
Workshop Python 101

Release 1.0

Indocisc

November 18, 2016

1	Pendahuluan	1
1.1	Persiapan	1
1.2	Editor Teks	2
2	Dasar	9
2.1	Menggunakan Python sebagai kalkulator	9
2.2	Halo Dunia!	10
2.3	Komentar	11
2.4	Konstanta Literal	12
2.5	Bilangan	12
2.6	String	12
2.7	Variabel	13
2.8	Tipe Data	14
2.9	Obyek	14
2.10	Penulisan Program Python	15
3	Operator dan Ekspresi	17
3.1	Operator	17
3.2	Urutan Evaluasi	20
3.3	Mengubah Urutan Evaluasi	21
3.4	Sifat Asosiatif	21
4	Alur Kontrol	23
4.1	Statemen If	23
4.2	Statemen While	24
4.3	Perulangan For (For Loop)	25
4.4	Statemen Break	25
4.5	Statemen Continue	26
5	Fungsi (Function)	27
5.1	Parameter Fungsi	27
5.2	Variabel Lokal	28
5.3	Penggunaan Statemen Global	28
5.4	Nilai Argumen Default	29
5.5	Keyword Argumen	29
5.6	Parameter VarArgs	29
5.7	Statemen Return	30
5.8	Doc String	30

6	Modul	31
6.1	Byte-compiled (file .pyc)	31
6.2	Statemen from ... import	31
6.3	Nama Modul	32
6.4	Fungsi dir	32
6.5	Package	32
7	Struktur Data	35
7.1	List	35
7.2	Tuple	36
7.3	Dictionary	36
7.4	Sequence	37
7.5	Set	37
7.6	Referensi	38
7.7	String	38
8	Object-Oriented Programming (Pemrograman berorientasi Obyek)	41
8.1	this -> self	41
8.2	Class	41
8.3	Method Obyek	42
8.4	Method init	42
8.5	Variabel Class dan Variabel Obyek (Instance)	42
8.6	Inheritance	43
9	Input dan Output	45
9.1	Input dari Pengguna	45
9.2	File	45
9.3	Pickle	46
10	Eksepsi (Exception)	47
10.1	Syntax Error	47
10.2	Exception	47
10.3	Penanganan Exception	47
10.4	Mengeluarkan Exception	48
10.5	Try ... Finally	48
10.6	Statemen with	49
11	Library Standar (Standard Library)	51
11.1	Module getpass	51
11.2	Modul random	51
11.3	Modul datetime	51
11.4	Modul math	52
11.5	Modul sys	53
11.6	PYMOTW (Python Module of The Week)	53
12	Kontributor	55
12.1	Cara kontribusi	55
13	Referensi	57
14	Indeks dan tabel	59

Pendahuluan

Selamat datang di Workshop Python 101. Setelah workshop ini diharapkan anda bisa menggunakan bahasa pemrograman Python untuk memecahkan masalah di kehidupan sehari-hari (yang bisa dipecahkan dengan pemrograman tentunya).

1.1 Persiapan

Untuk dapat mengikut workshop ini pastikan Python interpreter sudah terinstal di komputer anda. Versi python yang digunakan untuk tutorial ini adalah versi 2.7.*.

Terdapat dua versi python yang saat ini ada versi 2 vs versi 3. Ada beberapa perbedaan syntax, operasi IO, perubahan struktur modul. Untuk *library* tambahan Python versi 3 masih kurang daripada versi 2. Jika anda sudah menguasai python 2 akan lebih mudah untuk bermigrasi ke versi 3.

Note: *Python 2.x is the status quo, Python 3.x is the present and future of the language*

1.1.1 Pengguna Windows

Untuk pengguna MS Windows. Python interpreter dapat di download di [Python.org](https://python.org) [Download](#). Kemudian pilih *individual releases*. Ada beberapa alternatif python installer untuk Windows (ActiveState, Enthought). Untuk workshop ini gunakan *default installer* dari python.org.

