 |
| Inheritance..... | 117 |

Install Python di Windows

Artikel ini bersumber pada :

<http://kernelnotfound.wordpress.com/2012/02/22/install-python-di-windows/>

Pendahuluan

Python merupakan bahasa pemrograman yang sedang lagi trend dikalangan linux , terlebih dalam program-program yang berbau hacking, saya juga baru mengenal pemrograman python ini beberapa minggu yang lalu karena mencoba sebuah tools yang diberikan teman dalam bahasa pemrograman python . Karena saya baru dalam dunia program dan memprogram maka , saya mencoba mencari-cari beberapa literatur tentang bahasa pemrograman ini . Berikut ini adalah beberapa langkah mudah dalam instalasi python pada system windows (xp , vista, seven).

Tools yang dibutuhkan

Distribusi Python 2.7 (dalam bentuk installer)

unduh : <http://www.python.org/download/>

Download Python

Download Python

The current production versions are [Python 3.4.1](#) and [Python 2.7.8](#).

Start with one of these versions for learning Python or if you want the most stability; they're both considered stable production releases.

If you don't know which version to use, try Python 3.4. Some existing third-party software is not yet compatible with Python 3; if you need to use such software, you can download Python 2.7.x instead.

For the MD5 checksums and OpenPGP signatures, look at the [detailed Python 3.4.1](#) page:

- [Python 3.4.1 Windows x86 MSI Installer](#) (Windows binary -- does not include source)
- [Python 3.4.1 Windows X86-64 MSI Installer](#) (Windows AMD64 / Intel 64 / X86-64 binary [1] -- does not include source)
- [Python 3.4.1 Mac OS X 64-bit/32-bit x86-64/i386 Installer](#) (for Mac OS X 10.6 and later [2])
- [Python 3.4.1 Mac OS X 32-bit i386/PPC Installer](#) (for Mac OS X 10.5 and later [2])
- [Python 3.4.1 compressed source tarball](#) (for Linux, Unix or Mac OS X)
- [Python 3.4.1 xzipped source tarball](#) (for Linux, Unix or Mac OS X, better compression)

ActivePython 2.7 (dalam bentuk installer)

unduh : <http://downloads.activestate.com/ActivePython/releases/2.7.1.4/ActivePython-2.7.1.4-win32-x86.msi>

ActivePython is the leading commercial-grade distribution of the open source Python scripting language. Download ActivePython Community Edition free binaries for your development projects and internal deployments.

By downloading ActivePython Community Edition, you comply with the terms of use of the [ActiveState Community License](#) . Please refer to our [documentation](#) for install/uninstall instructions.

Looking for more Python packages and modules? Download ActivePython Community Edition then launch [PyPM](#).



**Download ActivePython 2.7.8
for Windows (x86)**

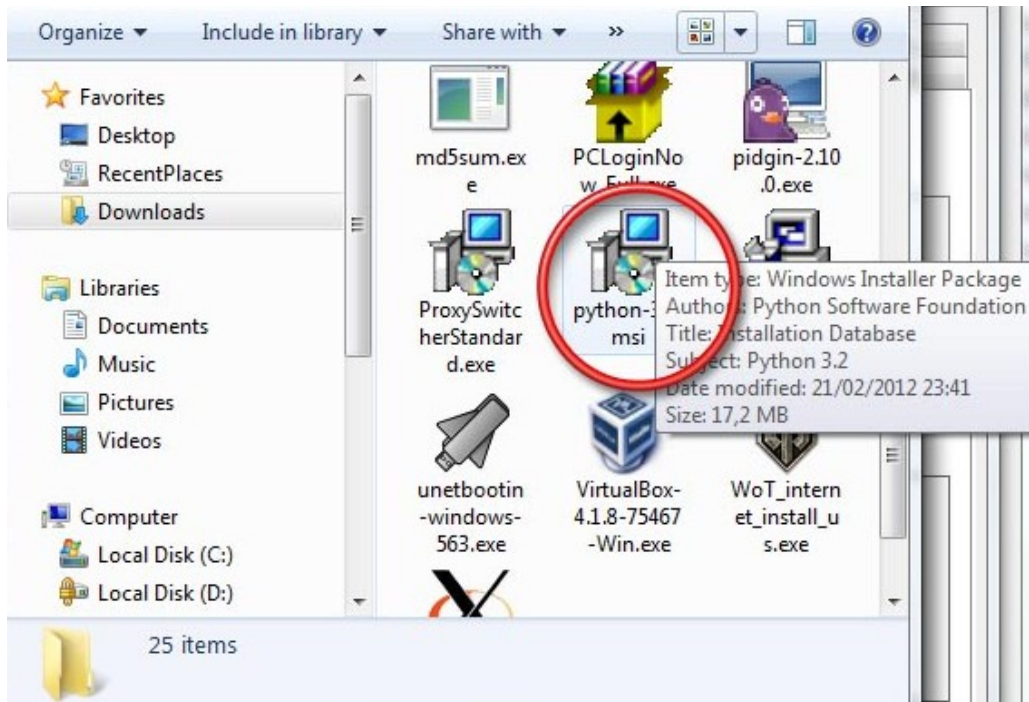


**Download ActivePython 2.7.8
for Windows (64-bit, x64)**

Instal Python 2.7

Pastikan hasil download dua buah file diatas tidak corrupt , soalnya kalau corrupt bisa terjadi kegagalan dalam instalasi.

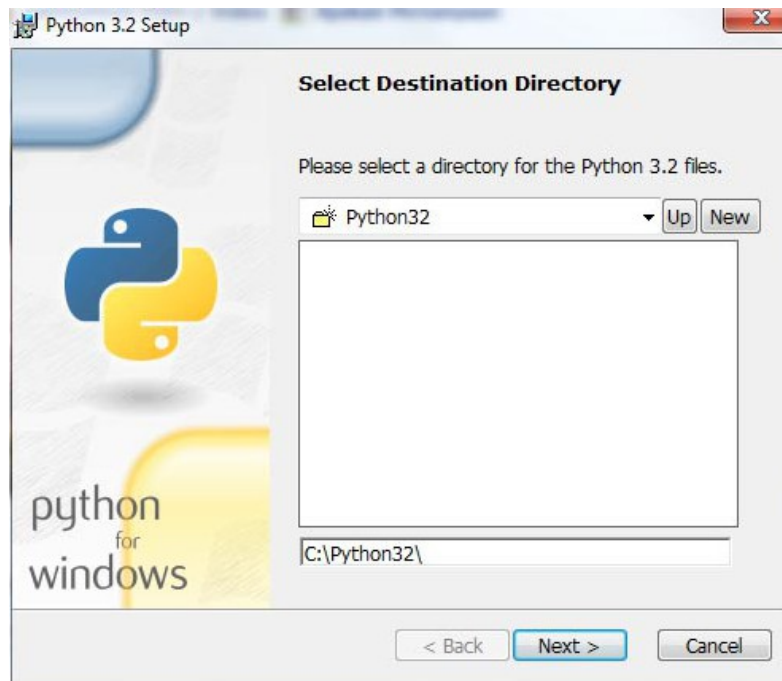
Double klik **instaler python-3.2.msi**



Setelah anda double klik maka akan muncul jendela instalasi seperti di bawah ini , pilih install for all users agar bisa digunakan oleh semua user.



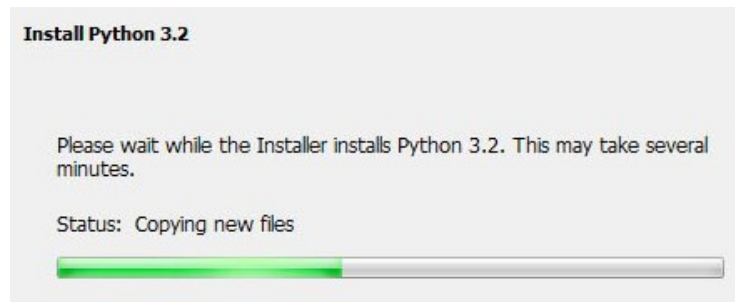
Pada bagian ini kita dapat menentukan di mana letak file hasil instaler nantinya , opsi saya biarkan default. pilih next.



pada kotak instaler dibawah ini kita pilih semua agar semua fitur dari pemograman python akan tersedia nantinya. pilih next jika selesai.



Proses instalasi sedang berlangsung tunggu sampai selesai.



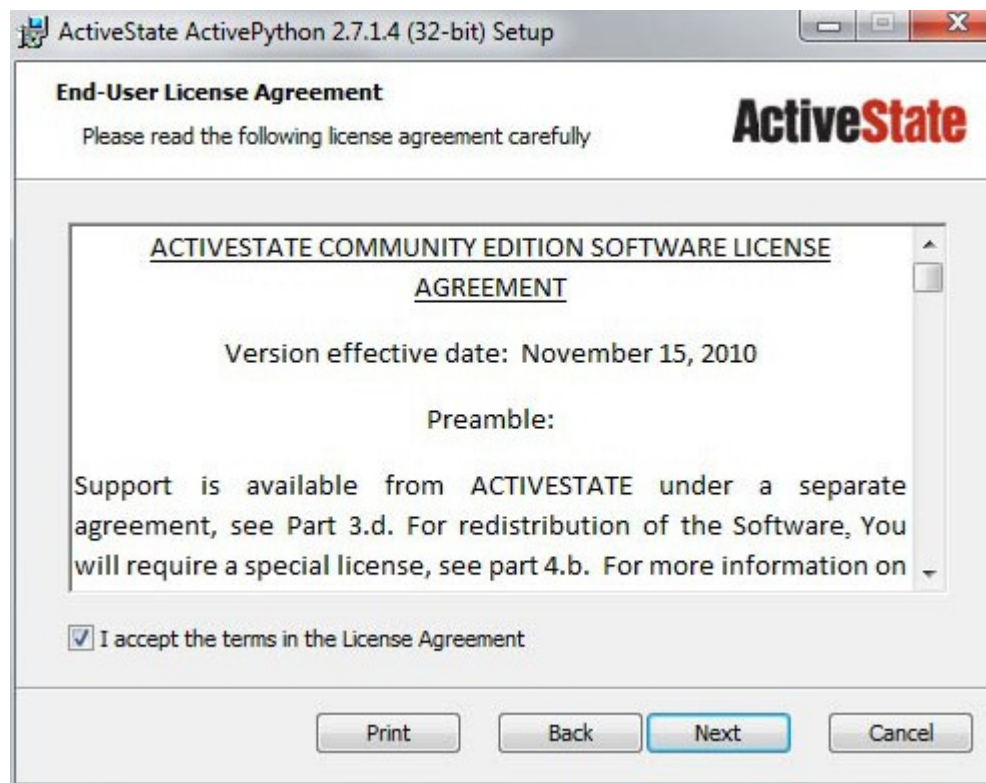
Klik finish.



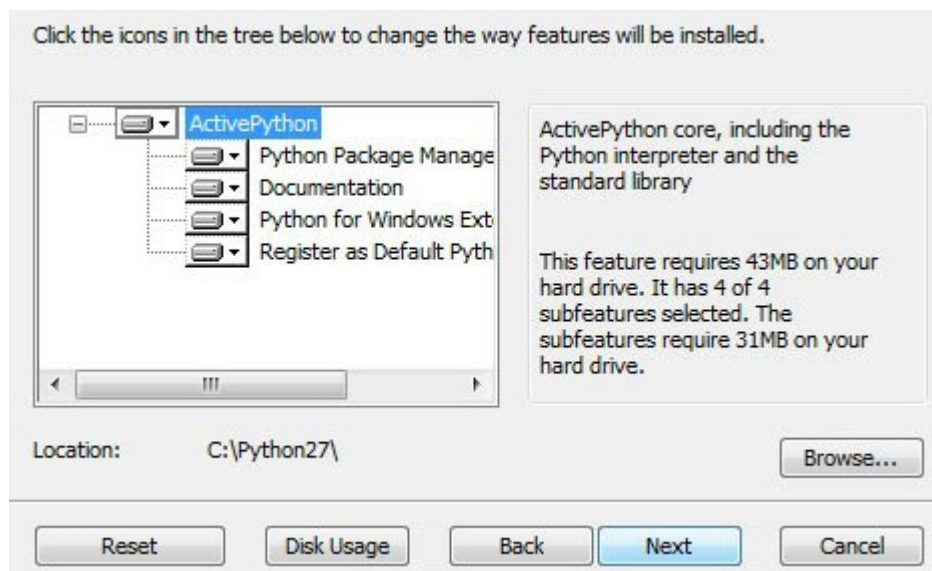
Pada tahap ini kita telah selesai melakukan instalasi Python pada windows , selanjutnya kita harus menginstal ActivePython 2.7.

Instal ActivePython 2.7

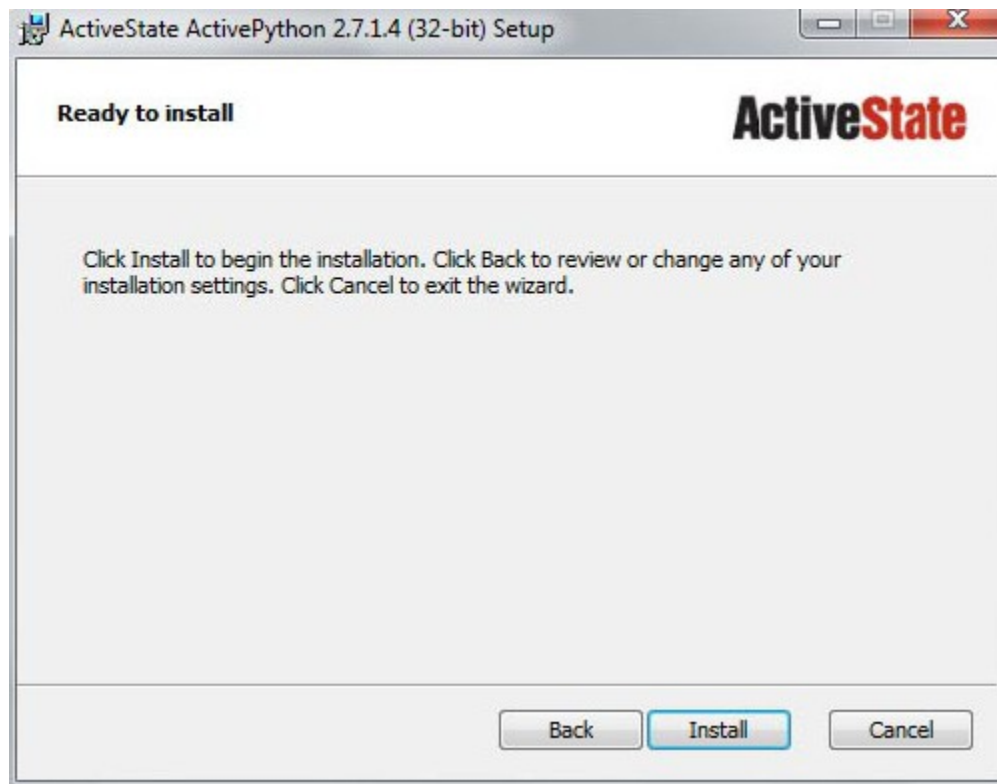
cari file instaler ActivePython 2.7, Double klik maka akan muncul jendela instalasi seperti dibawah ini.



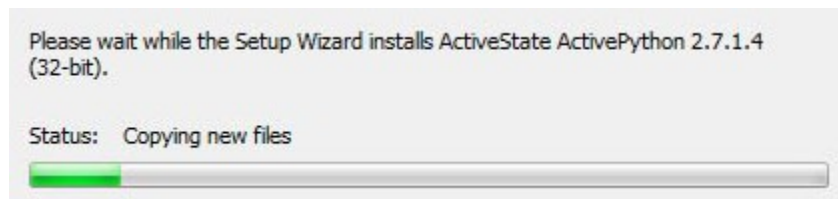
Pada bagian ini kita biarkan default , lalu pilih next.



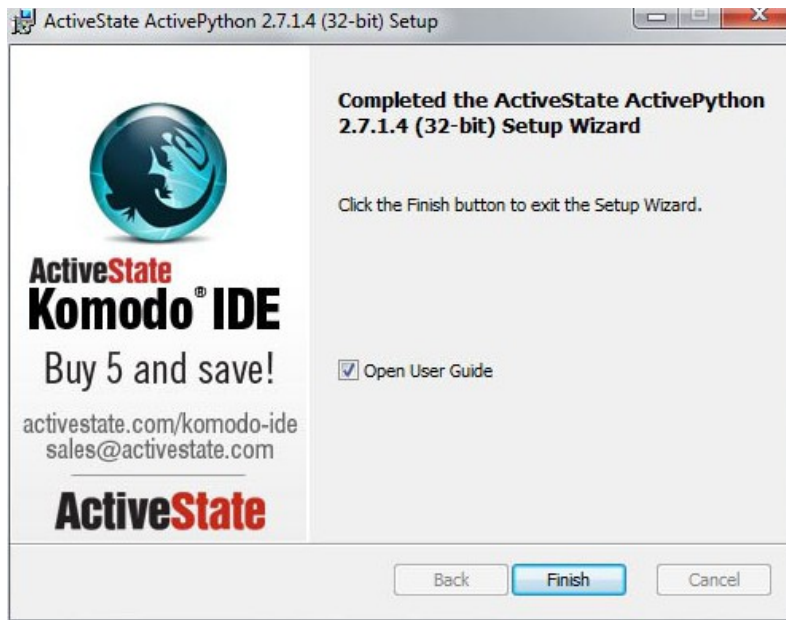
Pada kotak dialog dibawah ini pilih instal untuk melakukan instalasi.



