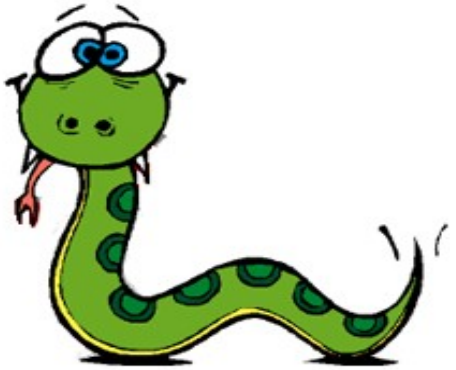
python

www.python.org



Files I/O

Dalam pembahasan kali ini kita akan mempelajari fungsi dasar file I/O yang tersedia di Python. Untuk fungsi lebih lanjut, lihat dokumentasi standar Python.

Fungsi dasar I/O

- Mencetak ke layar, print()
- Membaca input dari keyboard.
  - raw_input()
  - Input()
- Operasi baca dan tulis ke file
  - Open()
  - Close()
  - Write()
  - Read()
  - Tell()
  - Seek()

Mencetak ke layar

```
#!/usr/bin/python
print(“Hello World”)
```

Membaca input dari keyboard dengan raw_input() dan input()

```
>>> nama = raw_input(“Masukkan nama Anda: “)
>>> type(nama)
>>> print(nama)

>>> nilai = input(“Masukkan nilai: “)
>>>type(nilai)
>>>print (nilai)
```

sys.argv

- Argument system sebagai input

```
import sys
arg0 = sys.argv[0]
arg1 = sys.argv[1]
print (“Argument 0 :” + arg0)
print (“Argument 1 :” + arg1)
```

Membuka dan menutup File

- Sebelum Anda dapat membaca atau menulis ke file, Anda harus membukanya dengan menggunakan fungsi built-in python yaitu fungsi open().
- Fungsi open() ini membuat objek file, yang akan digunakan untuk memanggil metode pendukung lain yang terkait dengannya.
- **Syntax**:

  file object = open(file_name [, access_mode][, buffering])

Membuka dan menutup File

- ***file_name*** - Argumen file_name adalah nilai string yang berisi nama file yang ingin Anda akses.
- ***access_mode*** - *Access_mode* menentukan mode di mana file harus dibuka, yaitu membaca, menulis, menambahkan, dll. Daftar lengkap nilai yang mungkin diberikan di bawah ini dalam tabel. Ini adalah parameter opsional dan mode akses file default adalah baca (r).
- ***buffering*** - Jika nilai buffer diatur ke 0, tidak ada buffering yang terjadi. Jika nilai buffering adalah 1, line buffering dilakukan saat mengakses file. Jika Anda menentukan nilai buffering sebagai bilangan bulat lebih besar dari 1, maka aksi buffering dilakukan dengan ukuran buffer yang ditunjukkan. Jika negatif, ukuran buffer adalah default sistem (default behavior).

Membuka dan menutup File

- Mode akses:
  - **r,** Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
  - **rb**, Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
  - **r+**, Opens a file for both reading and writing. The file pointer placed at the beginning of the file.

Membuka dan menutup File

- Mode akses:
  - **w**, Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
  - **wb**, Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
  - **a**, Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.

Membuka dan menutup File

- Mode akses:
  - **ab**, Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writin
  - **a+**, Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
  - **ab+**, Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

Membuka dan menutup File

- Atribut Objek file:
  - **file.closed**, Returns true if file is closed, false otherwise.
  - **file.mode**, Returns access mode with which file was opened.
  - **file.name**, Returns name of the file.

Membuka file

```
#!/usr/bin/python
# Open a file
fo = open("test.txt", "w")
print "Nama file: ", fo.name
print "Tertutup atau tidak: ", fo.closed
print "Mode membuka: ", fo.mode
```

Menutup file

- Metode *close()*, menutup objek file, setelah itu tidak ada lagi proses menulis yang bisa dilakukan.
- Python secara otomatis menutup file ketika objek referensi dari sebuah file dipindahkan ke file lain.
- Adalah praktik yang baik untuk menggunakan metode close() untuk menutup file.

```
#!/usr/bin/python

# Open a file

fo = open("test.txt", "w")

print "Nama file: ", fo.name

fo.close()
```

Menulis File

- Metode write(), menulis string apapun ke file yang terbuka. Penting untuk dicatat bahwa string Python dapat berupa data biner dan bukan hanya teks.
- Metode write () tidak menambahkan karakter baris baru ('\n') ke akhir string
- Syntax:

```
fileObject.write(string);
```