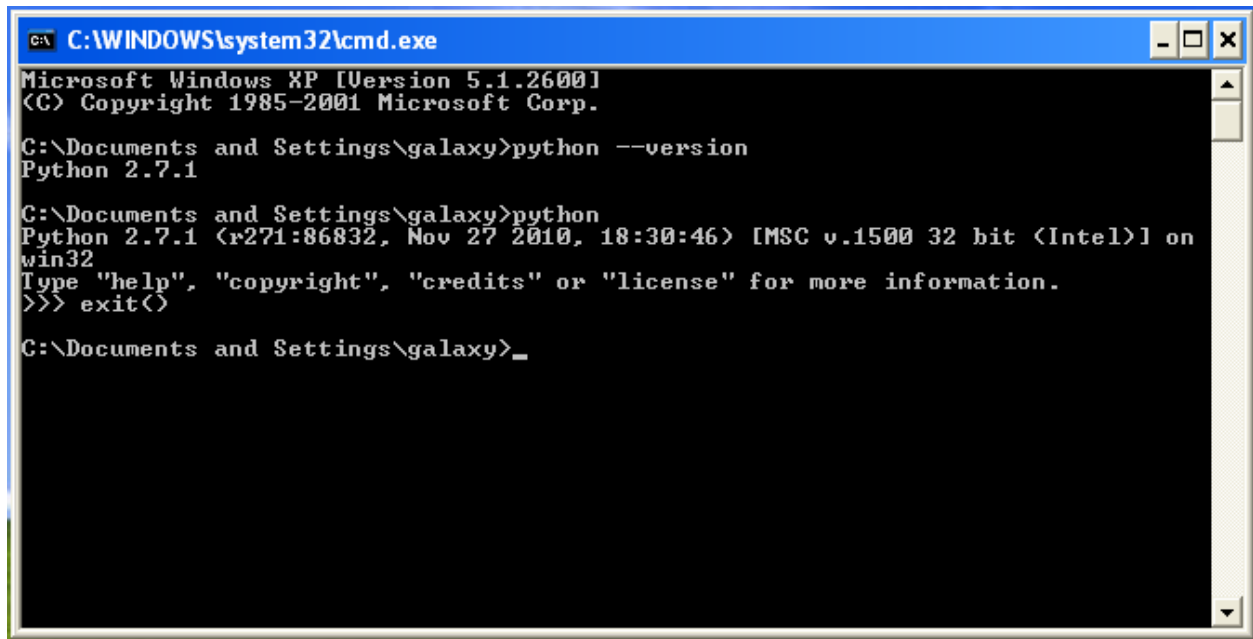
Note: Jika menemukan kesalahan 'python' is not recognized as an internal or external command, operable program or batch file.

cek windows PATH environment variable

1.1.2 Pengguna Linux

Pada umumnya distro linux sudah menyertakan Python secara default. Pastikan default python interpreter menunjuk ke python versi 2.*:

```
$ python --version
Python 2.7.3
```

A screenshot of a Windows command prompt window. The title bar at the top reads "C:\WINDOWS\system32\cmd.exe". The window contains the following text:

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\galaxy>python --version
Python 2.7.1

C:\Documents and Settings\galaxy>python
Python 2.7.1 (r271:86832, Nov 27 2010, 18:30:46) [MSC v.1500 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