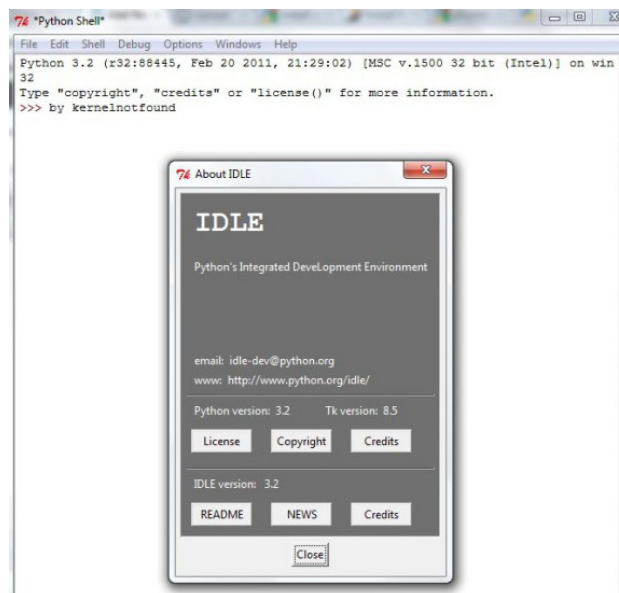
Proses instalasi berjalan , tunggu sampai selesai.



Klik finish , maka python anda telah siap untuk digunakan , selamat berkreasi dengan pemograman Python.



Hasil program Python yang sudah terinstal pada system.



Kesimpulan

sebenarnya saat kita menginstall ActivePython, itu sudah termasuk menginstal tools Pythonnya.

Python Dasar

sumber : <http://sakti.github.io/python101/dasar.html>

Pendahuluan

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python mendukung multi paradigma pemrograman, utamanya; namun tidak dibatasi; pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi.

Menggunakan Python sebagai kalkulator

Program Python dapat dijalankan dengan beberapa mode. Jika kita mengeksekusi Python interpreter tanpa argument script Python yang telah kita buat, Python interpreter akan masuk ke mode interaktif (REPL, read-eval-print loop).

```
$ python
Python 2.7.3 (default, Aug 1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
```

Kita dapat memanfaatkan Python dengan mode interaktif sebagai kalkulator.

```
$ python
Python 2.7.3 (default, Aug 1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 1
2
>>> 40 * 2
80
>>> 40 / 5
8
>>> 9 - 10
-1
>>> 2 + 3 + 4 + 5
14
>>> 2 ** 32
4294967296
```


Operasi aritmatika

Berikut table operasi aritmatika yang ada di Python:

| Operasi | Keterangan |
|---------|---|
| + | Menambahkan dua obyek |
| - | Mengurangi obyek dengan obyek yang lain |
| * | Perkalian |
| ** | Pangkat |
| / | Pembagian |
| // | Pembagian bulat ke bawah |
| % | Sisa hasil bagi (modulus) |

Contoh

Pembagian, perhatikan perbedaan antara bilangan bulat dan pecahan / desimal.

```
>>> 10 / 3
3

>>> 10.0 / 3
3.3333333333333335

>>> 10 / 3.0
3.3333333333333335

>>> 10.0 / 3.0
3.3333333333333335

>>> 10.0 // 3.0
3.0

>>> 10.0 // 3
3.0
```

Sisa hasil bagi.

```
>>> 10 % 3
1
>>> 2 % 3
2
>>> -5 % 4
3
>>> -5 % -4
-1
```

mode eksekusi lain

-m Mengeksekusi module, contoh: `python -m SimpleHTTPServer` untuk membuat webserver statis

-c Mengeksekusi command dari parameter yang diterima, contoh: `python -c 'import this'` untuk menampilkan **Zen of Python**.

Halo Dunia !

Program pertama yaitu program yang jika dijalankan akan mengeluarkan hasil teks berupa **Halo Dunia!**.

```
# lat1.py
print 'Halo Dunia!'
```

Anda bisa membuat file `lat1.py` menggunakan teks editor pilihan anda. Untuk catatan, anda sebaiknya menset teks editor anda agar untuk indentasi menggunakan spasi / space sebanyak 4. Standar PEP (Python Enhancement Proposal) menyarankan agar indentasi selalu konsisten.

Setelah file `lat1.py` disimpan, anda dapat menjalankannya melalui terminal.

```
$ python lat1.py
Halo Dunia!
```

Komentar

Komentar adalah teks apapun yang diawali dengan tanda `#`, digunakan untuk memberikan catatan kepada pembaca kode. Anda dapat melihat kembali **lat1.py** untuk memberikan keterangan nama file kita dapat memberikan komentar.

```
# lat2.py
# lat2.py adalah nama file ini
# program ini akan menampilkan 'Halo Indonesia!'
# kemudian akan menampilkan 'Halo Jakarta!'

print 'Halo Indonesia!'
print 'Halo Jakarta!'

# print 'Teks ini tidak akan dicetak.'
```

outputnya :

```
Halo Indonesia!
Halo Jakarta!
```

Konstanta Literal

Salah satu contoh konstanta literal yaitu bilangan seperti **5**, **1.23**, atau string seperti **'hari senin'** atau **"hari jum'at"**. Hal ini disebut literal atau harfiah karena anda bisa menggunakan nilai ini secara langsung. Bilangan 2 selalu merepresentasikan dirinya sendiri, dinamakan konstanta karena nilainya tidak dapat diubah.

Dalam latihan 2, **'Halo Indonesia!'** dan **'Halo Jakarta!'** merupakan string literal.

Bilangan

Di Python bilangan dibagi menjadi dua tipe utama - `integer` (bulat) dan `float` (pecahan). Salah satu contoh dari integer yaitu 2 yang merupakan bilangan bulat. Contoh untuk `float` yaitu 3.23 dan 52.3e-4. Notasi e mengindikasikan pangkat 10. Untuk kasus ini 52.3e-4 berarti $52.3 * 10^{-4}$.

String

String adalah rangkaian karakter. Anda bisa menuliskan string literal dengan beberapa cara :

Single Quote

Contoh: **'Halo Bandung!'**, **'Hari Jum'\at'**.

Double Quote

Contoh: **"Halo Surabaya!"**, **"Hari Jum'at"**. Perhatikan tanda quote ' harus di escape pada single quote. Selain itu tidak ada perbedaan antara single quote dan double quote, anda bebas untuk memilih.

Triple Quote

Python mendukung multi-line string atau string dengan baris lebih dari satu. Anda dapat dengan bebas menuliskan single quote ' dan double quote " dalam string literal yang diapit dengan triple quote.

Contoh:

Immutable

String bersifat immutable yang berarti setelah string dibuat, string tersebut tidak bisa diubah.

Format String

Terkadang kita ingin membuat string dari informasi lain, untuk hal ini kita dapat menggunakan format string.

```
# lat3.py
# format string menggunakan operator '%' dan method format
```

```
print '%s pergi ke %s' % ('ibu', 'pasar')
print '{0} pergi ke {1}'.format('ibu', 'pasar')

print 'jumlah total: %10.3f' % 10.3333
print 'jumlah total: {0:10.3f}'.format(10.3333)
```

outputnya :

```
ibu pergi ke pasar
ibu pergi ke pasar
jumlah total:      10.333
jumlah total:      10.333
```

Operator % jika digunakan untuk string bukan berarti modulus melainkan string format.

Variabel

Hanya menggunakan konstanta literal saja cukup membosankan, kita membutuhkan cara untuk menyimpan dan memanipulasi informasi. Untuk hal ini kita bisa menggunakan variabel. Seperti namanya, variabel dapat diisi dengan bermacam-macam nilai, anda dapat menyimpan apapun menggunakan variabel. Variabel adalah sebagian dari memori komputer yang digunakan untuk menyimpan informasi. Berbeda dengan konstanta literal, anda membutuhkan cara untuk mengakses variabel ini, oleh karena itu kita memberi nama kepada variabel.

Nama Pengenal

Berikut aturan penamaan variabel dalam python.

- Karakter pertama harus berupa karakter alfabet (huruf besar atau huruf kecil ASCII, atau unicode) atau underscore _.
- Karakter selanjutnya dapat berupa alfabet (huruf besar atau huruf kecil ASCII, atau unicode), underscore _ atau digit (0-9).
- Nama variabel bersifat case-sensitif. Sebagai contoh, namaMhs dan namamhs adalah variabel yang berbeda.

```
# lat4.py
# menggunakan variabel
a = 10
b = 20
c = 30
total = a + b + c
nama = 'ibu'
tempat = 'kantor'
print 'jumlah total = %s' % total
print '%s pergi ke %s' % (nama, tempat)
```

outputnya :

```
jumlah total = 60
ibu pergi ke kantor
```

Tipe Data

Variabel dapat menyimpan nilai dengan berbagi tipe disebut dengan tipe data. **Bilangan** dan **string** adalah tipe dasar, yang sudah dibahas sebelumnya. Pada latihan berikutnya akan dibahas tipe data yang lain.

Perlunya Anda menggunakan `type` untuk menentukan tipe data variabel / obyek yang ada.

```
>>> type(1)
<type 'int'>

>>> type(3.2)
<type 'float'>

>>> type(2 ** 1000)
<type 'long'>

>>> type('abc')
<type 'str'>

>>> type('a')
<type 'str'>
```

Obyek

Semua yang ada dalam Python adalah obyek / object. Obyek memiliki field yang memiliki nilai tertentu dan method untuk operasi tertentu.

Untuk melihat field dan method yang ada dalam suatu obyek kita dapat gunakan fungsi builtin `dir`.

```
>>> dir('abc')

['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattr__', '__getitem__', '__getnewargs__',
'__getslice__', '__gt__', '__hash__', '__init__', '__le__', '__len__', '__lt__',
'__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__', '_formatter_field_name_split', '_formatter_parser',
'capitalize', 'center', 'count', 'decode', 'encode', 'endswith', 'expandtabs',
'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace',
'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace',
'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines',
'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

>>> 'abc'.upper
<built-in method upper of str object at 0x7fe601a1f800>

>>> 'abc'.upper()
'ABC'
```

Selain dapat melihat isi obyek, anda dapat mengakses dokumentasi object menggunakan `help`.

```
>>> help(str)
```

```

Help on class str in module __builtin__:

class str(basestring)
|   str(object) -> string
|
|   Return a nice string representation of the object.
|   If the argument is a string, the return value is the same object.
|
|   Method resolution order:
|       str
|       basestring
|       object
|
...

>>> help(str.upper)
Help on method_descriptor:

upper(...)
    S.upper() -> string

    Return a copy of the string S converted to uppercase.

```

Penulisan Program Python

Berikut cara menulis program Python.

1. Buka teks editor pilihan anda, seperti: **vim**, **emacs**, **gedit**, **notepad++**, **sublimetext2** atau **Aptana Studio 3**.
2. Ketikkan kode program seperti contoh yang ada (hindari copy-paste).
3. Simpan sesuai nama yang ada.
4. Untuk menjalankan program gunakan terminal / command line, ketik **python namaprogram.py**.

Untuk pengguna **sublimetext2** anda dapat menjalankan program python menggunakan shortcut **Ctrl+b**.

Baris Logis dan Fisik

Baris fisik adalah apa yang anda lihat ketika anda melihat program. Baris logis adalah apa yang Python lihat sebagai statemen tunggal. Python mengasumsikan bahwa setiap baris fisik sesuai dengan baris logis.

Sebagai contoh baris logis seperti statemen print 'Halo Dunia!', jika anda menulis sebagai satu baris maka baris logis sesuai dengan baris fisik.

Catatan

Anda dapat menulis print '**Halo Dunia!**' menjadi dua baris, contoh :

```

print \
'Halo Dunia!'

```

Anda juga dapat membuat beberapa baris logis menjadi satu baris fisik, contoh:

```
nama = 'budi'; print nama
```

Secara implisit, Python menyarankan menggunakan satu statemen tiap baris untuk menjadikan kode menjadi lebih mudah dibaca.

Indentasi

Karakter spasi penting untuk bahasa pemrograman Python. Lebih tepatnya **spasi diawal baris** atau indentasi. Spasi diawal (baik berupa spasi atau tab) baris logis digunakan untuk menentukan level indentasi, yang akan mempengaruhi pengelompokan statemen.

Statemen yang mempunyai level indentasi sama masuk dalam satu kelompok yang disebut **blok / block**. Hal ini akan digunakan pada bab berikutnya.

```
# lat5.py
# error indentasi

a = 10
b = 20
c = 30

    total = a + b + c

nama = 'ibu'
tempat = 'kantor'

print 'jumlah total = %s' % total
print '%s pergi ke %s' % (nama, tempat)
```

ouputnya :

```
File "D:\codes\python\lat5.py", line 8
    total = a + b + c
    ^
IndentationError: unexpected indent
```

Variabel di Python

sumber :

<http://klinikpython.wordpress.com/2012/01/03/dasar-python-semua-harus-mempunyai-nama/>

Pendahuluan

Yes! Nama sangatlah penting artinya dalam dunia pemrograman, karena nama adalah sebuah **Identitas**. Saatnya kita mengintip tentang Identitas dan Kata kunci di Python. Jangan lupa secangkir kopi plus alunan MP3 layak untuk mendampingi Anda.

Kita telah mempelajari sebelumnya bahwa setiap program akan selalu memanggil variabel dengan menggunakan nama dari variabel tersebut. Setiap bahasa pemrograman memiliki aturan sendiri-sendiri untuk penamaan suatu variabel. Nama ini sering kita sebut sebagai Identitas (identifier). Python tidak mengizinkan kita untuk menggunakan beberapa kata, yang termasuk Kata Kunci (keyword), sebagai nama variabel. Berikut daftar kata kunci yang terdapat dalam Python.

Kata Kunci di Python & Syarat Pembuatan Variabel

| | | | | |
|--------|----------|---------|----------|--------|
| False | class | finally | is | return |
| None | continue | for | lambda | try |
| True | def | from | nonlocal | while |
| and | del | global | not | with |
| as | elif | if | or | yield |
| assert | else | import | pass | |
| break | except | in | raise | |

Interpreter Python juga telah mendefinisikan beberapa aturan untuk penamaan identitas seperti berikut ini :

1. Nama variabel tidak boleh ada spasi atau simbol-simbol seperti ? ! @ # + - % ^ & * () [] { } . , ; : " ' / dan \. Tanda garis bawah (_) dapat digunakan untuk mengganti spasi. Contoh: **klinik_python**.
2. Nama variabel harus unik. Misalkan kita ingin menyimpan empat bilangan yang berbeda-beda, maka kita harus membuat empat nama variabel yang unik alias tidak boleh sama. Identitas bersifat case-sensitive, yang artinya penulisan huruf besar atau huruf kecil dianggap berbeda. Contohnya: klinikPython itu berbeda dengan KlinikPython.
3. Nama variabel dapat terdiri dari angka dan karakter.
4. Nama variabel harus dimulai dengan huruf atau tanda garis-bawah, dan selanjutnya boleh diisi dengan huruf, angka (0 – 9), atau garis-bawah. Karakter pertama dari nama variabel tidak boleh berupa angka.

Secara umum, kita bisa mengikuti aturan penamaan variabel seperti berikut :

1. Nama variabel harus ringkas dan bermakna. Nama harus berhubungan dengan isi dari variabel tersebut. Contohnya, misalkan kita akan menyimpan data tentang umur siswa, maka nama variabel yang tepat adalah umur_siswa.
2. Nama variabel secara umum ditulis dalam huruf kecil.
3. Jika nama variabel terdiri dari dua kata, gabungkan dua kata tersebut dengan gunakan huruf besar di depan setiap kata. Atau pisahkan tiap-tiap kata dengan tanda garis-bawah.

Tipe Data dan Variabel Pada Python

sumber artikel :

<http://kamarpython.blogspot.com/2012/05/tipe-data-dan-variabel-pada-python.html>

Pendahuluan

Variabel adalah entitas yang memiliki nilai dan berbeda satu dengan yang lain. Variabel mengalokasikan memori untuk menyimpan nilai. Hal ini berarti ketika Anda membuat variabel, maka Anda memesan beberapa ruang di memori. Variabel bisa digunakan untuk menyimpan bilangan bulat, desimal atau juga karakter.

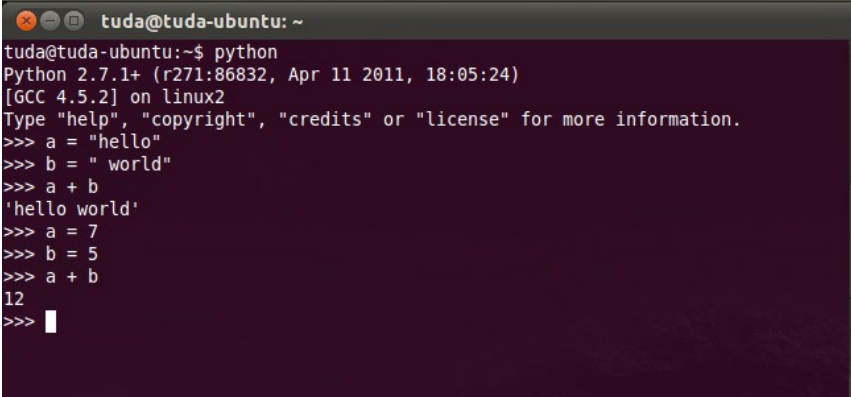
Pada banyak bahasa pemrograman, ada dua cara untuk membuat variabel. Cara yang pertama variabel langsung dengan nilai disebut dengan inisialisasi. Sedangkan cara kedua dengan memasukkan nilai pada variabel yang biasa disebut penempatan.