Menulis File

- Contoh:

```
#!/usr/bin/python

# Open a file

fo = open("test.txt", "wb")

fo.write( "Python is a great
language.\nYeah its great!!\n")

# Close opend file

fo.close()
```

Membaca File

- Metode read(), membaca sebuah string dari file yang terbuka. Penting untuk dicatat bahwa string Python dapat memiliki data biner. selain data teks.
- Syntax:

```
fileObject.read([count])
```

Disini, parameter count yang dilewatkan adalah jumlah **byte** yang bisa dibaca dari file yang dibuka. Metode ini mulai membaca dari awal file dan jika tidak disebutkan count, maka mencoba baca sebanyak mungkin, mungkin sampai akhir file.

Membaca file

- Contoh:

```
#!/usr/bin/python
# Open a file
fo = open("test.txt", "r+")
str = fo.read(10);
print "String yang dibaca : ", str
# Close opend file
fo.close()
```

Posisi File

- Metode **tell()** memberitahu Anda posisi saat ini dalam file; Dengan kata lain, membaca atau menulis berikutnya akan terjadi pada jumlah byte dari awal file.
- Metode **seek (offset [, from])** mengubah posisi file saat ini. Argumen **offset** menunjukkan jumlah byte yang akan dipindahkan. Argumen **from** menentukan posisi referensi dari mana byte harus dipindahkan.
- Jika **from** diset ke 0, berarti menggunakan awal file sebagai posisi referensi dan 1 berarti menggunakan posisi saat ini sebagai posisi referensi dan jika diatur ke 2 maka akhir file akan diambil sebagai posisi referensi. .

Posisi file

- Contoh:

```
#!/usr/bin/python
# Open a file
fo = open("test.txt", "r+")
str = fo.read(10);
print "String yang dibaca : ", str
# Check current position
position = fo.tell();
print "Posisi file saat ini : ", position
# Reposition pointer at the beginning once again
position = fo.seek(0, 0);
str = fo.read(10);
print "String yang dibaca lagi : ", str
# Close opend file
fo.close()
```

Modul

- Apa itu modul didalam python ?

  Modul memungkinkan Anda mengatur kode Python secara logis. Mengelompokkan kode terkait ke dalam modul membuat kode lebih mudah dipahami dan digunakan.

- Secara sederhana, modul adalah file yang terdiri dari kode Python. Modul dapat mendefinisikan fungsi, kelas dan variabel. Modul juga bisa menyertakan kode **runnable**.

Pernyataan import

- Anda dapat menggunakan file sumber Python apapun sebagai modul dengan mengeksekusi pernyataan import di file sumber Python lainnya. Import memiliki sintaks berikut :

```
import module1 [, module2 [, ... moduleN]
```

Pernyataan from .. import

- Pernyataan Python **from** memungkinkan Anda mengimpor atribut tertentu dari modul ke dalam namespace saat ini.
- from ... import memiliki sintaks berikut :

```
from modname import name1[, name2[, ... nameN]]
```

Modul os

- Modul **os** Python menyediakan metode yang membantu Anda melakukan operasi pemrosesan file, seperti mengganti nama dan menghapus file.
- Untuk menggunakan modul ini, Anda perlu mengimpornya terlebih dulu dan kemudian Anda dapat memanggil fungsi terkait apa pun.

Modul os

- Contoh, berikut adalah contoh untuk mengganti nama file yang ada test1.txt -

```
#!/usr/bin/python

import os


# Rename a file from test1.txt to
test2.txt

os.rename( "test1.txt", "test2.txt" )
```

Modul os

- Contoh, berikut adalah contoh untuk menghapus file yang ada test2.txt -

```
#!/usr/bin/python

import os


# Delete file test2.txt

os.remove("text2.txt")
```

Modul os

- Anda dapat menggunakan metode **mkdir()** modul os untuk membuat direktori di direktori saat ini. Anda perlu memberikan argumen pada metode ini yang berisi nama direktori yang akan dibuat.
- Contoh berikut adalah contoh untuk membuat uji direktori di direktori saat ini -

```
#!/usr/bin/python

import os


# Create a directory "test"

os.mkdir("test")
```

Modul os

- Metode lainnya dari modul os:
  - os.chdir("newdir")
  - os.getcwd()
  - os.rmdir('dirname')
  - os.system(cmd)
  - os.popen(cmd)