C:\Documents and Settings\galaxy>_
```

Fig. 1.1: Python di Windows

1.1.3 Pengguna Mac OS X

Mac OS X secara default menyertakan Python interpreter.

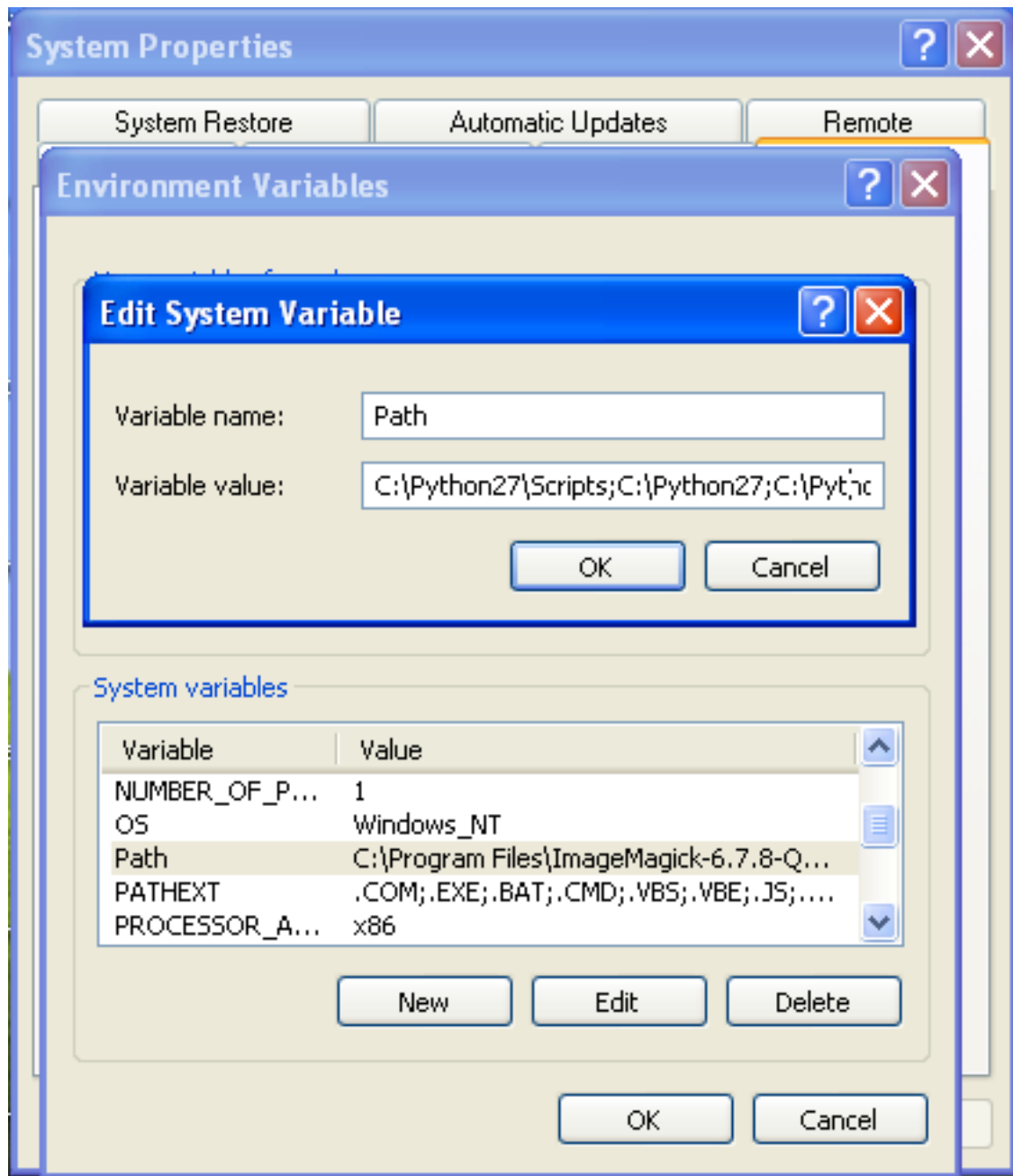
1.2 Editor Teks

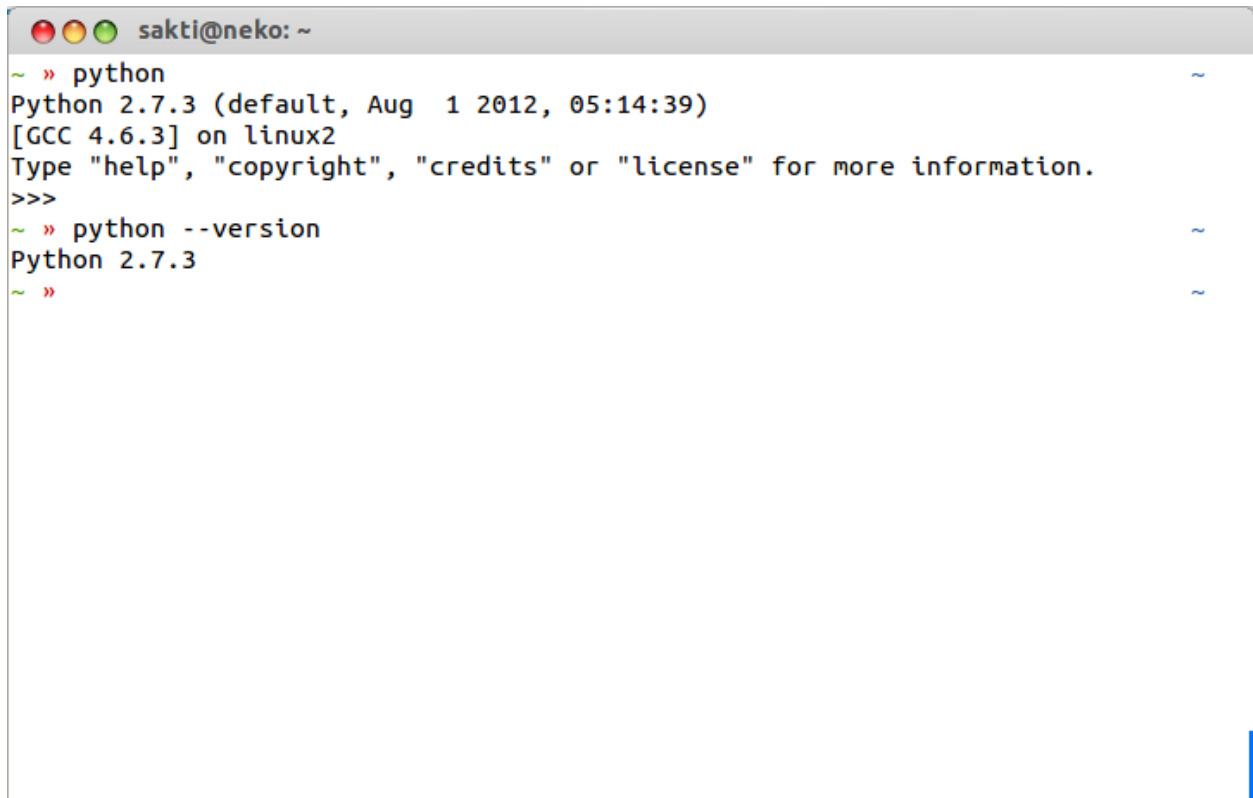
Anda bebas menggunakan editor teks kesayangan anda.

1.2.1 Vim

1.2.2 Emacs