Tipe data di bahasa pemrograman python dibagi menjadi dua kelompok yaitu :

1. **immutable** = tipe data yang tidak bisa diubah (string dan bilangan)
2. **mutable** = tipe data yang bisa diubah (list dan dictionary)

Tidak seperti pemrograman lainnya, variabel pada Python tidak harus dideklarasikan secara eksplisit. Pendeklarasian variabel terjadi secara otomatis ketika kita memberikan sebuah nilai pada suatu variabel. Untuk pemberian nilai, bisa langsung dengan tanda "=". Misalnya :

```
nama  = 'Jack Port'
no    = 7
```



```
tuda@tuda-ubuntu: ~  
tuda@tuda-ubuntu:~$ python  
Python 2.7.1+ (r271:86832, Apr 11 2011, 18:05:24)  
[GCC 4.5.2] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> a = "hello"  
>>> b = " world"  
>>> a + b  
'hello world'  
>>> a = 7  
>>> b = 5  
>>> a + b  
12  
>>> 
```

Tipe data Number

1. **Plain integer.** Integer ini mempunyai range nilai antara -2^{32} sampai $2^{31} - 1$.
2. **Long integer.** Perhitungan di luar range nilai integer.
3. **Floating Point Real Number.** Bilangan real.
4. **Complex number.** Untuk bilangan real dan imajiner.



```
tuda@tuda-ubuntu: ~
>>> v_i = 10
>>> v_f = 24.5
>>> v_c = 3+4j
>>> print v_i
10
>>> print v_f
24.5
>>> print v_c
(3+4j)
>>>
```

... nilai pada sebuah variabel dengan tipe data yang Anda inginkan. Operator pengisian adalah tand...

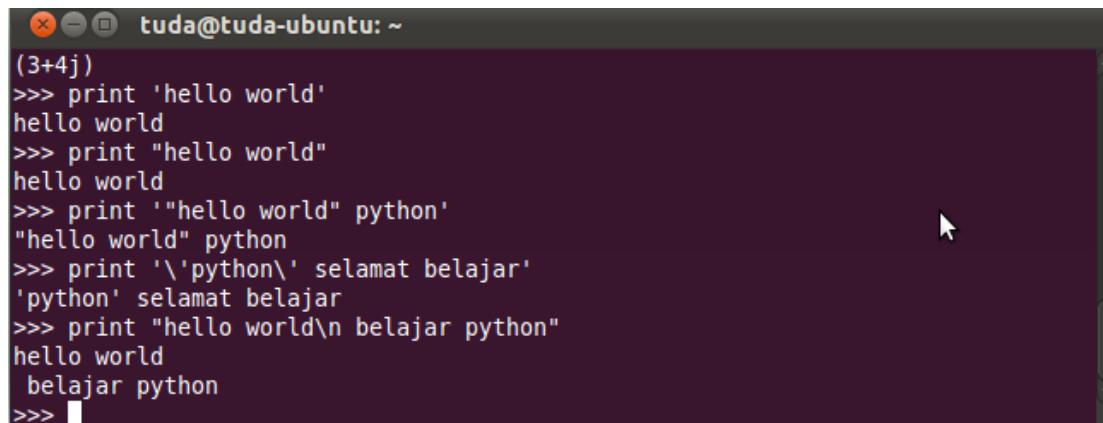
... er, bilangan pecahan (floating point), dan bilangan kompleks. Bilangan kompleks bisa dituliskan dengan for...

... n tanpa dijelaskan sudah bisa dimengerti dan contoh berikut:

Tipe data String

Penulisan string bisa ditulis dengan cara, antara lain :

1. Diapit dengan tanda petik tunggal (').
2. Diapit dengan tanda petik ganda (").
3. Diapit dengan tiga tanda petik ganda (""").



```
tuda@tuda-ubuntu: ~
(3+4j)
>>> print 'hello world'
hello world
>>> print "hello world"
hello world
>>> print '"hello world" python'
"hello world" python
>>> print '\python\' selamat belajar'
'python' selamat belajar
>>> print "hello world\n belajar python"
hello world
belajar python
>>>
```

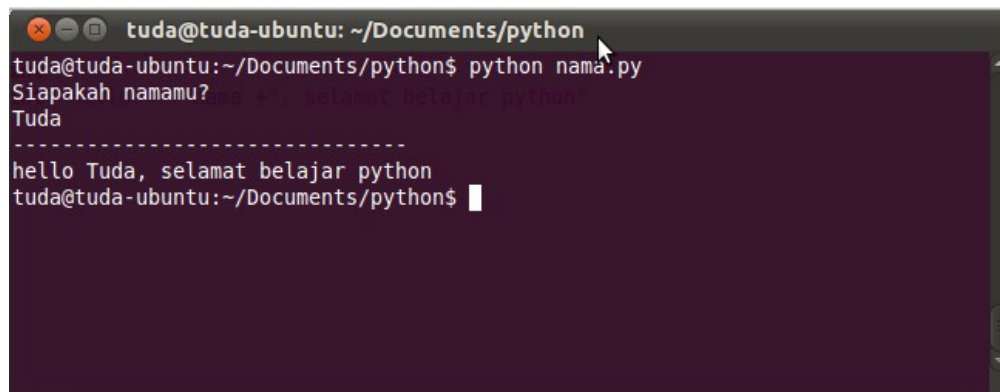
Menggunakan whitespace character

| | |
|----|-------------------------------------|
| \n | garis baru |
| \r | menghapus string sebelumnya |
| \t | tab |
| \v | vertikal tab |
| \e | escape |
| \f | karakter sesudahnya pada garis baru |
| \b | backspace |
| \a | bell |

Contoh Penggunaan Variabel

```
#!/usr/bin/python
```

```
nama = raw_input("Siapakah namamu?\n")
print "-----"
print "hello " + nama + ", selamat belajar python"
```

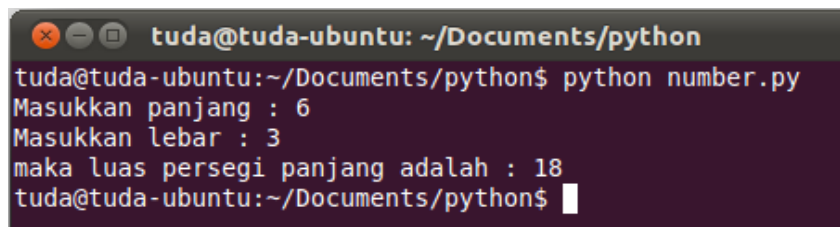


```
tuda@tuda-ubuntu: ~/Documents/python
tuda@tuda-ubuntu:~/Documents/python$ python nama.py
Siapakah namamu?
Tuda
-----
hello Tuda, selamat belajar python
tuda@tuda-ubuntu:~/Documents/python$
```

```
#!/usr/bin/python
```

```
p = input('Masukkan panjang : ')
l = input('Masukkan lebar : ')
luas = p * l

print "maka luas persegi panjang adalah : %d" % luas
```



```
tuda@tuda-ubuntu: ~/Documents/python
tuda@tuda-ubuntu:~/Documents/python$ python number.py
Masukkan panjang : 6
Masukkan lebar : 3
maka luas persegi panjang adalah : 18
tuda@tuda-ubuntu:~/Documents/python$
```

Berikut ini penamaan variabel yang tepat :

+ alamat1
nama_karyawan
nama_variabel_ini_sangat_panjang

Berikut ini penamaan variabel yang salah :

⊘ #alamat
1Nomor

Operator Pada Bahasa Pemrograman Python

Sumber artikel :

<http://www.pintarcoding.com/2014/03/operator-pada-bahasa-pemrograman-python.html>

Pendahuluan

Pembahasan kali ini kita akan belajar bersama tentang penggunaan beberapa operator yang ada pada bahasa pemrograman python. Apa sih operator itu? Agak susah memang menjelaskan operator tersebut, kalau menurut saya sendiri operator itu merupakan suatu perintah yang dapat menghubungkan antara data / variable dengan data / variable yang lain, ya itu memang kalimat saya itu tidak ada dasarnya. Tapi untuk lebih jelasnya yuks kita kupas bersama operator pada bahasa pemrograman python.

Ada beberapa tipe operator antara lain operator aritmatika, operator untuk membandingkan, operator logika seperti and dan or, dan mungkin masih banyak lagi yang tidak bisa saya sebutkan satu persatu, nanti teman-teman tambahkan sendiri ya kalau masih ada yang tidak disebutkan.

Operator Aritmatika

Operator aritmatika sendiri merupakan operator yang dipergunakan dalam melakukan **operasi matematika**, seperti *pengurangan, pembagian, penambahan, perkalian, pangkat, modulus* dll. Berikut contoh penggunaan operator aritmatika pada bahasa pemrograman python :

```
print 3 + 2 * 3 - 10 / 5
print (3 + 2) * (3 - 10 / 5)

#pangkat
print 3 ** 2

#modulus / sisa bagi
print 100 % 3
```

Outputnya :

```
7
5
9
1
```

Perhatikan perintah pada baris pertama **tanpa harus menuliskan tanda kurung pun python sudah melakukan operasi matematika sesuai dengan aturan prioritas matematika.**

Operator yang biasa dipergunakan pada aritmatika ini pun beberapa dapat dipergunakan dalam melakukan manipulasi string contohnya :

```
print 'Pintarcoding.com! ' * 3
print 'hallo' + ' ' + 'Pintarcoding.com!'
```

outputnya :

```
Pintarcoding.com! Pintarcoding.com! Pintarcoding.com!
hallo Pintarcoding.com!
```

Operator Pembandingan

Ada beberapa operator pembandingan antara lain, seperti contoh berikut :

```
print 10 > 2      # lebih besar, bernilai True
print 10 < 2      # lebih kecil, bernilai False
print 10 != 2     # tidak sama dengan, bernilai True
print 5 >= 5      # lebih besar atau sama dengan, bernilai True
print 5 <= 4      # lebih kecil atau sama dengan, bernilai False
print 5 == 5      # sama dengan, bernilai True
```

outputnya :

```
True
False
True
True
False
True
```

Operator pembandingan ini biasanya untuk menentukan nilai kebenaran dalam perintah if, yang akan dibahas pada artikel berikut tentang percabangan.

Operator Logika

Pada bahasa python terdapat 3 operator logika antara lain **and**, **or** dan **not**. Adapun cara penggunaan dari operator-operator tersebut adalah sebagai berikut :

```
print 10 > 2 and 10 > 5    # bernilai True
print True or False       # bernilai True
print not False            # bernilai True
```

outputnya :

```
True
True
True
```

Teman-teman bisa menuliskan perintah diatas untuk mencobanya dikomputer teman-teman dan ketika dijalankan maka akan menghasilkan nilai True pada semua baris perintah diatas.

Operator dan Ekspresi

Sumber artikel :

http://workshop-python-101.readthedocs.org/en/latest/operator_dan_ekspresi.html

Pendahuluan

Hampir semua statemen (baris logis) yang Anda tulis akan mengandung ekspresi. Contoh sederhana dari ekspresi adalah $2+3$. Sebuah ekspresi dapat diturunkan menjadi operator dan operand.

Operator adalah fungsi yang menjalankan sesuatu dan direpresentasikan oleh simbol, seperti $+$ atau kata kunci khusus. Operator membutuhkan data untuk dioperasikan dan data ini disebut operand.

Dalam kasus ini 2 dan 3 adalah operand.

Operator

Kita akan melihat operator secara singkat dan bagaimana penggunaannya:

| Operator | Keterangan |
|----------|---|
| $+$ | Menambahkan dua obyek |
| $-$ | Mengurangi obyek dengan obyek yang lain |
| $*$ | Perkalian |
| $**$ | Pangkat |
| $/$ | Pembagian |
| $//$ | Pembagian bulat ke bawah |
| $\%$ | Sisa hasil bagi (modulus) |
| $<<$ | (geser kiri) Menggeser bit ke sebelah kiri sesuai dengan jumlah bit yang ditentukan. $2 << 2$ menghasilkan 8. 2 direpresentasikan 10 dalam bit (binary digit). Menggeser 2 bit kekiri akan menghasilkan 1000 yang merupakan representasi dari desimal 8. |
| $>>$ | (geser kanan) Menggeser bit ke sebelah kanan sesuai dengan jumlah bit yang ditentukan. $11 > 1$ menghasilkan 5. 11 direpresentasikan oleh bit dengan 1101 kemudian digeser kekanan 1 bit menghasilkan 101 yang merupakan desimal angka 5. |
| $\&$ | (bit-wise AND) Operasi bit-wise AND dari angka (bit-wise adalah operasi angka berbasis bit yakni dengan 0 dan 1). $5 \& 3$ menghasilkan 1. |
| $ $ | (bit-wise OR) Operasi bit-wise OR dari angka. $5 3$ menghasilkan 7. |
| \wedge | (bit-wise XOR) Operasi bit-wise XOR (eksklusif OR). $5 \wedge 3$ menghasilkan 6. |
| \sim | (bit-wise invert) Operasi membalikkan angka bitwise dari x, menghasilkan $-x - 1$. ~ 5 akan menghasilkan -6. lihat two's complement. |
| $<$ | (kurang dari) Mengembalikan apakah x kurang dari y. Semua operator perbandingan mengembalikan True atau False. $5 <$ |

| Operator | Keterangan |
|----------|---|
| | 3mengembalikan False, $3 < 5$ mengembalikan True dan $2 < 5 < 7$ mengembalikan True. |
| > | (lebih dari) Mengembalikan apakah x lebih dari y. $5 > 3$ mengembalikan True. |
| <= | (kurang dari atau sama dengan) Mengembalikan apakah x kurang dari atau sama dengan y. $5 <= 5$ mengembalikan True. |
| >= | (lebih dari atau sama dengan) Mengembalikan apakah x lebih dari atau sama dengan y. $5 >= 5$ mengembalikan True. |
| == | (sama dengan) Membandingkan apakah kedua obyek sama. $2 == 2$ mengembalikan True, $'nama' == 'Nama'$ mengembalikan False, $'nama' == 'nama'$ mengembalikan True. |
| != | (tidak sama dengan) Membandingkan apakah kedua obyek berbeda. $2 != 3$ mengembalikan True. |
| not | (boolean NOT) Jika x bernilai True akan mengembalikan False. Jika x bernilai False akan mengembalikan True. $x = \text{True}; \text{not } x$ mengembalikan False. |
| and | (boolean AND) x and y mengembalikan False jika x bernilai False, selain itu akan mengembalikan nilai y. $x = \text{False}; y = \text{True}; x \text{ and } y$ akan mengembalikan False karena x bernilai False. Pada kasus ini Python tidak akan mengevaluasi y karena nilai x. Hal ini disebut short-circuit evaluasi. |
| or | (boolean OR) Jika x bernilai True, x or y akan mengembalikan True, selain itu akan mengembalikan nilai y. $x = \text{True}; y = \text{False}; x \text{ or } y$ mengembalikan True. short-circuit evaluasi berlaku juga disini. |

```
# lat6.py
# Operator dan ekspresi

bilangan1 = 5
bilangan2 = 3

print 'bil1 = ', bilangan1
print 'bil2 = ', bilangan2

print 'bil1 + bil2 = ', bilangan1 + bilangan2
print 'bil1 - bil2 = %s' % (bilangan1 - bilangan2)
print 'bil1 * bil2 = {0}'.format(bilangan1 * bilangan2)
```

```

print 'bil1 ** bil2 = ', bilangan1 ** bilangan2

bilangan1 = 5.0
print 'bil1 = ', bilangan1
print 'bil2 = ', bilangan2

print 'bil1 / bil2 = ', bilangan1 / bilangan2
print 'bil1 // bil2 = ', bilangan1 // bilangan2
print 'bil1 % bil2 = ', bilangan1 % bilangan2

print '-' * 80

bilangan1 = 5
print 'bil1 = ', bilangan1
print 'bil2 = ', bilangan2

print 'bil1 << bil2 = ', bilangan1 << bilangan2
print 'bil1 >> bil2 = ', bilangan1 >> bilangan2
print 'bil1 & bil2 = ', bilangan1 & bilangan2
print 'bil1 | bil2 = ', bilangan1 | bilangan2
print 'bil1 ^ bil2 = ', bilangan1 ^ bilangan2
print '~bil1 = ', ~bilangan1

print '-' * 80

print 'bil1 < bil2 = ', bilangan1 < bilangan2
print 'bil1 > bil2 = ', bilangan1 > bilangan2
print 'bil1 <= bil2 = ', bilangan1 <= bilangan2
print 'bil1 >= bil2 = ', bilangan1 >= bilangan2
print 'bil1 == bil2 = ', bilangan1 == bilangan2
print 'bil1 != bil2 = ', bilangan1 != bilangan2

print '-' * 80

print 'not True = ', not True
print 'True and False = ', True and False
print 'True or False = ', True or False

```

Operasi matematika dan pengisian variabel

Ketika melakukan operasi matematika kita sering setelah dilakukan operasi hasil tersebut kita simpan dalam variabel. Di python ada jalan pintas untuk melakukan operasi dan melakukan assignment.

Anda bisa menulis:

```
a = 2
a = a * 3
```

sebagai:

```
a = 2
a *= 3
```

Berikut latihan 7 untuk menghitung uang kembalian.

```
# lat7.py

total_uang = 10000
harga_barang = 5000
diskon = 0.10

# harga barang setelah diskon
harga_barang *= (1 - diskon)

total_uang -= harga_barang

print 'total uang = %s' % total_uang
```

Urutan Evaluasi

Jika ada rantai ekspresi seperti $2 + 3 * 4$, apakah penambahan dilakukan terlebih dahulu atau perkalian? Saat pelajaran matematika kita diajari bahwa perkalian harus dikerjakan terlebih dahulu. Hal ini menandakan perkalian mempunyai urutan lebih tinggi daripada penambahan.

Berikut tabel urutan evaluasi ekspresi dalam Python, dari terendah sampai tertinggi.

| Operator | Keterangan |
|----------------------|-----------------|
| lamda | Ekspresi lambda |
| or | Boolean OR |
| and | Boolean AND |
| not x | Boolean NOT |
| in, not in | Tes Keanggotaan |
| is, is not | Tes Identitas |
| <, <=, >, >=, !=, == | Perbandingan |
| | Bitwise OR |
| ^ | Bitwise XOR |

| Operator | Keterangan |
|------------------|---|
| & | Bitwise AND |
| <<, >> | Shift |
| +, - | Penambahan dan Pengurangan |
| *, /, //, % | Perkalian, Pembagian, Pembagian ke bawah, mod |
| +x, -x | Positif, Negatif |
| ~x | Bitwise NOT / inverse |
| ** | Pangkat |
| x.attribute | Referensi atribut |
| x[index] | Akses item |
| x[index1:index2] | Slicing |
| f(argument ...) | Pemanggilan fungsi |
| (ekspresi, ...) | literal tuple |
| [ekspresi, ...] | literal list |
| {key:value, ...} | literal dictionary |

Mengubah Urutan Evaluasi

Untuk membuat ekspresi lebih mudah dibaca, kita dapat menggunakan tanda kurung. Sebagai contoh, $2 + (3 * 4)$ lebih mudah dipahami daripada $2 + 3 * 4$ dimana pembaca harus mengetahui urutan evaluasi operator. Namun pemakaian tanda kurung jangan terlalu berlebihan seperti $(2 + (3 * 4))$.