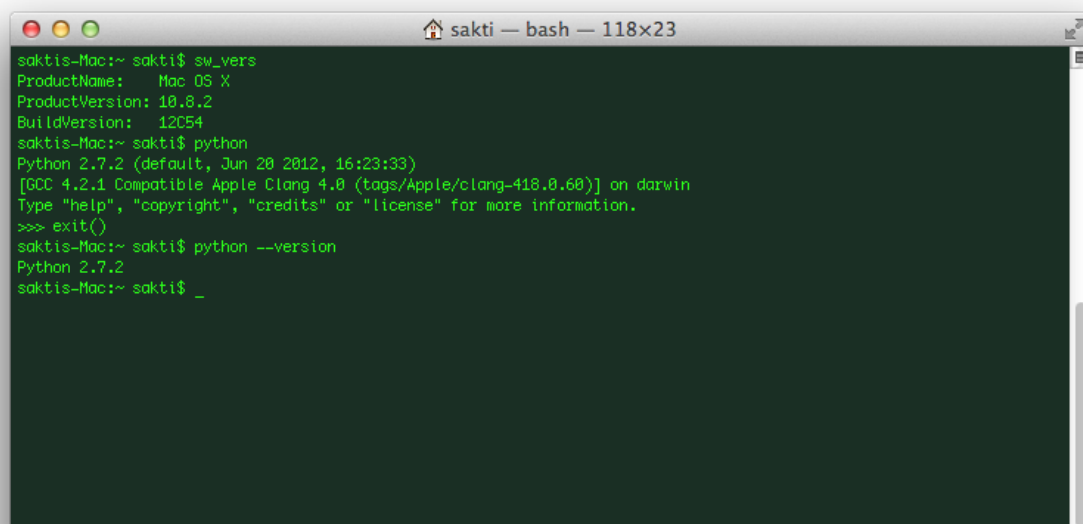
1.2.3 SublimeText 2





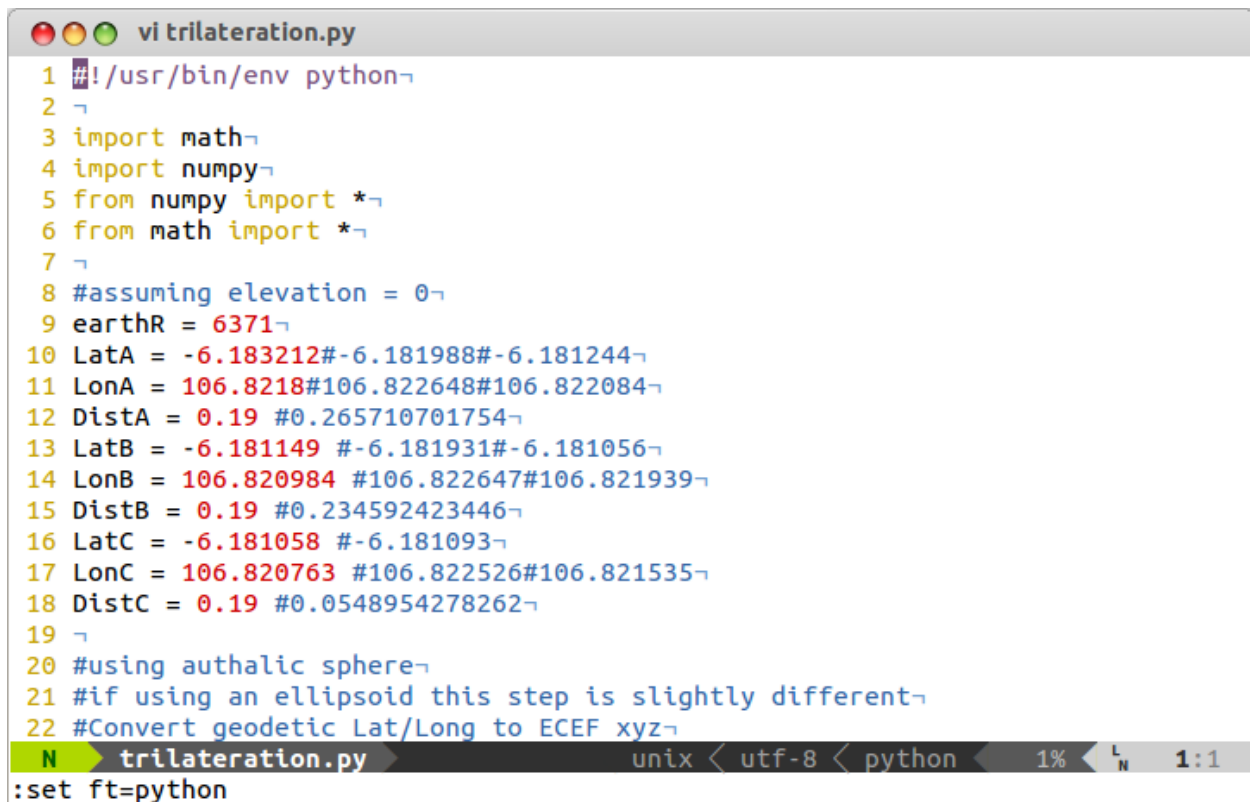
```
sakti@neko: ~  
~ » python  
Python 2.7.3 (default, Aug  1 2012, 05:14:39)  
[GCC 4.6.3] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>>  
~ » python --version  
Python 2.7.3  
~ »
```

Fig. 1.2: Python di Linux