Selain itu, tanda kurung dapat mengubah urutan evaluasi operator. Sebagai contoh $(2 + 3) * 4$, operasi penambahan akan dievaluasi terlebih dahulu.

```
# lat8.py

hasil = 2 + 3 * 4
print '2 + 3 * 4 = %s' % hasil

hasil = (2 + 3) * 4
print '(2 + 3) * 4 = %s' % hasil

hasil = 2 / 3 * 4
print '2 / 3 * 4 = %s' % hasil

hasil = 2.0 / 3 * 4
print '2.0 / 3 * 4 = %s' % hasil
```

Sifat Asosiatif

Operator dengan level urutan evaluasi yang sama akan dievaluasi dari kiri ke kanan. Sebagai contoh $2 + 3 + 4$ akan dievaluasi sebagai $(2 + 3) + 4$. Beberapa operator seperti pengisian nilai (assignment) mempunyai sifat asosiatif dari kanan ke kiri, contoh: $a = b = c$ akan dievaluasi $a = (b = c)$.

Penggunaan If Pada Python

Sumber :

<http://www.pintarcoding.com/2014/03/percabangan-bahasa-pemrograman-python.html#more>

<http://sikecil91.blogspot.com/2014/03/if-pada-python.html>

Pendahuluan

Seperti halnya bahasa pemrograman yang lain, tentu python juga mempunyai perintah untuk pengambilan suatu keputusan terhadap kondisi tertentu, yang disebut percabangan. Percabangan pada bahasa pemrograman python menggunakan perintah if, ya sama dengan bahasa pemrograman yang lain. Bagaimana cara menggunakan perintah if ini dalam bahasa pemrograman python? Yuk mari kita sama-sama melihat cara penggunaan perintah if ini.

Menggunakan IF Tunggal

```
n= raw_input("Yakin (y/t)?")  
  
if n in ["y","Y"]:  
    print "Baiklah"
```

outputnya :

```
Yakin (y/t) ? y  
Baiklah
```

Menggunakan IF ELSE

Contoh berikut digunakan untuk menentukan Ganjil / Genap nya sebuah bilangan.

```
#!/usr/bin/python  
  
a = raw_input("Masukkan Angka = ")  
b = int(a)  
  
if ( b%2 == 0 ):  
    print "Genap"  
else:  
    print "Ganjil"
```

outputnya :

```
Masukkan Angka = 2  
Genap
```

Menggunakan ELIF

```
#!/usr/bin/python  
  
y = raw_input("masukkan angka = ")  
x = int(y)  
  
if (x>50 and x<=60):  
    print "Nilai C"
```

```
elif (x>60 and x<=80):  
    print "Nilai B"  
  
elif (x>80 and x<=100):  
    print "Nilai A"  
else:  
    print "Nilai E"
```

outputnya :

```
masukkan angka = 90  
Nilai A
```

Alur Kontrol

Sumber :

http://workshop-python-101.readthedocs.org/en/latest/alur_kontrol.html

Pendahuluan

Di dalam program yang kita lihat hingga saat ini, selalu saja urutan statemen yang dijalankan oleh Python, berurutan dari atas ke bawah. Bagaimana jika Anda ingin mengubah bagaimana alur kerjanya? Sebagai contoh Anda ingin program untuk mengambil keputusan dan bertindak secara berbeda tergantung keputusan yang diambil. Sebagai contoh, misalnya mencetak 'Selamat Pagi' atau 'Selamat Sore' tergantung waktu yang ada saat itu?

Sebagaimana yang Anda sudah bisa tebak, ini dapat dilakukan lewat statemen alur kontrol. Ada tiga macam statemen alur kontrol di Python - **if**, **for** dan **while**.

Statemen If

Statemen **if** digunakan untuk mengecek kondisi: jika kondisi **if** bernilai benar, maka kita akan menjalankan satu blok statemen (disebut **if-block**), jika tidak akan diteruskan dengan statemen **else** kita gunakan untuk memproses blok statemen yang lain (dinamakan **else-block**). Bagian **else** tersebut sifatnya tidak wajib atau opsional.

Kita dapat menambahkan kondisi dalam **else-block** menggunakan **elif**.

```
# lat9.py

nomor_acak = 7
print 'tebak nomor acak dari 1 - 10'

# ``raw_input`` digunakan untuk mendapatkan input dari pengguna
# ``int`` digunakan untuk konversi tipe data ``str`` ke ``int``

tebakan = int(raw_input('Tebakan anda (bil bulat): '))

if tebakn == nomor_acak:
    print 'Selamat! tebakn anda benar'
    print 'tapi tidak ada hadiah untuk anda :( '

elif tebakn < nomor_acak:
    print 'tebakn anda terlalu kecil'

else:
    print 'tebakn anda terlalu besar'

print 'selesai'
```

output :

```
tebak nomor acak dari 1 - 10
Tebakan anda (bil bulat): 7
Selamat! tebakn anda benar
tapi tidak ada hadiah untuk anda :(
selesai
```

Bagaimana program ini bekerja ?

Program ini akan meminta inputan tebakan dari pengguna berupa bilangan ini. Untuk mendapatkan inputan ini kita gunakan fungsi **raw_input**. Keluaran dari fungsi ini adalah string yang diinputkan oleh user, oleh karena itu kita harus **melakukan konversi ke tipe data int**. Untuk konversi ini kita gunakan fungsi **int**. Hasil dari inputan pengguna yang sudah dikonversi disimpan dalam variabel **tebakan**.

Sebelumnya program telah menentukan bilangan acak yang disimpan dalam variabel **nomor_acak**. Setelah mendapatkan input dari pengguna, program masuk kedalam alur kontrol **if**. Jika tebakan dan nomor acak sama maka tampilkan pesan berhasil, jika tebakan kurang dari nomor acak maka tampilkan pesan tebakan terlalu kecil, dan terakhir berarti tebakan terlalu besar.

Statemen While

Statemen **while** merupakan statemen untuk perulangan, block kode akan dijalankan terus menerus selama kondisi benar. Statemen **while** dapat mempunyai bagian **else** (opsional).

```
# lat10.py
# acak looping

nomor_acak = 77
berjalan = True

print 'tebak nomor acak dari 1 - 100'

while berjalan:

    tebakan = int(raw_input('Tebakan anda (bil bulat): '))

    if tebakan == nomor_acak:
        print 'Selamat! tebakan anda benar'
        print 'tapi tidak ada hadiah untuk anda :('
        berjalan = False
    elif tebakan < nomor_acak:
        print 'tebakan anda terlalu kecil'
    else:
        print 'tebakan anda terlalu besar'

else:
    print 'selesai'
```

outputnya :

```
tebak nomor acak dari 1 - 100

Tebakan anda (bil bulat): 20
tebakan anda terlalu kecil

Tebakan anda (bil bulat): 90
tebakan anda terlalu besar

Tebakan anda (bil bulat): 50
```

```
tebakan anda terlalu kecil

Tebakan anda (bil bulat): 77
Selamat! tebakkan anda benar

tapi tidak ada hadiah untuk anda :(
selesai
```

Perulangan diatas berhenti jika berjalan (kondisi) bernilai False. True dan False merupakan obyek bertipe boolean, dan nilai True sama dengan nilai 1, nilai False sama dengan nilai 0.

```
>>> True == 1
True

>>> False == 0
True
```

Obyek dapat dinilai atau dikonversi ke nilai boolean.

```
>>> bool('nama')
True

>>> bool('')
False

>>> bool(0)
False

>>> bool(-5)
True
```

Perulangan For (For Loop)

Statemen perulangan for ... in ... `` merupakan statemen perulangan selain while. Statemen ini melakukan iterasi dari rangkaian obyek, berjalan melalui tiap item yang ada pada rangkaian / sequence. Apa itu rangkaian / sequence? rangkaian yaitu koleksi item yang terurut.

```
# lat11.py

for i in range(1, 6):
    print I

else:
    print 'Perulangan sudah selesai'
```

outputnya :

```
1
2
3
4
```

Program ini akan mencetak rangkaian / sequence bilangan, dari 1 sampai 5. Kita membuat rangkaian bilangan ini menggunakan fungsi builtin range. Apa yang kita lakukan yaitu memanggil fungsi range dengan dua parameter, range akan mengembalikan rangkaian bilangan dari parameter pertama sampai batas parameter kedua (eksklusif). Sebagai contoh range(1, 6) menghasilkan rangkaian [1, 2, 3, 4, 5].

Jika kita memanggil range dengan parameter ketiga, yaitu parameter jumlah langkah. Contoh range(1, 6, 2) mengembalikan rangkaian [1, 3, 5]. Bagian else adalah opsional dan akan selalu dijalankan kecuali jika ada statemen break.

Statemen Break

Statemen break digunakan untuk keluar dari perulangan, misalnya keluar dari perulangan walaupun kondisi perulangan masih True atau rangkaian / sequence belum diiterasi seluruhnya.

```
# lat12.py

while True:
    data = raw_input('Masukkan sesuatu : ')

    if data == 'keluar':
        break

    print 'Inputan pengguna "%s"' % data

print 'Selesai'
```

outputnya :

```
Masukkan sesuatu : alpa
Inputan pengguna "alpa"
Masukkan sesuatu : keluar
Selesai
```

Program ini akan terus meminta inputan pengguna dan akan berhenti ketika pengguna menginputkan keluar.

```
# lat13.py

for i in range(1, 11):
    print i

    if i == 5:
        break

else:
    print "Tidak dijalankan karena break"
```


outputnya :

```
1
2
3
4
5
```

Bagian else tidak akan dijalankan karena perulangan tidak berhenti secara normal.

Statemen Continue

Statemen continue digunakan untuk melewati statemen yang ada dalam blok perulangan dan continue / melanjutkan ke iterasi berikutnya.

```
# lat14.py

for i in range(1, 11):
    if i % 2 == 0:
        # skip bilangan genap
        continue
    print i
```

outputnya :

```
1
3
5
7
9
```

Struktur Data Python

Sumber :

http://sakti.github.io/python101/struktur_data.html

Pendahuluan

Struktur Data adalah struktur yang dapat menyimpan dan mengorganisasikan kumpulan data. Berikut struktur data yang ada dalam Python.

List

List adalah struktur data yang menyimpan koleksi data terurut, anda dapat menyimpan sequence / rangkaian item menggunakan list.

Item dalam list ditutup menggunakan kurung siku [] (list literal). Setelah list dibuat anda bisa menambah, mengurangi, dan mencari item pada list. Karena kita dapat menambah dan mengurangi item, list bersifat **mutable**.

Pengenalan singkat obyek dan class

List adalah contoh penggunaan obyek dan class. Ketika kita menggunakan variabel **i** dan mengisinya dengan nilai integer **5**, sama dengan kita membuat obyek (**instance**) **i** dari class (**type**) **int**. Anda dapat membacanya dengan **help(int)** untuk membaca dokumentasi class integer.

Class mempunyai method, fungsi yang didefinisikan dalam class. Anda bisa menggunakan method ini pada obyek class tersebut. Sebagai contoh, Python menyediakan method **append** untuk class list. **contoh_list.append('item 1')** akan menambahkan string **'item 1'** kedalam list **contoh_list**. Perhatikan notasi titik untuk mengakses method pada obyek.

Class juga mempunyai field yang sama halnya variabel yang digunakan hanya untuk class. Anda bisa menggunakan variabel / nama ini pada obyek class tersebut.

```
# lat25.py

daftar_belanja = ['apel', 'mangga', 'wortel', 'pisang']
print 'saya punya %s barang yang akan dibeli' % len(daftar_belanja)

print 'barang tersebut:'
for barang in daftar_belanja:
    print barang,

print 'saya harus membeli beras'
daftar_belanja.append('beras')
print 'daftar belanja sekarang :', daftar_belanja

print 'saya akan mengurutkan daftar belanja saya'
daftar_belanja.sort()
print 'daftar belanja setelah diurutkan', daftar_belanja

print 'barang yang harus saya beli pertama', daftar_belanja[0]
barang_pertama = daftar_belanja[0]

del daftar_belanja[0]
```

```
print 'saya membeli', barang_pertama
print 'daftar belanja sekarang:', daftar_belanja
```

outputnya :

```
saya punya 4 barang yang akan dibeli
```

```
barang tersebut:
apel mangga wortel pisang saya harus membeli beras
daftar belanja sekarang : ['apel', 'mangga', 'wortel', 'pisang', 'beras']
saya akan mengurutkan daftar belanja saya
daftar belanja setelah diurutkan ['apel', 'beras', 'mangga', 'pisang', 'wortel']
barang yang harus saya beli pertama apel
saya membeli apel
daftar belanja sekarang: ['beras', 'mangga', 'pisang', 'wortel']
```

Tuple

Tuple mirip dengan list namun tuple bersifat **immutable** (tidak bisa diubah setelah didefinisikan). Tuple dibuat dengan menspesifikasikan item tuple dipisahkan menggunakan tanda koma dan opsional diapit dengan tanda kurung.

```
# lat26.py

kebun_binatang = ('ular python', 'gajah', 'penguin')
print 'jumlah binatang yang ada di kebun binatang :', len(kebun_binatang)

kebun_binatang_baru = 'monyet', 'unta', kebun_binatang
print 'jumlah kandang di kebun binatang baru:', len(kebun_binatang_baru)
print 'binatang yang ada di kebun binatang baru:', kebun_binatang_baru
print 'binatang dari kebun binatang lama:', kebun_binatang_baru[2]
print 'binatang terakhir dari kebun binatang lama:', kebun_binatang_baru[2][2]

jumlah_binatang = len(kebun_binatang_baru) - 1 + len(kebun_binatang_baru[2])

print 'jumlah binatang yang ada di kebun binatang baru :', jumlah_binatang
```

outputnya :

```
jumlah binatang yang ada di kebun binatang : 3
jumlah kandang di kebun binatang baru: 3
binatang yang ada di kebun binatang baru: ('monyet', 'unta', ('ular python',
'gajah', 'penguin'))
binatang dari kebun binatang lama: ('ular python', 'gajah', 'penguin')
binatang terakhir dari kebun binatang lama: penguin
jumlah binatang yang ada di kebun binatang baru : 5
```

Dictionary

Dictionary seperti buku alamat, dengan buku alamat anda bisa mencari alamat atau detail kontak hanya menggunakan nama orang yang anda cari. Kita mengasosiasikan **key** (nama) dengan **value** (detail). Catatan **key** harus bersifat unik, anda tidak bisa menemukan informasi yang tepat jika ada dua orang yang mempunyai nama yang sama dalam buku alamat anda.

Anda hanya bisa menggunakan obyek immutable (seperti string) untuk **key** / kunci dictionary. Anda bisa menggunakan obyek mutable atau immutable untuk **value** dalam dictionary.

Dictionary dispesifikasikan menggunakan pasangan **key** dan **value** diapit menggunakan kurung kurawal, {**key1: value1, key2: value2**}.

```
# lat27.py

# ba, singkatan buku alamat
ba = {'guido': 'guido@python.org',\
      'brandon': 'brandon@rhodesmill.org',\
      'spammer': 'spammer@domain.com'}

print 'alamat email guido:', ba['guido']

# menghapus item
del ba['spammer']

print 'ada %s kontak di buku alamat' % len(ba)

for nama, email in ba.items():
    print '%s, email: %s' % (nama, email)

# tambah entri
ba['jacob'] = 'jacob@jacobian.org'

if 'jacob' in ba:
    print 'Email jacob di', ba['jacob']
```

outputnya :

```
alamat email guido: guido@python.org
ada 2 kontak di buku alamat
brandon, email: brandon@rhodesmill.org
guido, email: guido@python.org
Email jacob di jacob@jacobian.org
```

Sequence

List, tuple dan string adalah contoh dari sequence. Kita dapat melakukan tes keanggotaan, operasi index(akses, slicing), dan iterasi pada sequence.

```
# lat28.py

daftar_belanja = ['apel', 'mangga', 'wortel', 'pisang']
nama = 'budi'

print 'Barang 0 =', daftar_belanja[0]
print 'Barang 1 =', daftar_belanja[1]
print 'Barang 2 =', daftar_belanja[2]
print 'Barang 3 =', daftar_belanja[3]

print 'Barang -1 =', daftar_belanja[-1]
print 'Barang -2 =', daftar_belanja[-2]

print 'Karakter 0 =', nama[0]

# slicing pada list
print 'Barang 1 ke 3:', daftar_belanja[1:3]
print 'Barang 2 ke terakhir:', daftar_belanja[2:]
print 'Barang 1 ke -1:', daftar_belanja[1:-1]
print 'Barang dari awal ke akhir:', daftar_belanja[:]

# slicing pada string
print 'Karakter 1 ke 3:', nama[1:3]
print 'Karakter 2 ke terakhir:', nama[2:]
print 'Karakter 1 ke -1:', nama[1:-1]
print 'Karakter dari awal ke akhir:', nama[:]
```

outputnya :

```
Barang 0 = apel
Barang 1 = mangga
Barang 2 = wortel
Barang 3 = pisang
Barang -1 = pisang
Barang -2 = wortel
Karakter 0 = b
Barang 1 ke 3: ['mangga', 'wortel']
Barang 2 ke terakhir: ['wortel', 'pisang']
Barang 1 ke -1: ['mangga', 'wortel']
Barang dari awal ke akhir: ['apel', 'mangga', 'wortel', 'pisang']
Karakter 1 ke 3: ud
Karakter 2 ke terakhir: di
Karakter 1 ke -1: ud
Karakter dari awal ke akhir: budi
```

Set

Set adalah koleksi obyek yang tidak terurut. Digunakan ketika keberadaan obyek pada koleksi lebih penting daripada urutan dan berapa kali obyek muncul pada koleksi.

```
# lat29.py
negara = set(['brazil', 'rusia', 'indonesia'])

print 'indonesia' in negara
print 'amerika' in negara

negara2 = negara.copy()
negara2.add('korea')

print negara2.issuperset(negara)

negara.remove('rusia')

print negara2 & negara
print negara2.intersection(negara)
```

outputnya :

```
True
False
True
set(['brazil', 'indonesia'])
set(['brazil', 'indonesia'])
```

Referensi

Jika anda membuat obyek dan mengisinya ke variabel, variabel hanya me-**refer** ke obyek dan tidak merepresentasikan obyek itu sendiri. Nama variabel menunjuk ke bagian memori komputer dimana obyek disimpan. Hal ini dinamakan **binding** antara nama ke obyek.

```
# lat29.py
daftar_belanja = ['apel', 'mangga', 'wortel', 'pisang']
print 'assignment biasa'
daftar_saya = daftar_belanja

del daftar_belanja[0]

print 'daftar belanja:', daftar_belanja
print 'daftar saya:', daftar_saya

print 'copy obyek daftar belanja menggunakan slice [:]'
daftar_saya = daftar_belanja[:] # membuat copy

del daftar_saya[0]

print 'daftar belanja:', daftar_belanja
print 'daftar saya:', daftar_saya
```

outputnya :

```
assignment biasa
daftar belanja: ['mangga', 'wortel', 'pisang']
daftar saya: ['mangga', 'wortel', 'pisang']
copy obyek daftar belanja menggunakan slice [:]
daftar belanja: ['mangga', 'wortel', 'pisang']
daftar saya: ['wortel', 'pisang']
```

String

Tipe atau class String mempunyai method-method untuk memudahkan operasi string.

```
# lat30.py

nama = 'Indonesia'

if nama.lower().startswith('ind'):
    print 'Nama diawal dengan "ind"'
if 'ne' in nama:
    print 'Nama berisi string "ne"'
if nama.find('done') != -1:
    print 'Nama berisi string "done"'

pembatas = ', '
daftar_belanja = ['apel', 'mangga', 'wortel', 'pisang']

print pembatas.join(daftar_belanja)
```

outputnya :

```
Nama diawal dengan "ind"
Nama berisi string "ne"
Nama berisi string "done"
apel, mangga, wortel, pisang
```


List di Python

Sumber : http://www.tutorialspoint.com/python/python_lists.htm

Pendahuluan

Struktur data dasar yang paling umum dalam Python adalah sequence (urutan). Setiap elemen yang berurutan tersebut memiliki sebuah alamat berbentuk nomor biasanya disebut indeks. Indeks pertama adalah nol, indeks kedua adalah satu, dan seterusnya.

Python memiliki enam built-in tipe Sequence, tetapi yang paling umum digunakan adalah **List []** dan **Tupel ()**.

di dalam menggunakan Sequence, Anda dapat melakukan operasi pengindeksan, pembagian (pengirisan), penambahan, perkalian, dan pemeriksaan anggota pada Sequence. Selain itu, Python juga memiliki fungsi dan metode built-in untuk mendapatkan panjang sebuah urutan, mencari elemen-elemen di dalam sebuah urutan, mengurutkan elemen-elemen tersebut dari yang terbesar ke yang terkecil dan masih banyak lagi.

List Python

List merupakan sekumpulan data dengan Tipe data apa saja yang di tulis di dalam sebuah tanda **kurung siku []** yang tiap anggota (elemen) di pisahkan dengan **tanda koma (,)**. berikut ini adalah contoh bagaimana List dibuat.

```
list1 = ['doni kusuma', 'anton wijaya', 1997, 2000];  
list2 = [1, 2, 3, 4, 5 ];  
list3 = ["a", "b", "c", "d"];
```

Mengakses Nilai di dalam List

Untuk mengakses nilai dalam List, gunakan tanda kurung siku [] dengan memberikan nilai indeks sesuai elemen yang ingin di ambil, Berikut adalah contohnya :

```
list1 = ['doni kusuma', 'anton wijaya', 1997, 2000]  
list2 = [1, 2, 3, 4, 5 ]  
list3 = ["a", "b", "c", "d"]  
  
print list1[1]  
print list2[1:3]
```

outputnya :

```
anton wijaya  
[2, 3]
```

untuk tanda [1:3] artinya ambil elemen pada list2 mulai dari elemen 1 sampai elemen 2.

Memperbarui List

Anda dapat memperbarui satu atau beberapa elemen dari List dengan memberikan indeks elemen di dalam tanda kurung siku di sebelah kiri operator tugas (=), dan Anda juga dapat menambahkan elemen dalam List dengan menggunakan metode `append()`. Berikut ini adalah contohnya :

```
list = ['doni kusuma', 'anton wijaya', 1997, 2000]

print "Nilai yang ada di index 2 : "
print list[2];
list[2] = 2001;
print "Nilai yang ada di index 2 : "
print list[2];
```

outputnya :

```
Nilai yang ada di index 2 :
1997
Nilai yang ada di index 2 :
2001
```

Menghapus Elemen List

Untuk menghapus sebuah elemen dari List, Anda dapat menggunakan pernyataan **`del`** jika Anda tahu persis elemen yang akan dihapus dari List atau dengan metode **`remove()`** jika Anda tidak tahu elemen yang ingin Anda hapus di List. Berikut ini adalah contohnya :

```
list = ['doni kusuma', 'anton wijaya', 1997, 2000]

print list;
del list[2];
print "Setelah nilai dari indeks ke 2 dihapus : "
print list;
```

outputnya :

```
['doni kusuma', 'anton wijaya', 1997, 2000]

Setelah nilai dari indeks ke 2 dihapus :

['doni kusuma', 'anton wijaya', 2000]
```

Operasi Dasar List

List menanggapi operator berupa **+** dan ***** dimana operator ini banyak digunakan dalam pemrosesan String, arti operator **+** berarti **menggabungkan dua List atau beberapa List menjadi satu List** dan operator ***** artinya **List akan direpetasi (diulang) sebanyak N**. Hasil yang akan diperoleh *dari proses operator ini tetap menjadikannya List dan bukan sebuah String*.

| Ekspresi Pada Python | Hasil | Deskripsi |
|---|---|--|
| <code>len([1, 2, 3])</code> | 3 | Menghitung banyaknya elemen-elemen di dalam sebuah List. |
| <code>[1, 2, 3] + [4, 5, 6]</code> | <code>[1, 2, 3, 4, 5, 6]</code> | Menggabungkan List. |
| <code>['Hi!'] * 4</code> | <code>['Hi!', 'Hi!', 'Hi!', 'Hi!']</code> | Merepeat (mengulang) elemen List sebanyak 4 kali. |
| <code>3 in [1, 2, 3]</code> | True | Mengecek ke anggotaan elemen pada suatu List. Apakah elemen 3 ada di dalam List [1, 2, 3] jika ada, hasilnya True. |
| <code>for x in [1, 2, 3]: print x,</code> | 1 2 3 | Iterasi, dimana terjadi perulangan sebanyak elemen List. X adalah variabel yang menampung satu elemen List dalam satu kali putaran perulangan. |

Pengindeksan, pengirisan dan matrixes

Karena List adalah Sequences (Urutan), pengindeksan dan pengirisan bekerja dengan cara yang sama seperti yang dilakukan untuk string.

Asumsikan terdapat List seperti berikut :

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

| Expresi pada Python | Hasil | Deskripsi |
|---------------------|--------------------------------|--|
| <code>L[2]</code> | <code>'SPAM!'</code> | Offset dimulai dari nol |
| <code>L[-2]</code> | <code>'Spam'</code> | Nilai negatif: menghitung dari kanan |
| <code>L[1:]</code> | <code>['Spam', 'SPAM!']</code> | Pengirisan mulai dari index ke 1 sampai indeks paling akhir. |

Fungsi Built-in List

Fungsi cmp

Membandingkan elemen-elemen dari dua List

```
cmp(list1, list2)
```

1. list1 : Ini adalah daftar pertama **yang dibandingkan** dengan list2.
2. list2 : Ini adalah daftar kedua **yang akan dibandingkan** dengan list1.

Jika elemen-elemennya dari jenis yang sama, program akan membandingkan dan mengembalikan hasil. Jika elemen-elemennya campuran, program akan memeriksa apakah elemen-elemennya merupakan angka.

```
#!/usr/bin/python

list1, list2 = [123, 'xyz'], [456, 'abc']

print cmp(list1, list2);
print cmp(list2, list1);
list3 = list2 + [786];
print cmp(list2, list3)
```

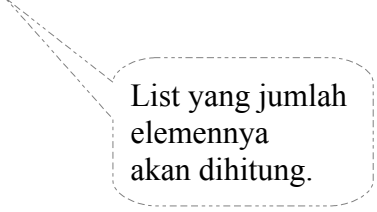
outputnya :

```
-1
1
-1
```

Fungsi len

Mengembalikan nilai berupa jumlah elemen dalam List.

```
len(list)
```



List yang jumlah
elemennya
akan dihitung.

```
#!/usr/bin/python

list1, list2 = [123, 'xyz', 'zensu'], [456, 'abc']

print "Panjang List pertama : ", len(list1);
print "Panjang List kedua : ", len(list2);
```

outputnya :

```
Panjang List pertama : 3
Panjang List kedua : 2
```

Fungsi max

Mencari dan Mengembalikan elemen maksimum dari sebuah List.

```
max(list)
```

paramater list merupakan List yang akan di cari elemen maksimumnya.

```
#!/usr/bin/python

list1, list2 = [123, 'xyz', 'zensu', 'abc'], [456, 700, 200]

print "Elemen Maksimum List Pertama \t: ", max(list1);
print "Elemen Maksimum List Kedua \t: ", max(list2);
```

ouputnya :

```
Elemen Maksimum List Pertama      : zensu
Elemen Maksimum List Kedua        : 700
```

Fungsi min

Mencari dan Mengembalikan elemen minimum dari sebuah List.

```
max(list)
```

paramater list merupakan List yang akan di cari elemen minimumnya.

```
#!/usr/bin/python

list1, list2 = [123, 'xyz', 'zensu', 'abc'], [456, 700, 200]

print "Elemen Maksimum List Pertama \t: ", min(list1);
print "Elemen Maksimum List Kedua \t: ", min(list2);
```

outputnya :

```
Elemen Maksimum List Pertama      : 123
Elemen Maksimum List Kedua        : 200
```

Fungsi list

Mengambil jenis Sequences (urutan) lain dan merubahnya ke dalam bentuk List. **Ini digunakan untuk mengkonversi Tuple ke dalam List.**

Tupel sangat mirip dengan **List**, perbedaannya terletak pada nilai-nilai elemen **Tupel**, ia **tidak dapat diubah** dan **unsur-unsur Tupel diletakkan antara tanda kurung biasa ()** bukan kurung siku [].

```
list( seq )
```

seq merupakan parameter yang dinyatakan dalam bentuk Tuple.

```
#!/usr/bin/python

ini_tuple = (123, 'xyz', 'zensu', 'abc');
ini_list = list(ini_tuple)

print "List elements : ", ini_list
```

outputnya :

```
List elements : [123, 'xyz', 'zensu', 'abc']
```

Metode Built-in List

Metode append

menambahkan sebuah objek ke List yang ada.

```
list.append(obj)
```

obj adalah parameter dimana ia merupakan objek (string, angka dll) yang akan dimasukan ke dalam List.

```
#!/usr/bin/python

ini_list = [123, 'xyz', 'zensu', 'abc'];
ini_list.append( "Junet" );
print "mengupdated List : ", ini_list;
```

outputnya :

```
mengupdated List : [123, 'xyz', 'zensu', 'abc', 'Junet']
```

Metode count

mengembalikan nilai hitungan **berapa kali objek terdaftar di dalam List**.

```
list.count(obj)
```

obj merupakan parameter dimana ini adalah objek yang akan dihitung dalam List.

```
#!/usr/bin/python

ini_list = [123, 'xyz', 'zensu', 'beni', 'zensu'];

print "beni di List sebanyak\t: ", ini_list.count('beni');
print "zensu di List sebanyak \t: ", ini_list.count('zensu');
```

outputnya :

```
beni di List sebanyak      : 1
zensu di List sebanyak     : 2
```


Metode extend

Menambahkan urutan (sequence) elemen (isi) ke sebuah List.

```
list.extend(seq)
```

seq ialah parameter yang isinya merupakan elemen List.

```
#!/usr/bin/python

aList = [123, 'xyz', 'zensu', 'abc', 123];
bList = [2009, 'yanzen'];

aList.extend(bList)

print "Mengextend List : ", aList ;
```

outputnya :

```
Mengextend List : [123, 'xyz', 'zensu', 'abc', 123, 2009, 'yanzen']
```

Metode index

Mengembalikan posisi index dari objek di dalam List.

```
list.index(obj)
```

obj ialah parameter yang merupakan objek (string, angka dll) yang akan di cari indeks nya dalam sebuah List.

```
#!/usr/bin/python

ini_list = [123, 'xyz', 'zensu', 'abc']

print "Indeks untuk xyz \t: ", ini_list.index( 'xyz' )
print "Indeks untuk zensu \t: ", ini_list.index( 'zensu' )
```

outputnya :

```
Indeks untuk xyz      : 1
Indeks untuk zensu    : 2
```

Metode insert

menyisipkan objek **obj** (param list) ke dalam List pada indeks (**index**) yang ditentukan.

```
list.insert(index, obj)
```

1. **index** ialah parameter dimana obj nantinya akan ditempatkan.
2. **obj** ialah parameter dimana isi dari nilai elemen yang baru yang akan di masukan ke sebuah List berdasarkan parameter index nya.

```
#!/usr/bin/python

aList = [123, 'xyz', 'zensu', 'abc']

aList.insert(3, 2009)

print "Final List : ", aList
```

kode di atas artinya memasukan objek 2009 ke dalam aList di index ke 3.

ouputnya :

```
Final List : [123, 'xyz', 'zensu', 2009, 'abc']
```

Metode pop

Menghapus elemen (objek) dari List lalu mengembalikannya ke List saat setelah penghapusan.

```
list.pop(obj=list[-1])
```

obj ialah parameter dimana defaultnya ia merujuk pada list yang paling kanan, bersifat opsional.

```
#!/usr/bin/python

ini_list = [123, 'xyz', 'zensu', 'abc'];

print "List POP -1 \t: ", ini_list.pop();
print "List POP 2 \t: ", ini_list.pop(2);
```

outputnya :

```
List POP -1 : abc
List POP 2 : zensu
```

Metode remove

menghapus objek pertama dari List.

```
list.remove(obj)
```

obj ialah parameter yang berisi elemen yang akan di hapus dari sebuah List.

```
#!/usr/bin/python

ini_list = [123, 'xyz', 'zensu', 'abc', 'xyz'];

ini_list.remove('xyz');
print "List : ", ini_list;
ini_list.remove('abc');
print "List : ", ini_list;
```

outputnya :

```
List : [123, 'zensu', 'abc', 'xyz']
List : [123, 'zensu', 'xyz']
```

Metode reverse

membalikkan objek List di tempat.

```
list.reverse()
```

contoh dalam menggunakan metode reverse :

```
#!/usr/bin/python

ini_list = [123, 'xyz', 'zensu', 'abc', 'xyz'];

ini_list.reverse();
print "List : ", ini_list;
```

outputnya :

```
List : ['xyz', 'abc', 'zensu', 'xyz', 123]
```

Metode sort

Mengurutkan objek dari List.

```
list.sort()
```

contoh dalam menggunakan metode sort untuk mengurutkan data (elemen-elemen) di dalam sebuah List :

```
#!/usr/bin/python  
  
ini_list = [123, 'xyz', 'zensu', 'abc', 'xyz'];  
  
ini_list.sort();  
print "List : ", ini_list;
```

outputnya :

```
List : [123, 'abc', 'xyz', 'xyz', 'zensu']
```

Tuple

Sumber :

http://www.tutorialspoint.com/python/python_tuples.htm

Pendahuluan

Tuple adalah urutan objek (Sequence Object) pada Python yang memiliki sifat konstan atau tidak berubah (Immutable) . Tuple bagian dari Sequence (urutan) dimana ia memiliki kesamaan dengan List namun perbedaannya nilai pada Tuple bersifat konstan atau tetap. Tuple di deklarasikan menggunakan tanda kurung biasa ().

```
tup1 = ('Python', 'Java', 1997, 2000)
tup2 = (1, 2, 3, 4, 5 )
tup3 = "a", "b", "c", "d"
```

Tuple kosong ditulis sebagai dua tanda kurung yang tidak berisi nilai apapun.

```
tup1 = ()
```

Untuk menulis Tuple mengandung nilai tunggal, Anda harus menyertakan tanda koma, meskipun hanya ada satu nilai.

```
tup1 = (50,)
```