```
sakti — bash — 118x23  
saktis-Mac:~ sakti$ sw_vers  
ProductName:   Mac OS X  
ProductVersion: 10.8.2  
BuildVersion:  12C54  
saktis-Mac:~ sakti$ python  
Python 2.7.2 (default, Jun 20 2012, 16:23:33)  
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin  
Type "help", "copyright", "credits" or "license" for more information.  
>>> exit()  
saktis-Mac:~ sakti$ python --version  
Python 2.7.2  
saktis-Mac:~ sakti$ _
```

Fig. 1.3: Python di Mac OS X



```
1 #!/usr/bin/env python
2
3 import math
4 import numpy
5 from numpy import *
6 from math import *
7
8 #assuming elevation = 0
9 earthR = 6371
10 LatA = -6.183212#-6.181988#-6.181244
11 LonA = 106.8218#106.822648#106.822084
12 DistA = 0.19 #0.265710701754
13 LatB = -6.181149 #-6.181931#-6.181056
14 LonB = 106.820984 #106.822647#106.821939
15 DistB = 0.19 #0.234592423446
16 LatC = -6.181058 #-6.181093
17 LonC = 106.820763 #106.822526#106.821535
18 DistC = 0.19 #0.0548954278262
19
20 #using authalic sphere
21 #if using an ellipsoid this step is slightly different
22 #Convert geodetic Lat/Long to ECEF xyz
N trilateration.py  unix < utf-8 < python  1%  1:1
:set ft=python
```

Fig. 1.4: Edit program python menggunakan Vim

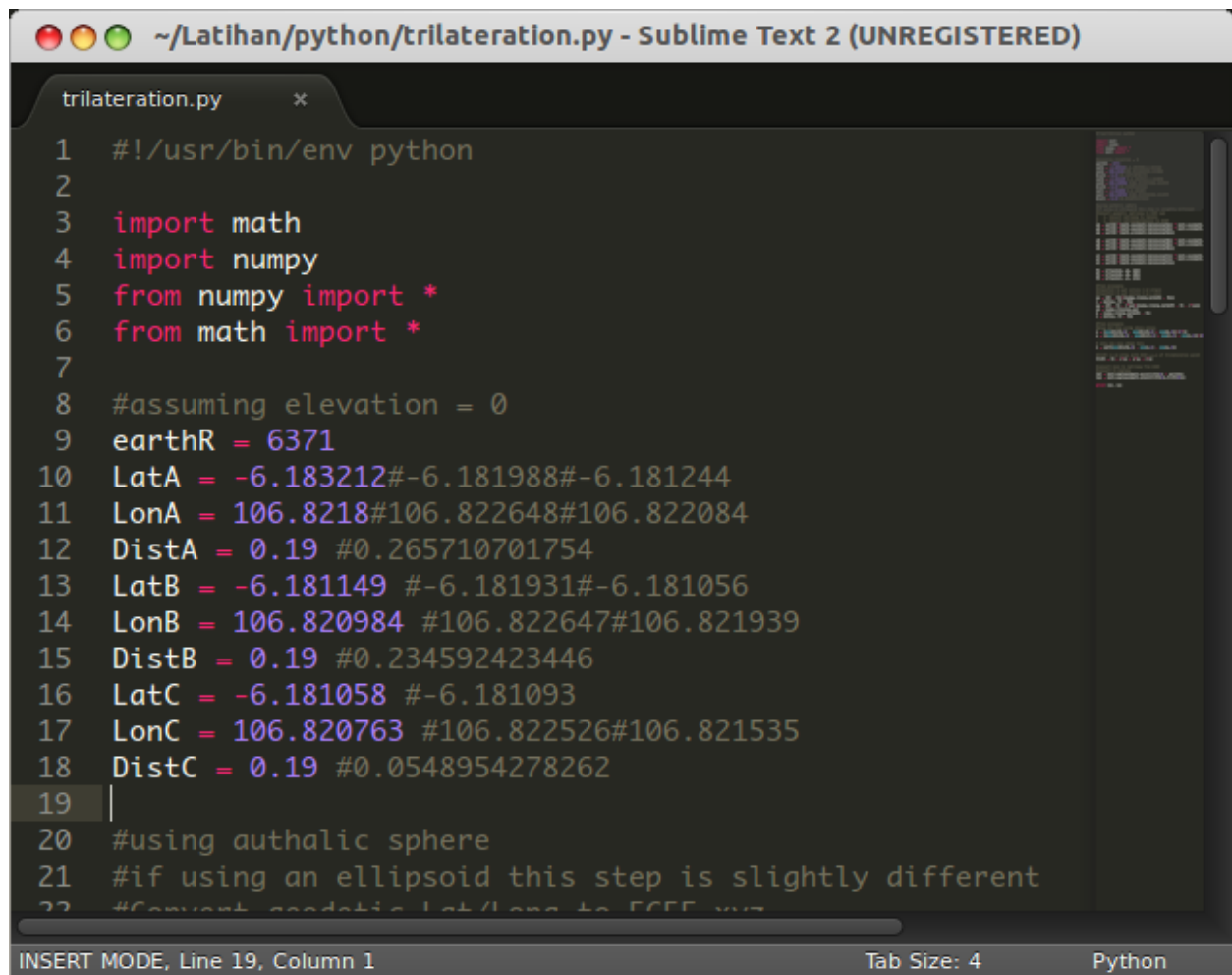
```
#!/usr/bin/env python