Mengakses Nilai di dalam Tuple

Untuk mengakses nilai di dalam Tuple, gunakan tanda kurung siku dengan memberikan nilai indeks sesuai dengan elemen yang dipilih.

```
tup1 = ('Python', 'Java', 1997, 2000)
tup2 = (1, 2, 3, 4, 5 )

print "tup1[0]: ", tup1[0]
print "tup2[1:5]: ", tup2[1:5]
```

outputnya :

```
tup1[0]: Python
tup2[1:5]: (2, 3, 4, 5)
```

Mengupdate Tuple

Tuple bersifat immutable (nilai konstan). Anda tidak dapat memperbarui atau mengubah nilai-nilai elemen pada Tuple. Teknik lain adalah dengan mengambil bagian dari Tuple yang ada untuk menciptakan Tuple baru seperti contoh berikut :

```
#!/usr/bin/python

tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');

# statement ini tidak berlaku untuk Tuple
# tup1[0] = 100;

# membuat Tuple baru sebagai berikut
tup3 = tup1 + tup2;
print tup3
```

outputnya :

```
(12, 34.56, 'abc', 'xyz')
```

Menghapus Tuple

Menghapus elemen-elemen secara individu (satu per satu) dari Tuple itu tidaklah mungkin karena sifat Tuple itu Immutable (Tetap Nilainya). Ada cara lain untuk melakukan hal ini yaitu dengan menggunakan kata kunci del di ikuti dengan nama Tuplenya.

```
tup = ('Python', 'Java', 1997, 2000);

print tup;

del tup;

print "Setelah Menghapus tup : "

print tup;
```

outputnya :

```
Traceback (most recent call last):
  File "D:\codes\php\latihan\testing.py", line 9, in <module>
    print tup;
NameError: name 'tup' is not defined
('Python', 'Java', 1997, 2000)
Setelah Menghapus tup :
```

Operasi Dasar Tuple

Tuple juga menanggapi operator + dan *. Operator + digunakan untuk menggabungkan dua Tuple, sedangkan operator * digunakan untuk merepeat (mengulang) nilai Tuple. Hasil operasi ini tetap kembali menjadi Nilai Tuple dan bukan sebuah String.

| Ekspresi Python | Hasil | Deskripsi |
|---|---|--|
| <code>len((1, 2, 3))</code> | 3 | Jumlah elemen-elemen di dalam Tuple. |
| <code>(1, 2, 3) + (4, 5, 6)</code> | <code>(1, 2, 3, 4, 5, 6)</code> | Menggabungkan tuple. |
| <code>('Hi!',) * 4</code> | <code>('Hi!', 'Hi!', 'Hi!', 'Hi!')</code> | Repeat (mengulang) nilai Tuple. |
| <code>3 in (1, 2, 3)</code> | True | Mengecek objek 3 apakah bagian dari member (nilai) Tuple. |
| <code>for x in (1, 2, 3): print x,</code> | 1 2 3 | Iterasi dimana perulangan membuat X mengakses nilai seluruh Tuple sebanyak jumlah elemen pada Tuple. |

Pengindeksan dan Pemotongan

Karena Tuple merupakan Sequence (Urutan), pengindeksan dan pengirisan bekerja dengan cara yang sama seperti yang mereka lakukan untuk string.

```
T = ('spam', 'Spam', 'SPAM!')
```

| Ekspresi Python | Hasil | Deskripsi |
|--------------------|--------------------------------|--|
| <code>L[2]</code> | <code>'SPAM!'</code> | Mengambil nilai pada elemen berindeks dua. |
| <code>L[-2]</code> | <code>'Spam'</code> | Membaca Tuple dari kanan. Kanan awal -1 dan seterusnya. |
| <code>L[1:]</code> | <code>['Spam', 'SPAM!']</code> | Mengambil nilai-nilai Tuple mulai dari indeks ke 1 sampai indeks paling akhir. |

Fungsi Built-In Tuple

Python menyertakan fungsi-fungsi bawaan untuk tuple seperti berikut ini :

| Fungsi dengan Deskripsi |
|--|
| <code>cmp(tuple1, tuple2)</code> Membandingkan elemen dari dua Tuple. |
| <code>len(tuple)</code> Mengembalikan nilai berupa banyaknya jumlah elemen di dalam Tuple. |
| <code>max(tuple)</code> Mengembalikan elemen dari Tuple yang bernilai maksimal. |
| <code>min(tuple)</code> Mengembalikan elemen dari Tuple yang bernilai minimum. |
| <code>tuple(seq)</code> Mengkonversi List ke Tuple. |

Fungsi cmp

Sintaks :

```
cmp(tuple1, tuple2)
```

Parameter :

tuple1 → Tuple pertama yang akan dibandingkan dengan Tuple ke dua.

tuple2 → Tuple kedua sebagai pembanding dengan Tuple ke pertama.

```
#!/usr/bin/python
```

```
tuple1, tuple2 = (123, 'xyz'), (456, 'abc')
```

```
print cmp(tuple1, tuple2);  
print cmp(tuple2, tuple1);  
tuple3 = tuple2 + (786,);  
print cmp(tuple2, tuple3)
```

outputnya :

```
-1  
1  
-1
```

Fungsi len

Sintaks :

```
len(tuple)
```

Parameter :

tuple → Tuple yang jumlah elemennya akan dihitung.

Nilai kembalinya berupa objek integer jumlah dari elemen-elemen yang ada di Tuple tersebut.

```
#!/usr/bin/python  
  
tuple1, tuple2 = (123, 'xyz'), (456, 'abc')  
  
print "Panjang Tuple pertama : ", len(tuple1);  
print "Panjang Tuple kedua : ", len(tuple2);
```

outputnya :

```
Panjang Tuple pertama : 2  
Panjang Tuple kedua : 2
```

Fungsi max

Sintax :

```
max(tuple)
```

Parameter :

tuple → Tuple yang akan di cari nilai maksimalnya.

Nilai kembalinya berupa nilai Tuple yang paling besar dari elemen lainnya.

```
#!/usr/bin/python  
  
tuple1, tuple2 = (123, 'xyz', 'zensu', 'abc'), (456, 700, 200)  
  
print "Nilai Maksimum Elemen Tuple 1 : ", max(tuple1);  
print "Nilai Maksimum Elemen Tuple 2 : ", max(tuple2);
```

outputnya :

```
Nilai Maksimum Elemen Tuple 1 : zensu
Nilai Maksimum Elemen Tuple 2 : 700
```

Fungsi min

Sintax :

```
min(tuple)
```

Parameter :

tuple → Tuple yang akan di cari nilai minimumnya.

Nilai kembalinya berupa nilai Tuple yang paling kecil dari elemen lainnya.

```
#!/usr/bin/python

tuple1, tuple2 = (123, 'xyz', 'zensu', 'abc'), (456, 700, 200)

print "Nilai Minimum Elemen Tuple 1 : ", min(tuple1);
print "Nilai Minimum Elemen Tuple 2 : ", min(tuple2);
```

outputnya :

```
Nilai Minimum Elemen Tuple 1 : 123
Nilai Minimum Elemen Tuple 2 : 200
```

Fungsi tuple

Sintaks :

```
tuple(seq)
```

Parameter :

seq → sebuah sequence berupa List.

Nilai kembalinya adalah berupa Tuple, dimana dari List di konversi ke Tuple.

```
#!/usr/bin/python

ini_list = (123, 'xyz', 'zensu', 'abc');
ini_tuple = tuple(ini_list)

print "Elemen-elemen Tuple : ", ini_tuple
```

outputnya :

Elemen-elemen Tuple : (123, 'xyz', 'zensu', 'abc')

Dictionary

Sumber :

http://www.tutorialspoint.com/python/python_dictionary.htm

Pendahuluan

Dictionary bersifat mutable dimana nilainya bisa di ubah. Dictionary menyimpan sejumlah objek Python (seperti string, angka dll). Dictionary memiliki sepasang key (kunci) dan value (nilai) yang saling berhubungan.

Dictionary juga akrab dikenal sebagai Array Asosiatif atau hash. Sintak umumnya sebagai berikut :

```
dict = {'Junox': '2341', 'Yanzen': '9102', 'Tricko': '3258'}
```

Anda juga dapat membuat Dictionary dengan cara seperti ini :

```
dict1 = { 'abc': 456 }  
dict2 = { 'abc': 123, 98.6 : 37 }
```

Setiap key dan value selalu dipisahkan dengan titik dua (:), key dan value merupakan item, setiap item dipisahkan dengan koma, dan semuanya tertutup dalam kurung kurawal. Untuk Dictionary kosong ditulis hanya dengan dua kurung kurawal, seperti ini: {}.

Mengakses Nilai di Dictionary

Untuk mengakses elemen (nilai) pada Dictionary, Anda dapat menggunakan tanda kurung siku bersama dengan key (kunci) untuk memperoleh nilainya. Berikut ini adalah contohnya :

```
#!/usr/bin/python  
  
dict_mhs = {'Nama': 'Zensu', 'Umur': 21, 'Jurusan': 'TI'}  
  
print "dict_mhs['Nama']      : ", dict_mhs['Nama']  
print "dict_mhs['Umur']      : ", dict_mhs['Umur']
```

outputnya :

```
dict_mhs['Nama']      : Zensu  
dict_mhs['Umur']      : 21
```

Mengupdate Dictionary

Anda dapat mengupdate Dictionary dengan menambahkan entri baru atau item baru (yaitu, sepasang Key-Value), memodifikasi item yang sudah ada, atau menghapus item yang ada seperti ditunjukkan di bawah ini :

```
#!/usr/bin/python

dict_mhs = {'Nama': 'Zensu', 'Umur': 21, 'Jurusan': 'TI'}

# Mengupdate Umur
dict_mhs['Umur'] = 22;

# Menambahkan Item baru
# dengan Key = sekolah dan
# Value = zenuszen@mail.com"
dict_mhs['Email'] = "zenuszen@mail.com";

print "dict_mhs['Umur']    : ", dict_mhs['Umur'];
print "dict_mhs['Email']  : ", dict_mhs['Email'];
```

outputnya :

```
dict_mhs['Umur']    : 22
dict_mhs['Email']   : zenuszen@mail.com
```

Menghapus Elemen-Element Dictionary

Anda dapat menghapus elemen (item) Dictionary per item atau bisa juga menghapus keseluruhan item di dalam Dictionary dan Anda juga bisa menghapus seluruh kamus dalam satu operasi saja.

Secara jelas untuk menghapus seluruh item di Dictionary, cukup gunakan pernyataan del. Berikut adalah contohnya :

```
#!/usr/bin/python

dict_mhs = {'Nama': 'Zensu', 'Umur': 21, 'Jurusan': 'TI'};

del dict_mhs['Nama']; # menghapus item dengan kunci 'Nama'
print dict_mhs

dict_mhs.clear();    # menghapus semua item di dalam dict_mhs
print dict_mhs

del dict_mhs ;       # menghapus Dictionary
print dict_mhs
```

outputnya :

```
{'Jurusan': 'TI', 'Umur': 21}
{}

Traceback (most recent call last):
  File "D:\codes\php\latihan\testing.py", line 12, in <module>
    print dict_mhs
NameError: name 'dict_mhs' is not defined
```

error terjadi untuk membuktikan apakah dictionary dict_mhs telah di hapus.

Key Dictionary

Nilai-nilai (Value) di dalam Dictionary tidak terbatas. Nilai-nilai ini bersifat bebas artinya Objeknya sembarang, baik objek standar atau objek yang ditetapkan si programmernya. ini tidak berlaku pada Kunci (Key).

ada dua hal penting untuk selalu Anda ingat tentang Key Dictionary :

1. Tidak boleh memiliki key duplikat atau key yang sama dalam satu Dictionary yang sama. Jika masih terdapat dua Key duplikat atau sama, maka saat mengakses value dengan key tersebut, key terakhir lah yang akan di ambil valuenya.

```
#!/usr/bin/python

dict_mhs = {'Nama': 'Zensu', 'Umur': 21, 'Jurusan': 'TI', 'Nama': 'Yanzen'};

print "dict_mhs['Nama']: ", dict_mhs['Nama'];
```

outputnya :

```
dict_mhs['Nama']: Yanzen
```

2. Kunci bersifat Immutable (tetap/konstan). Ini artinya Anda hanya dapat menggunakan string, numerik (angka) atau Tuple sebagai key pada Dictionary tersebut. Tidak diperbolehkan menggunakan List (karena sifat List Mutable / dapat berubah). Berikut ini adalah contohnya :

```
#!/usr/bin/python

dict_mhs = {['Nama']: 'Zensu', 'Umur': 21, 'Jurusan': 'TI', 'Nama': 'Yanzen'};

print "dict_mhs['Nama']: ", dict_mhs['Nama'];
```

outputnya :

```
Traceback (most recent call last):
  File "D:\codes\php\latihan\testing.py", line 3, in <module>
    dict_mhs = {['Nama']: 'Zensu', 'Umur': 21, 'Jurusan': 'TI', 'Nama': 'Yanzen'};
TypeError: unhashable type: 'list'
```


Fungsi Built-In Dictionary

Python menyediakan fungsi-fungsi untuk Dictionary sebagai berikut :

| Fungsi | Deskripsi |
|--------------------------------|---|
| <code>cmp(dict1, dict2)</code> | Membandingkan item dari kedua Dictionary. |
| <code>len(dict)</code> | Menghitung jumlah item di dalam Dictionary. |
| <code>str(dict)</code> | Representasi string dari sebuah Dictionary. |
| <code>type(variable)</code> | Mengecek jenis variabel, jika variabel adalah Dictionary maka nilai kembalinya ke Dictionary. |

Fungsi cmp

Sintaks :

```
cmp(dict1, dict2)
```

Parameter :

1. `dict1` → ini adalah Dictionary pertama yang akan dibandingkan dengan Dictionary kedua.
2. `dict2` → ini adalah Dictionary kedua sebagai perbandingan dari Dictionary pertama.

Fungsi ini mengembalikan nilai **0** jika kedua Dictionary sama, **-1** jika Dictionary pertama lebih kecil dari Dictionary kedua dan **1** jika Dictionary pertama lebih dari Dictionary kedua.

```
#!/usr/bin/python

dict1 = {'Nama': 'Zensu', 'Umur': 7};
dict2 = {'Nama': 'Tricko', 'Umur': 27};
dict3 = {'Nama': 'Yanzen', 'Umur': 27};
dict4 = {'Nama': 'Vad', 'Umur': 7};

print "Nilai Kembali : %d" % cmp (dict1, dict2)
print "Nilai Kembali : %d" % cmp (dict2, dict3)
print "Nilai Kembali : %d" % cmp (dict1, dict4)
```

outputnya :

```
Nilai Kembali : 1
Nilai Kembali : -1
Nilai Kembali : 1
```

Fungsi len

Sintaks :

```
len(dict)
```

Parameter :

dict → Dictionary yang akan dihitung jumlah itemnya.

```
#!/usr/bin/python
dict_mhs = {'Nama': 'Zensu', 'Umur': 21, 'Jurusan': 'TI'}
print "Panjang dict_mhs : %d" % len(dict_mhs)
```

outputnya :

```
Panjang dict_mhs : 3
```

Fungsi str

Sintaks :

```
str(dict)
```

Parameter :

dict → Dictionary.

```
#!/usr/bin/python
dict_mhs = {'Nama': 'Zensu', 'Umur': 21, 'Jurusan': 'TI'}
print "Equivalent String : %s" % str (dict_mhs)
```

outputnya :

```
Equivalent String : {'Nama': 'Zensu', 'Jurusan': 'TI', 'Umur': 21}
```

Fungsi type

Sintaks :

```
type(dict)
```

Parameter :

dict → Dictionary.

```
#!/usr/bin/python
dict_mhs = {'Nama': 'Zensu', 'Umur': 21, 'Jurusan': 'TI'}
print "Type Variabel : %s" % type (dict_mhs)
```

outputnya :

```
Type Variabel : <type 'dict'>
```

Method Built-In Dictionary

Python menyediakan sejumlah method untuk Dictionary sebagai berikut :

| Method | Deskripsi |
|--|---|
| <code>dict.clear()</code> | Menghapus semua elemen pada Dictionary dict. |
| <code>dict.copy()</code> | Mengembalikan salinan dari Dictionary dict. |
| <code>dict.fromkeys()</code> | Membuat Dictionary baru dengan key dari sebuah urutan lalu memasukan nilai yang ditentukan dalam sebuah parameter 'value'. |
| <code>dict.get(key, default=None)</code> | Mengembalikan value (nilai) dari sebuah key atau default jika kunci tidak ada dalam Dictionary. |
| <code>dict.has_key(key)</code> | Mengembalikan nilai True jika key ada di dalam Dictionary. Mengembalikan nilai False jika key tidak ada di dalam Dictionary. |
| <code>dict.items()</code> | Membuat item (keys-values) menjadi Tuple. |
| <code>dict.keys()</code> | Mengembalikan / Mendapatkan semua key di dalam Dictionary dalam bentuk List. |
| <code>dict.update(dict2)</code> | Menambahkan item-item yang ada di dict2 ke dict. |
| <code>dict.values()</code> | Mendapatkan value dari Dictionary dict dalam bentuk List. |

Method clear

Sintaks :

```
dict.clear()

#!/usr/bin/python

dict = {'Nama': 'Zensu', 'Umur': 21};

print "Start Len : %d" % len(dict)

dict.clear()

print "End Len : %d" % len(dict)
```

outputnya :

```
Start Len    : 2
End Len      : 0
```

Method copy

Sintaks :

```
dict.copy()

#!/usr/bin/python

dict1 = {'Nama': 'Zensu', 'Umur': 21};

dict2 = dict1.copy()
print "Dictionary yang baru : %s" % str(dict2)
```

outputnya :

```
Dictionary yang baru : {'Nama': 'Zensu', 'Umur': 21}
```

Method fromkeys

Sintaks :

```
dict.fromkeys(seq[, value])
```

Parameter :

seq → nilai List yang akan digunakan untuk menyusun key Dictionary.
value → jika parameter ini tersedia maka value akan diatur dengan isi parameter ini.

```
#!/usr/bin/python

key_mhs = ('nama', 'umur', 'jenkel')

dict = dict.fromkeys(key_mhs)
print "Dictionary baru : %s" % str(dict)

dict = dict.fromkeys(key_mhs, 10)
print "Dictionary baru : %s" % str(dict)
```

outputnya :

```
Dictionary baru : {'nama': None, 'jenkel': None, 'umur': None}
Dictionary baru : {'nama': 10, 'jenkel': 10, 'umur': 10}
```

Method get

Sintaks :

```
dict.get(key, default=None)
```

Parameter :

key → key yang akan dicari dalam Dictionary.