import math
import numpy
from numpy import *
from math import *

#assuming elevation = 0
earthR = 6371
LatA = -6.183212#-6.181988#-6.181244
LonA = 106.8218#106.822648#106.822084
DistA = 0.19 #0.265710701754
LatB = -6.181149 #-6.181931#-6.181056
LonB = 106.820984 #106.822647#106.821939
DistB = 0.19 #0.234592423446
LatC = -6.181058 #-6.181093
LonC = 106.820763 #106.822526#106.821535
DistC = 0.19 #0.0548954278262

#using authalic sphere
#if using an ellipsoid this step is slightly different
#Convert geodetic Lat/Long to ECEF xyz
# 1. Convert Lat/Long to radians
# 2. Convert Lat/Long(radians) to ECEF
xA = earthR *(math.cos(math.radians(LatA)) * math.cos(math.radians(LonA)))
yA = earthR *(math.cos(math.radians(LatA)) * math.sin(math.radians(LonA)))
--:**- trilateration.py Top L17 (Python)-----
;; This buffer is for notes you don't want to save, and for Lisp evaluation.
;; If you want to create a file, visit that file with C-x C-f,
;; then enter the text in that file's own bu
-U:**- *scratch* All L2 (Lisp Interaction)-----
Find file: ~/Latihan/python/
```

Fig. 1.5: Edit program python menggunakan Emacs



The image shows a screenshot of the Sublime Text 2 editor window. The title bar at the top reads "~/.Latihan/python/trilateration.py - Sublime Text 2 (UNREGISTERED)". The editor has a single tab open named "trilateration.py". The code is written in Python and is as follows:

```
1  #!/usr/bin/env python
2
3  import math
4  import numpy
5  from numpy import *
6  from math import *
7
8  #assuming elevation = 0
9  earthR = 6371
10 LatA = -6.183212#-6.181988#-6.181244
11 LonA = 106.8218#106.822648#106.822084
12 DistA = 0.19 #0.265710701754
13 LatB = -6.181149 #-6.181931#-6.181056
14 LonB = 106.820984 #106.822647#106.821939
15 DistB = 0.19 #0.234592423446
16 LatC = -6.181058 #-6.181093
17 LonC = 106.820763 #106.822526#106.821535
18 DistC = 0.19 #0.0548954278262
19 |
20 #using authalic sphere
21 #if using an ellipsoid this step is slightly different
22 #Convert geodetic lat/long to ECEF xyz
```

The status bar at the bottom of the editor shows "INSERT MODE, Line 19, Column 1" on the left, "Tab Size: 4" in the center, and "Python" on the right.

Fig. 1.6: Edit program python menggunakan Sublime Text2

Dasar

Dasar-dasar bahasa pemrograman Python.

2.1 Menggunakan Python sebagai kalkulator

Program Python dapat dijalankan dengan beberapa mode. Jika kita mengeksekusi Python interpreter tanpa argumen 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
```

2.1.1 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
```

Note: 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*.

2.2 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.

Note: Untuk catatan, anda sebaiknya menset teks editor anda agar 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!
```

Jika anda menggunakan SublimeText2 anda dapat menjalankannya menggunakan menu `Tools -> Build`, untuk linux anda dapat menggunakan shortcut `Ctrl+b`.



Fig. 2.1: Running program python menggunakan SublimeText2

2.3 Komentar

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

Berikut file latihan 2, perhatikan statemen print terakhir tidak akan dieksekusi karena berupa 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.'
```

2.4 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.

2.5 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} .

2.6 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:

```
"""Ini adalah contoh multi-line string
saya tambahkan single quote ' dan double
quote ", tanpa perlu meng-escape \\\n\nlebih dahulu"""
```

Contoh lain:

```
'''Ini adalah contoh multi-line string
saya tambahkan single quote ' dan double
quote ", tanpa perlu meng-escape \\\n\nlebih dahulu'''
```

Perhatikan perbedaan antara dua contoh diatas.

2.6.1 Immutable

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

2.6.2 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)
```

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

2.7 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.

2.7.1 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)
```

2.8 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.

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'>
```

2.9 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__',
>>> '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.
```

2.10 Penulisan Program Python

Berikut cara menulis program Python.

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

Note: Untuk pengguna sublimetext2 anda dapat menjalankan program python menggunakan shortcut `Ctrl+b`.

2.10.1 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 logic.

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

Note: 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.

2.10.2 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)
```

Operator dan Ekspresi

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.

3.1 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 1011 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.
^	(bit-wise XOR) Operasi bit-wise XOR (eksklusif OR). 5 ^ 3 menghasilkan 6.
~	(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 < 3 mengembalikan 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) Chapter 3. Operator dan Ekspresi Jika x bernilai True akan mengembalikan False. Jika x bernilai False akan mengembalikan True. x = True; not x mengembalikan False.
and	(boolean AND)

```

# 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

```

3.1.1 Cara lain 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
```

3.2 Urutan Evaluasi

Jika ada rangkaian 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
&	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

3.3 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
```

3.4 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 $a = (b = c)$.

Alur Kontrol

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 alur kerjanya? Sebagai contoh Anda ingin program untuk mengambil keputusan dan bertindak secara berbeda tergantung pada kondisi yang ada. Sebagai contoh, misalnya mencetak 'Selamat Pagi' atau 'Selamat Sore' tergantung waktu yang ada saat itu?

Hal ini dapat dilakukan lewat statemen alur kontrol. Ada tiga macam statemen alur kontrol di Python - `if`, `for` dan `while`.