default → jika Key tidak ada di Dictionary, nilai parameter ini yang akan menggantikan.

```
#!/usr/bin/python

dict = {'Nama': 'Zensu', 'Umur': 21};

print "Nilai : %s" % dict.get('Age')
print "Nilai : %s" % dict.get('Sex', "Never")
```

outputnya :

```
Nilai : 21
Nilai : Never
```

Method has_key

Sintaks :

```
dict.has_key(key)
```

Parameter :

key → key yang akan dicari dalam Dictionary.

```
#!/usr/bin/python

dict = {'Nama': 'Zensu', 'Umur': 21};
print "Value : %s" % dict.has_key('Umur')
print "Value : %s" % dict.has_key('Jenkel')
```

outputnya :

```
Value : True
Value : False
```

Method value

Sintaks :

```
dict.value()
```

Parameter :

```
#!/usr/bin/python  
  
dict = {'Nama': 'Zensu', 'Umur': 21};  
  
print "Value : %s" % dict.values()
```

outputnya :

```
Value : ['Zensu', 21]
```

Library Standar

Sumber :

http://sakti.github.io/python101/standard_library.html

Module *getpass*

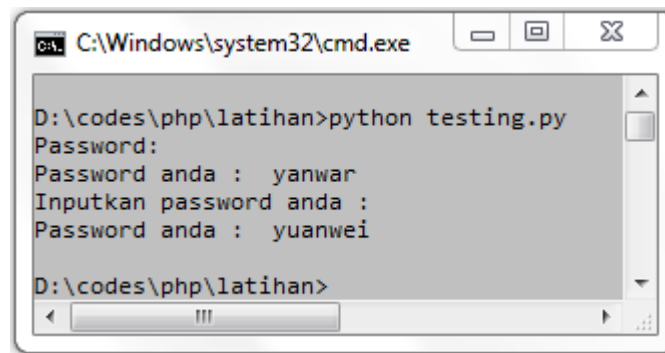
Mendapatkan password pengguna tanpa **echo** kembali ke pengguna.

```
import getpass

password = getpass.getpass()
print 'Password anda : ', password

password = getpass.getpass(prompt='Inputkan password anda :')
print 'Password anda : ', password
```

outputnya :



Modul *random*

Modul **random** menyediakan **fast pseudorandom number generator** berdasarkan algoritma **Mersenne Twister**.

```
import random

print 'bilangan random antara 0<= n < 1.0 : ', random.random()
print 'bilangan random antara 0<= n < 1.0 : ', random.random()
print 'bilangan random antara 0<= n < 1.0 : ', random.random()

# random integer
print 'bilangan random antara 1<= n <= 100 : ', random.randint(1, 100)
print 'bilangan random antara 1<= n <= 100 : ', random.randint(1, 100)
print 'bilangan random antara 1<= n <= 100 : ', random.randint(1, 100)
```

outputnya :

```
bilangan random antara 0<= n < 1.0 : 0.0273700885598
bilangan random antara 0<= n < 1.0 : 0.313057807001
bilangan random antara 0<= n < 1.0 : 0.389464341689
bilangan random antara 1<= n <= 100 : 35
bilangan random antara 1<= n <= 100 : 12
bilangan random antara 1<= n <= 100 : 72
```

Modul datetime

Modul **datetime** berisi fungsi dan class untuk operasi tanggal dan waktu.

```
# contoh penggunaan module datetime

import datetime
import time

sekarang = datetime.datetime.now()

tanggal = sekarang.date()
waktu = sekarang.time()

print 'Hari : ', tanggal.day
print 'Bulan : ', tanggal.month
print 'Tahun : ', tanggal.year
print 'Jam : ', waktu.hour
print 'Menit : ', waktu.minute
print 'Detik : ', waktu.second

time.sleep(5)

sekarang2 = datetime.datetime.now()

delta = sekarang2 - sekarang

print 'selisih detik : ', delta.total_seconds()
```

outputnya :

```
Hari : 3
Bulan : 9
Tahun : 2014
Jam : 20
Menit : 32
Detik : 43
selisih detik : 5.001
```

Modul math

Modul **math** berisi fungsi-fungsi matematika.

```
# contoh penggunaan modul math

import math

# konstanta
print 'pi = ', math.pi
print 'e = ', math.e

# faktorial, n!
for i in range(1, 11):
    print '%s! = %s' % (i, math.factorial(i))

# pangkat
print '2 pangkat 12 = ', math.pow(2, 12)

# akar kuadrat
print 'akar kuadrat 10 = ', math.sqrt(10)

# logaritma
print 'log 8 = ', math.log(8)
print 'log 8 basis 10 = ', math.log(8, 10)
print 'log 8 basis 10 = ', math.log10(8)

# trigonometri
print 'sin 90 derajat = ', math.sin(math.radians(90))
```

outputnya :

```
pi = 3.14159265359
e = 2.71828182846
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
2 pangkat 12 = 4096.0
akar kuadrat 10 = 3.16227766017
log 8 = 2.07944154168
log 8 basis 10 = 0.903089986992
log 8 basis 10 = 0.903089986992
sin 90 derajat = 1.0
```

Modul sys

Modul **sys** digunakan untuk mengakses konfigurasi interpreter pada saat runtime dan berinteraksi dengan **environment** sistem operasi.

```
# contoh penggunaan modul sys / System-specific Configuration

import sys

# argumen terminal
print sys.argv
# versi python
print 'versi python: ', sys.version
# platform
print 'platform : ', sys.platform
# letak python interpreter
print 'executable : ', sys.executable
# byteorder
print 'byteorder : ', sys.byteorder

# module yang diimport
print 'modul yang diimport : ', sys.modules
# module built-in
print 'modul built-in : ', sys.builtin_module_names

# path import
print 'path import : ', sys.path
```

outputnya :

```
['D:\\codes\\php\\latihan\\testing.py']
versi python: 2.7.8 (default, Jul 2 2014, 19:50:44) [MSC v.1500 32 bit (Intel)]
platform : win32
executable : C:\\Python27\\python.exe
byteorder : little
modul yang diimport : {'copy_reg': <module 'copy_reg' from
'C:\\Python27\\lib\\copy_reg.pyc'>, 'sre_compile': <module 'sre_compile' from
'C:\\Python27\\lib\\sre_compile.pyc'>, 'locale': <module 'locale' from
'C:\\Python27\\lib\\locale.pyc'>, '_sre': <module '_sre' (built-in)>, 'functools':
<module 'functools' from 'C:\\Python27\\lib\\functools.pyc'>, 'encodings': <module
'encodings' from 'C:\\Python27\\lib\\encodings\\__init__.pyc'>, 'site': <module 'site'
from 'C:\\Python27\\lib\\site.pyc'>, '__builtin__': <module '__builtin__' (built-in)>,
'sysconfig': <module 'sysconfig' from 'C:\\Python27\\lib\\sysconfig.pyc'>, 'operator':
<module 'operator' (built-in)>, '__main__': <module '__main__' from
'D:\\codes\\php\\latihan\\testing.py'>, 'types': <module 'types' from
'C:\\Python27\\lib\\types.pyc'>, 'encodings.encodings': None, 'msvcrt': <module
'msvcrt' (built-in)>, 'abc': <module 'abc' from 'C:\\Python27\\lib\\abc.pyc'>,
'_weakrefset': <module '_weakrefset' from 'C:\\Python27\\lib\\_weakrefset.pyc'>,
'errno': <module 'errno' (built-in)>, 'encodings.codecs': None, 'sre_constants':
<module 'sre_constants' from 'C:\\Python27\\lib\\sre_constants.pyc'>, 're': <module
're' from 'C:\\Python27\\lib\\re.pyc'>, '_abcoll': <module '_abcoll' from
'C:\\Python27\\lib\\_abcoll.pyc'>, 'ntpath': <module 'ntpath' from
'C:\\Python27\\lib\\ntpath.pyc'>, '_codecs': <module '_codecs' (built-in)>, 'nt':
```

```

<module 'nt' (built-in)>, '_warnings': <module '_warnings' (built-in)>,
'genericpath': <module 'genericpath' from 'C:\Python27\lib\genericpath.pyc'>,
'stat': <module 'stat' from 'C:\Python27\lib\stat.pyc'>, 'zipimport': <module
'zipimport' (built-in)>, 'encodings.__builtin__': None, 'warnings': <module
'warnings' from 'C:\Python27\lib\warnings.pyc'>, 'UserDict': <module 'UserDict' from
'C:\Python27\lib\UserDict.pyc'>, 'encodings.utf_8': <module 'encodings.utf_8' from
'C:\Python27\lib\encodings\utf_8.pyc'>, 'encodings.cp1252': <module
'encodings.cp1252' from 'C:\Python27\lib\encodings\cp1252.pyc'>, 'sys': <module
'sys' (built-in)>, 'codecs': <module 'codecs' from 'C:\Python27\lib\codecs.pyc'>,
'os.path': <module 'ntpath' from 'C:\Python27\lib\ntpath.pyc'>, '_functools':
<module '_functools' (built-in)>, 'getpass': <module 'getpass' from
'C:\Python27\lib\getpass.pyc'>, '_locale': <module '_locale' (built-in)>,
'sitecustomize': <module 'sitecustomize' from
'C:\Users\yanwar\AppData\Roaming\AppDataLocal\Aptana
Studio\plugins\org.python.pydev_3.0.0.1388187472\pysrc\pydev_sitecustomize\sitecusto
mize.pyc'>, 'signal': <module 'signal' (built-in)>, 'traceback': <module 'traceback'
from 'C:\Python27\lib\traceback.pyc'>, 'linecache': <module 'linecache' from
'C:\Python27\lib\linecache.pyc'>, 'encodings.aliases': <module 'encodings.aliases'
from 'C:\Python27\lib\encodings\aliases.pyc'>, 'exceptions': <module 'exceptions'
(built-in)>, 'sre_parse': <module 'sre_parse' from 'C:\Python27\lib\sre_parse.pyc'>,
'os': <module 'os' from 'C:\Python27\lib\os.pyc'>, '_weakref': <module '_weakref'
(built-in)>
modul built-in : ('__builtin__', '__main__', '_ast', '_bisect', '_codecs',
'_codecs_cn', '_codecs_hk', '_codecs_iso2022', '_codecs_jp', '_codecs_kr',
'_codecs_tw', '_collections', '_csv', '_functools', '_heapq', '_hotshot', '_io',
'_json', '_locale', '_lsprof', '_md5', '_multibytecodec', '_random', '_sha',
'_sha256', '_sha512', '_sre', '_struct', '_subprocess', '_symtable', '_warnings',
'_weakref', '_winreg', 'array', 'audioop', 'binascii', 'cPickle', 'cStringIO',
'cmath', 'datetime', 'errno', 'exceptions', 'future_builtins', 'gc', 'imageop',
'imp', 'itertools', 'marshal', 'math', 'mmap', 'msvcrt', 'nt', 'operator', 'parser',
'signal', 'strop', 'sys', 'thread', 'time', 'xxsubtype', 'zipimport', 'zlib')
path import : ['D:\\codes\\php\\latihan', 'C:\\Python27\\DLLs',
'C:\\Python27\\lib', 'C:\\Python27\\lib\\lib-tk', 'C:\\Python27',
'C:\\Users\\yanwar\\AppData\\Roaming\\Python\\Python27\\site-packages',
'C:\\Python27\\lib\\site-packages', 'C:\\Python27\\lib\\site-packages\\win32',
'C:\\Python27\\lib\\site-packages\\win32\\lib', 'C:\\Python27\\lib\\site-
packages\\Pythonwin', 'C:\\Users\\yanwar\\AppData\\Roaming\\Python\\Python27\\site-
packages\\PIL', 'C:\\Users\\yanwar\\AppData\\Roaming\\Python\\Python27\\site-
packages\\wx-2.8-msw-unicode', 'C:\\Windows\\system32\\python27.zip',
'C:\\Python27\\lib\\plat-win']

```

PYMOTW (Python Module of The Week)

Masih ada banyak modul yang ada di Python. Untuk menjelajahi modul-modul yang tersedia di Python anda dapat membaca **Python Module of The Week** yang membahas modul python satu per satu.

Python Module of The Week : <http://pymotw.com/2/>

Fungsi (Function)

Sumber :

<http://sakti.github.io/python101/fungsi.html>

Pendahuluan

Fungsi adalah bagian dari program yang dapat digunakan ulang. Hal ini bisa dicapai dengan memberi nama pada blok statemen, kemudian nama ini dapat dipanggil di manapun dalam program. Kita telah menggunakan beberapa fungsi builtin seperti **range**.

Fungsi dalam Python didefinisikan menggunakan kata kunci **def**. Setelah **def** ada nama pengenalan fungsi diikuti dengan parameter yang diapit oleh tanda kurung dan diakhiri dengan tanda titik dua (:). Baris berikutnya berupa blok fungsi yang akan dijalankan jika fungsi dipanggil.

```
# lat15.py

def halo_dunia():
    print 'Halo Dunia!'

halo_dunia() # memanggil fungsi halo_dunia
halo_dunia() # fungsi halo_dunia dipanggil lagi
```

outputnya :

```
Halo Dunia!
Halo Dunia!
```

Parameter Fungsi

Fungsi dapat membaca parameter, parameter adalah nilai yang disediakan kepada fungsi, dimana nilai ini akan menentukan **output** yang akan dihasilkan fungsi.

Parameter dikirim dalam tanda kurung saat pemanggilan fungsi. Nilai parameter saat pemanggilan fungsi dinamakan *argument*.

```
# lat16.py

def halo(nama):
    print 'Halo %s!' % nama

def cetak_maksimal(a, b):
    if a > b:
        print '%s merupakan nilai maksimal' % a
    elif a == b:
        print '%s sama dengan %s' % (a, b)
    else:
        print '%s merupakan nilai maksimal' % b

# memanggil fungsi halo dengan argumen 'Dunia'
halo('Dunia')

# memanggil fungsi halo dengan argumen 'Indonesia'
halo('Indonesia')

cetak_maksimal(10, 100)
```

```
x = 9
y = 3

cetak_maksimal(x, y)
```

outputnya :

```
Halo Dunia!
Halo Indonesia!
100 merupakan nilai maksimal
9 merupakan nilai maksimal
```

Variabel Lokal

Jika ada variabel yang dideklarasikan didalam blok fungsi, variabel ini tidak ada kaitannya dengan variabel lain dengan nama yang sama diluar fungsi, dengan kata lain nama variabel hanya lokal untuk fungsi. Hal ini disebut juga **scope variabel**.

```
# lat17.py

x = 50

def fungsi(x):

    print 'x = ', x
    x = 2
    print 'merubah lokal variabel x = ', x

fungsi(100)

print 'nilai x masih %s' % x
```

outputnya :

```
x = 100
merubah lokal variabel x = 2
nilai x masih 50
```


Penggunaan Statemen Global

Dalam blok fungsi kita dapat mengakses variabel diluar fungsi, akses ini terbatas hanya akses baca. Jika blok fungsi ingin menulis variabel diluar fungsi anda dapat menggunakan statemen global.

```
# lat17.py

x = 50

def fungsi():
    print 'x = ', x

def fungsi2():
    x = 100 # menulis ke lokal variabel
    print 'x = ', x

def fungsi3():
    global x
    x = 100
    print 'x = ', x

fungsi()
print 'nilai x = ', x

fungsi2()
print 'nilai x = ', x

fungsi3()
print 'nilai x = ', x
```

outputnya :

```
x = 50
nilai x = 50
x = 100
nilai x = 50
x = 100
nilai x = 100
```

Nilai Argumen Default

Untuk beberapa fungsi yang ingin menyediakan paramater opsional dan menggunakan nilai default jika pengguna tidak menyediakan argumen saat fungsi dipanggil. Anda bisa menspesifikasikan nilai default dengan tanda sama dengan (=) setelah nama parameter.

```
# lat18.py

def katakan(pesan, jumlah=1):
    print pesan * jumlah

katakan('Halo ')
katakan('Halo ', 3)
```

outputnya :

```
Halo
Halo Halo Halo
```

Keyword Argumen

Jika anda membuat fungsi dengan banyak parameter dan anda hanya ingin menspesifikasikan sebagian, anda dapat menggunakan keyword argumen. Kita menggunakan nama (keyword) melainkan posisi (argumen posisi, normal pemanggilan).

```
# lat19.py

def fungsi(a, b=5, c=10):
    print 'a = ', a
    print 'b = ', b
    print 'c = ', c

fungsi(3, 7)
fungsi(25, c=24)
fungsi(c=50, a=100)
```

outputnya :

```
a = 3
b = 7
c = 10
a = 25
b = 5
c = 24
a = 100
b = 5
c = 50
```

Parameter VarArgs

Terkadang anda ingin membuat fungsi yang dapat menerima jumlah argumen yang tidak tentu, hal ini dapat dilakukan menggunakan tanda bintang (*).

```
# lat20.py

def total(*bilangan, **keywords):
    hitung = 0
    for bil in bilangan:
        hitung += bil
    for key in keywords:
        hitung += keywords[key]
    return hitung

print total(1, 2, 3, 4, 5)
print total(daging=2, sayur=10, buah=3)
print total(7, 8, 5, daging=2, sayur=10, buah=3)
```

outputnya :

```
15
15
35
```

Statemen Return

Statemen return digunakan untuk keluar dari fungsi. Kita juga dapat menspesifikasikan nilai kembalian. Seperti pada latihan 20 melainkan mencetak hasil jumlah dalam blok fungsi, fungsi total mengembalikan nilai jumlah ke pemanggil. Doc String
Python memiliki fitur documentation string, seringnya disebut dengan nama docstring. Docstring berguna untuk mendokumentasikan program agar mudah untuk dipahami dan digunakan.

```
# lat21.py

def katakan(pesan, jumlah=1):
    "mencetak pesan <pesan> dengan jumlah <jumlah>"
    print pesan * jumlah

print katakan.__doc__
```

outputnya :

```
mencetak pesan <pesan> dengan jumlah <jumlah>
```

Secara interaktif anda dapat mengakses docstring dengan fungsi help.d

```
>>> import lat21
>>> help(lat21.katakan)
```

Doc String

Python memiliki fitur documentation string, seringnya disebut dengan nama docstring. Docstring berguna untuk mendokumentasikan program agar mudah untuk dipahami dan digunakan.

```
# lat21.py

def katakan(pesan, jumlah=1):
    "mencetak pesan <pesan> dengan jumlah <jumlah>"
    print pesan * jumlah

print katakan.__doc__
```

outputnya :

```
mencetak pesan <pesan> dengan jumlah <jumlah>
```

Secara interaktif anda dapat mengakses docstring dengan fungsi help.

```
>>> import lat21
>>> help(lat21.katakan)
```

Modul

Sumber :

<http://sakti.github.io/python101/modul.html>

Pendahuluan

lat21.py

```
# lat21.py

def katakan(pesan, jumlah=1):
    "mencetak pesan <pesan> dengan jumlah <jumlah>"
    print pesan * jumlah
```

Anda dapat menggunakan kode ulang dalam program menggunakan fungsi. Bagaimana cara menggunakan fungsi yang ada di file .py yang berbeda? jawabnya adalah modul. File latihan yang sudah anda buat dari **lat1.py** sampai **lat21.py** merupakan modul. *Untuk menggunakan fungsi atau variabel yang ada di file tersebut kita dapat melakukan import.*

```
>>> import lat21
>>> print lat21.katakan('A', 10)
AAAAAAAAAA
```

Kita dapat membuat modul .py dengan bahasa pemrograman C.