4.1 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`.

```
1 # lat9.py
2
3 nomor_acak = 7
4 print 'tebak nomor acak dari 1 - 10'
5
6 # ``raw_input`` digunakan untuk mendapatkan input dari pengguna
7 # ``int`` digunakan untuk konversi tipe data ``str`` ke ``int``
8 tebakkan = int(raw_input('Tebakan anda (bil bulat): '))
9
10 if tebakkan == nomor_acak:
11     print 'Selamat! tebakkan anda benar'
12     print 'tapi tidak ada hadiah untuk anda :('
13 elif tebakkan < nomor_acak:
14     print 'tebakkan anda terlalu kecil'
15 else:
16     print 'tebakkan anda terlalu besar'
17
18 print 'selesai'
```

Note: Baris 11-12 adalah `if-block`, baris 14 adalah `elif-block`, dan baris 16 adalah `else-block`.

Bagaimana program ini bekerja?

Program ini akan meminta inputan tebakan dari pengguna berupa bilangan. 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.

Note: Cek modul `random` pada bab library standar untuk menghasilkan nomor acak.

4.2 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'
```

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
```

4.3 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 teratur.

```
# lat11.py

for i in range(1, 6):
    print i
else:
    print 'Perulangan sudah selesai'
```

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`.

4.4 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'
```

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

```
# lat13.py

for i in range(1, 11):
    print i
    if i == 5:
        break
else:
    print "Tidak dijalankan karena break"
```

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

4.5 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
```

Fungsi (Function)

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
```

5.1 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

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

cetak_maksimal(10, 100)
```

```
x = 9
y = 3

cetak_maksimal(x, y)
```

5.2 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
```

5.3 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
```


5.4 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)
```

5.5 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)
```

5.6 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)
```

5.7 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.

5.8 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__
```

Secara interaktif anda dapat mengakses docstring dengan fungsi help.

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

Modul

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
```

Selain modul .py kita dapat membuat modul dengan bahasa pemrograman C.

6.1 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*.

6.2 Statemen 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)
```

6.3 Nama Modul

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

Jika modul kita dijalankan *standalone* maka isi variabel `__nama__` 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__
```

6.4 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',
```

6.5 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

`__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)
```

Struktur Data

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

7.1 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*.

7.1.1 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 (tipe) `int`. Anda dapat membaca `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
```

7.2 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
```

7.3 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']
```

7.4 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[:]
```

7.5 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)
```

7.6 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
```

7.7 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)
```

Object-Oriented Programming (Pemrograman berorientasi Obyek)

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`.

8.1 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.

Note: 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)`.

8.2 Class

Berikut contoh class yang sederhana.

```
# lat31.py

class Orang:
    pass

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

jika dijalankan akan mengeluarkan

```
<__main__.Orang instance at 0x7f67decc9bd8>
```

Menunjukkan variabel `org` adalah instance class `Orang` pada alamat memory `0x7f67decc9bd8`.

8.3 Method Obyek

Berikut contoh deklarasi method pada class.

```
# lat32.py

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

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

Note: Perhatikan walaupun method `katakanHalo` tidak membaca parameter, masih ada `self` pada deklarasi method.

8.4 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*).

```
# 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()
```

8.5 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()

```

8.6 Inheritance

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

```

# lat35.py

# 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)

# 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

# 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

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

# cetak baris kosong
print

anggota = [guru, siswa]

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

Input dan Output

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.

9.1 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'
```

9.2 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()
```

9.3 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
```

Eksepsi (Exception)

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`

10.1 Syntax Error

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

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

10.2 Exception

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

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

10.3 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
```

10.4 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
```

10.5 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.'
```

10.6 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,
```

Library Standar (Standard Library)

11.1 Module getpass

Mendapatkan password pengguna tanpa *echo* kembali ke pengguna.

```
# contoh penggunaan modul getpass

import getpass

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

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

11.2 Modul random

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

```
# contoh penggunaan modul random

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)
```

11.3 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()
```

11.4 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))
```


11.5 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
```

11.6 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.

Kontributor

- Swaroop C H, untuk bukunya ‘Byte of Python’.
- Ahmad Sofyan, untuk translasi Bahasa Indonesia buku Byte of Python.
- Sakti Dwi Cahyono.

12.1 Cara kontribusi

Anda dapat berkontribusi untuk pengembangan modul ini dengan *forking* [Repo Python101](#) dan mengirimkan *pull request*.

Referensi

Belajar Python online:

- [Python Monk](#). Free, interactive tutorials to help you discover Python idioms, in your browser!
- [Code Academy](#). Learn to code interactively, for free.
- [Byte of Python](#), ([Byte of Python versi Indonesia](#) in progress)
- [Learn Python The Hard Way](#)

Note: Modul workshop ini diadopsi dari buku [Byte of Python](#) (lihat referensi).

Indeks dan tabel

- `genindex`
- `modindex`
- `search`