Byte-compiled (file .pyc)

Setelah mencoba modul **lat21** (tanpa .py). Anda akan menemukan file **lat21.pyc** pada direktori yang sama.

Jika anda melakukan **import** suatu modul, modul tersebut akan di interpret terlebih dahulu. Untuk optimisasi python akan membuat file byte-compiled modul tersebut dalam file .pyc sehingga import modul tidak harus melakukan *compile*.

Statement from ... import

Anda dapat mengakses fungsi, variabel atau class dalam modul menggunakan berbagai cara.

```
# lat22.py

import lat21
from lat21 import katakan
from lat21 import katakan as hi

print lat21.katakan('a', 10)
print katakan('a', 20)
print hi('a', 30)
```

outputnya :

```
aaaaaaaaaa
```

```
aaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaa
```

Nama Modul

Setiap module memiliki nama, anda bisa mengakses nama ini menggunakan variabel `__name__`. Kita dapat tahu apakah modul ini dijalankan **standalone** atau di **import** oleh modul lain.

Jika modul kita dijalankan standalone maka isi variabel `__name__` berisi `__main__`.

```
# lat23.py

if __name__ == '__main__':
    # akan dijalankan jika dieksekusi secara langsung
    # bukan import
    print 'nama modul ini : ', __name__

    import lat21
    print 'nama modul yang di import : ', lat21.__name__
```

outputnya :

```
nama modul ini : __main__
nama modul yang di import : lat21
```

Fungsi dir

Untuk melihat isi dalam suatu modul kita dapat menggunakan fungsi builtin **dir**.

```
>>> import sys

>>> dir(sys)
['_displayhook_', '__doc__', '__egginsert', '__excepthook__', '__name__',
 '__package__', '__plen', '__stderr__', '__stdin__', '__stdout__',
 '_clear_type_cache', '_current_frames', '_getframe', '_mercurial', 'api_version',
 'argv', 'builtin_module_names', 'byteorder', 'call_tracing', 'callstats',
 'copyright', 'displayhook', 'dont_write_bytecode', 'exc_clear', 'exc_info',
 'exc_type', 'excepthook', 'exec_prefix', 'executable', 'exit', 'flags',
 'float_info', 'float_repr_style', 'getcheckinterval', 'getdefaultencoding',
 'getdlopenflags', 'getfilesystemencoding', 'getprofile', 'getrecursionlimit',
 'getrefcount', 'getsizeof', 'gettrace', 'hexversion', 'last_traceback', 'last_type',
 'last_value', 'long_info', 'maxint', 'maxsize', 'maxunicode', 'meta_path',
 'modules', 'path', 'path_hooks', 'path_importer_cache', 'platform', 'prefix', 'ps1',
 'ps2', 'py3kwarning', 'pydebug', 'setcheckinterval', 'setdlopenflags', 'setprofile',
 'setrecursionlimit', 'settrace', 'stderr', 'stdin', 'stdout', 'subversion',
 'version', 'version_info', 'warnoptions']
```

Package

Sekarang anda dapat mengamati struktur program Python. Variabel ada di dalam fungsi. Fungsi dan variabel global ada dalam modul. Bagaimana caranya mengorganisasikan modul? jawabannya adalah

Package.

Package adalah direktori yang berisi modul python dan file spesial `__init__.py`. File `__init__.py` menandakan bahwa direktori ini merupakan package Python.

Untuk latihan kali ini, kita buat direktori **lat24**. Direktor ini berisi :

```
latihan [DIR] /
|
|_ lat24 [DIR] /
|   * __init__.py      [FILE]
|   * kata.py          [FILE]
|   * nomor.py         [FILE]
|_ lat24tes.py [FILE]
```

`__init__.py`

```
# __init__.py
print "di jalankan ketika package di import"
```

`kata.py`

```
# kata.py
def balik_huruf(kata):
    return kata[::-1]
```

`nomor.py`

```
# nomor.py
def lebih_besar(a, b):
    if a > b:
        return a
    else:
        return b
```

Pada direktori latihan kita buat file **lat24tes.py**.

```
# lat24tes.py
from lat24 import kata
from lat24 import nomor
print kata.balik_huruf('selamat datang')
print nomor.lebih_besar(10, 18)
```

outputnya :

```
di jalankan ketika package di import
gnatad tamales
18
```


Input dan Output

Sumber :

<http://sakti.github.io/python101/io.html>

Pendahuluan

Akan ada situasi dimana program yang anda buat harus berinteraksi dengan pengguna. Sebagai contoh program anda ingin mendapatkan inputan pengguna kemudian mencetak hasil operasi program. Kita dapat melakukannya menggunakan fungsi **raw_input** dan statemen **print**.

Selain itu salah satu input/output yang umum yaitu operasi file. Kemampuan untuk membuat, membaca dan menulis file.

Input dari Pengguna

```
# lat36.py

def balik_string(teks):
    return teks[::-1]

def apakah_palindrom(teks):
    return teks == balik_string(teks)

inputan = raw_input('Masukkan teks: ')

if apakah_palindrom(inputan):
    print 'Ya, inputan berupa palindrom'
else:
    print 'Tidak, inputan bukan palindrom'
```

outputnya :

```
Masukkan teks: juan pablo zensu
Tidak, inputan bukan palindrom
```

File

Anda bisa membuka dan menggunakan file untuk membaca atau menulis dengan membuat **file** obyek.

```
# lat37.py

teks = """ini adalah isi dari file
yang akan ditulis
menggunakan python"""

# membuka dengan mode tulis
f = open('coba.txt', 'w')
f.write(teks)
f.close()

# default membuka file dengan mode baca
f = open('coba.txt')

while True:
    baris = f.readline()
```

```

        if len(baris) == 0:
            # EOF
            break
        print baris,

f.close()

```

outputnya :

```

ini adalah isi dari file
yang akan ditulis
menggunakan python

```

Pickle

Python menyediakan modul **pickle** untuk menyimpan obyek Python kedalam file dan membaca obyek Python dari file.

```

# lat38.py

import pickle

daftar_belanja_file = 'daftar.data'
daftar_belanja = ['apel', 'mangga', 'wortel', 'pisang']

# membuka file penyimpanan obyek dengan mode tulis binary
f = open(daftar_belanja_file, 'wb')

# dump obyek ke file
pickle.dump(daftar_belanja, f)
f.close()

# hapus daftar_belanja dari memori
del daftar_belanja

# membaca dari file
f = open(daftar_belanja_file, 'rb')
daftar_tersimpan = pickle.load(f)
print daftar_tersimpan

```

outputnya :

```

['apel', 'mangga', 'wortel', 'pisang']

```

Eksepsi

Sumber :

<http://sakti.github.io/python101/exception.html>

Pendahuluan

Eksepsi terjadi ketika ada sesuatu yang terduga muncul dalam program. Misalnya program anda akan membaca suatu file, namun file tersebut tidak ada. Hal seperti ini ditangani dengan **exception**.

Syntax Error

Syntax error, atau dikenal juga sebagai parsing error, adalah error ketika Python memparsing program anda.

```
>>> Print 'halo'
```

outputnya :

```
File "<stdin>", line 1
Print 'halo'
    ^
SyntaxError: invalid syntax
>>> while True print 'Hello world'
      File "<stdin>", line 1
        while True print 'Hello world'
            ^
SyntaxError: invalid syntax
```

Exception

Kita akan mencoba / **try** membaca input dari pengguna. Tekan **Ctrl-d** apa yang akan terjadi.

```
>>> teks = raw_input('Ketikkan sesuatu: ')
```

outputnya :

```
Ketikkan sesuatu: Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
EOFError
```

Python mengeluarkan eksepsi **EOFError** yang berarti menemukan simbol **end of file** (direpresentasikan oleh **Ctrl-d**) ketika program berharap tidak akan ada.

Penanganan Exception

Kita dapat menangani eksepsi menggunakan statemen **try ... except**. Sederhananya kita letakkan statemen yang mungkin mengeluarkan eksepsi kedalam **try-block**, dan letakan kode penanganan eksepsi kedalam **except-block**.

```
# lat39.py

try:
    teks = raw_input('Ketikkan sesuatu: ')
except EOFError:
    print '\nKenapa sudah EOF?'
except KeyboardInterrupt:
    print '\nAnda membatalkan operasi'
else:
    print 'Anda mengetikkan "%s"' % teks
```

outputnya :

```
Ketikkan sesuatu: zensu
Anda mengetikkan "zensu"
```

Mengeluarkan Exception

Anda dapat mengeluarkan eksepsi menggunakan statemen **raise** dengan menyediakan obyek eksepsi. Anda dapat membuat eksepsi sendiri dengan membuat class turunan **Exception**.

```
# lat40.py

class InputPendekError(Exception):
    "exception jika input terlalu pendek"

    def __init__(self, panjang, minimal):
        Exception.__init__(self)
        self.panjang = panjang
        self.minimal = minimal

try:
    teks = raw_input('Ketikkan sesuatu: ')
    panjang = len(teks)
    minimal_panjang = 3

    if panjang < minimal_panjang:
        raise InputPendekError(panjang, minimal_panjang)

except EOFError:
    print '\nKenapa sudah EOF?'
except KeyboardInterrupt:
    print '\nAnda membatalkan operasi'
except InputPendekError as e:
    print 'input terlalu pendek: panjang input: %s, minimal: %s' % (e.panjang,
```

```
e.minimal)
else:
    print 'Anda mengetikkan "%s"' % teks
```

outputnya :

```
Ketikkan sesuatu: sa
input terlalu pendek: panjang input: 2, minimal: 3
```

Try ... Finally

Ketika anda membaca file dari program anda. Bagaimana anda memastikan file akan ditutup baik ada eksepsi maupun tidak. Anda bisa menggunakan blok **finally** pada blok **try**.

```
# lat41.py

import time

try:
    f = open('coba.txt')
    while True:
        baris = f.readline()
        if len(baris) == 0:
            # EOF
            break
        print baris,
        time.sleep(2) # delay 2 detik

except KeyboardInterrupt:
    print '\nAnda membatalkan operasi'
finally:
    f.close()
    print '\nfile ditutup.'
```

outputnya :

```
ini adalah isi dari file
yang akan ditulis
menggunakan python
file ditutup.
```

Statemen with

Mendapatkan *resource* pada blok ``try`` dan melepasnya pada blok ``finally`` merupakan pola yang umum ditemukan. Oleh karena itu, anda dapat menggunakan menggunakan statemen ``with`` yang menyediakan mekanisme diatas secara otomatis.

```
# lat42.py

with open('coba.txt') as f:
    for baris in f:
        print baris,
```

outputnya :

```
ini adalah isi dari file  
yang akan ditulis  
menggunakan python
```


Object- Oriented Programming

Sumber :

<http://sakti.github.io/python101/oop.html>

Pendahuluan

Pada program yang selama ini kita buat, kita mendesain program kita berdasarkan fungsi (blok statemen yang memanipulasi data). Hal ini disebut pemrograman **procedure-oriented**.

Ada cara lain untuk mengorganisasi program dengan menggabungkan data dan operasi yang dibungkus dalam suatu obyek yaitu paradigma pemrograman berorientasi obyek.

Obyek memiliki field berupa variabel obyek dan method berupa fungsi obyek. Keduanya disebut atribut obyek. Class juga dapat memiliki field class (variabel class) dan method class. Class didefinisikan dengan keyword **class**.

this : self

Dalam deklarasi method pada class terdapat perbedaan yaitu ada parameter pertama yang harus ditambahkan pada parameter fungsi. Parameter ini diberi nama **self**, nilai dari parameter ini menunjuk ke obyek / instance itu sendiri.

*programmer Java, C# dan C++ terbiasa dengan keyword **this**.
Bedanya untuk Python variabel ini dikirim ke method secara eksplisit.*

Nilai self ini disediakan oleh Python. Contoh, ada class **ClassSaya** yang mempunyai instance obyek **obyeksaya**. Ketika method dipanggil pada obyek **obyeksaya.method(arg1, arg2)**, secara otomatis diubah oleh Python menjadi **ClassSaya.method(obyeksaya, arg1, arg2)**.

Class

Berikut contoh class yang sederhana.

```
# lat31.py

class Orang:
    pass

org = Orang()
print(org)
```

jika dijalankan akan mengeluarkan :

```
<__main__.Orang instance at 0x01800F80>
```

Menunjukkan variabel **org** adalah instance class **Orang** pada alamat memory **0x01800F80**.

Method Obyek

Berikut contoh deklarasi method pada class.

```
# lat32.py

class Orang:
    def katakanHalo(self):
        print 'Halo, apa kabar?'

org = Orang()
org.katakanHalo()
```

outputnya :

```
Halo, apa kabar?
```

*Perhatikan walaupun method **katakanHalo** tidak membaca parameter, masih ada **self** pada deklarasi method.*

Method init

Ada nama-nama method spesial pada class Python. **__init__** adalah salah satunya, method ini akan dijalankan ketika obyek dibuat. Method ini berguna untuk melakukan inisialisasi. Perhatikan garis bawah dua kali di awal dan di akhir method (**double underscore, dunder**). Hal ini seperti **constructor**.

```
# lat33.py

class Orang:

    def __init__(self, nama):
        self.nama = nama

    def katakanHalo(self):
        print 'Halo, nama saya %s, apa kabar?' % self.nama

org = Orang('budi')
org.katakanHalo()
```

outputnya :

```
Halo, nama saya budi, apa kabar?
```

Variabel Class dan Variabel Obyek (Instance)

Variabel Class yaitu variabel yang dimiliki oleh class, sedangkan variabel obyek adalah variabel yang dimiliki oleh tiap-tiap obyek instance dari class.

```
# lat34.py

class Orang:
    # variabel class, untuk menghitung jumlah orang
    total = 0

    def __init__(self, nama):
        # inisiasi data, data yang dibuat pada self merupakan variabel obyek
        self.nama = nama
        # ketika ada orang yang dibuat, tambahkan total orang
        Orang.total += 1

    def __del__(self):
        # kurangi total orang jika obyek dihapus
        Orang.total -= 1

    def katakanHalo(self):
        print 'Halo, nama saya %s, apa kabar?' % self.nama

    def total_populasi(cls):
        print 'Total Orang %s' % cls.total

    # method class
    total_populasi = classmethod(total_populasi)

org = Orang('budi')
org.katakanHalo()
Orang.total_populasi()

org2 = Orang('andi')
org2.katakanHalo()
Orang.total_populasi()

print 'obyek dihapus'
del org
del org2

Orang.total_populasi()
```

outputnya :

```
Halo, nama saya budi, apa kabar?
Total Orang 1
Halo, nama saya andi, apa kabar?
Total Orang 2
obyek dihapus
Total Orang 0
```

Inheritance

Salah satu keuntungan dari OOP adalah penggunaan ulang kode dan salah satu caranya yaitu menggunakan mekanisme **inheritance** / turunan.

Main Class

```
# base class / superclass
class AnggotaSekolah:

    "representasi anggota sekolah"

    def __init__(self, nama, umur):
        self.nama = nama
        self.umur = umur
        print 'membuat anggota sekolah baru: %s' % self.nama

    def info(self):
        "cetak info"
        print 'Nama: %s, Umur: %s' % (self.nama, self.umur)
```

Sub Class Guru

```
# subclass
class Guru(AnggotaSekolah):
    "representasi guru"

    def __init__(self, nama, umur, gaji):
        AnggotaSekolah.__init__(self, nama, umur)
        self.gaji = gaji
        print 'membuat guru: %s' % self.nama

    def info(self):
        AnggotaSekolah.info(self)
        print 'Gaji: %s' % self.gaji
```

Sub Class Siswa

```
# subclass
class Siswa(AnggotaSekolah):
    "representasi siswa"

    def __init__(self, nama, umur, nilai):
        AnggotaSekolah.__init__(self, nama, umur)
        self.nilai = nilai
        print 'membuat siswa: %s' % self.nama

    def info(self):
        AnggotaSekolah.info(self)
        print 'Nilai: %s' % self.nilai
```

Pemanggil

```
guru = Guru('Budi', 40, 3000000)
siswa = Siswa('Andi', 25, 75)

# cetak baris kosong
print

anggota = [guru, siswa]

for orang in anggota:
    orang.info()
```

outputnya :

```
membuat anggota sekolah baru: Budi
membuat guru: Budi
membuat anggota sekolah baru: Andi
membuat siswa: Andi
```

```
Nama: Budi, Umur: 40
Gaji: 3000000
Nama: Andi, Umur: 25
Nilai: 75
```

*semua Class tersebut beserta pemanggilnya
ada di dalam satu buah file.*