

8. Membaca dan Menulis File

8.1 Tujuan Praktikum

Setelah mempelajari Bab ini, mahasiswa diharapkan dapat:

1. Dapat menjelaskan tentang File.
2. Dapat mengakses File dan memanipulasinya.
3. Dapat menggunakan File untuk penyimpanan data program.

8.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat komputer yang memiliki spesifikasi minimum sebagai berikut:

1. Terkoneksi ke Internet dan dapat mengunduh package-package Python.
2. Mampu menjalankan sistem operasi Windows 10 atau Ubuntu Linux.
3. File mbox.txt dan mbox-short.txt (ada di eclass)

Perangkat lunak yang diperlukan untuk mendukung praktikum ini adalah sebagai berikut:

1. Python 3.7 atau 3.8 yang terinstall menggunakan Anaconda atau Installer Python lainnya.
2. Web Browser (Mozilla Firefox, Microsoft Edge atau Google Chrome).
3. Command Prompt (jika menggunakan Windows).
4. Terminal (jika menggunakan Linux).
5. Editor Python (Visual Studio Code, PyCharm, Spyder atau editor-editor lainnya yang mendukung Python).

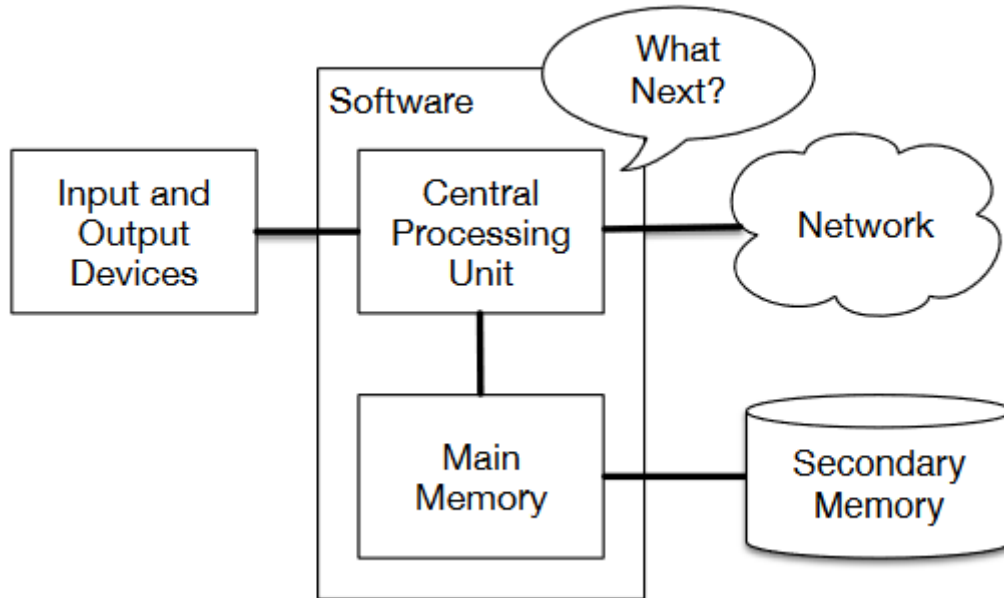
8.3 Materi

8.3.1 Pengantar File

Program yang berjalan membutuhkan memory primer di dalam komputer. Semua data yang ada di program tersebut disimpan di dalam memory dan ketika program selesai dijalankan dan dimatikan, maka semua data di dalam program tersebut juga ikut hilang. Penyimpanan data di dalam

memory bersifat tidak permanen (volatile). Karena sifat tersebut, program yang menggunakan memory primer tidak akan dapat menyimpan data setelah program dimatikan.

Untuk bisa menyimpan data pada program harus digunakan penyimpanan tetap yaitu secondary memory. Secondary memory dapat dilihat pada gambar 8.1



Gambar 8.1: Secondary Memory di Komputer

File disimpan pada secondary memory sehingga file dapat digunakan untuk menyimpan data dari program dan tidak akan hilang walaupun komputer dimatikan. File pada dasarnya adalah bit-bit data yang disimpan di dalam secondary memory secara permanen, berupa kumpulan informasi yang saling berelasi satu sama lain sebagai satu kesatuan. File bisa berupa file system, file program (binary), file multimedia, file teks, dan lain sebagainya. File memiliki property seperti nama file, ukuran, letak di harddisk, owner, hak akses, tanggal akses, dan lain-lain. Contoh property sebuah file dapat dilihat dari gambar 8.2

8.3.2 Pengaksesan File

Untuk dapat mengakses file, langkah-langkah yang harus dilakukan adalah:

1. Menyiapkan file dan path yang akan diakses
2. Open file
3. Lakukan sesuatu dengan file tersebut, seperti ditampilkan (read) isinya atau diubah / ditulisi (write)
4. Close file

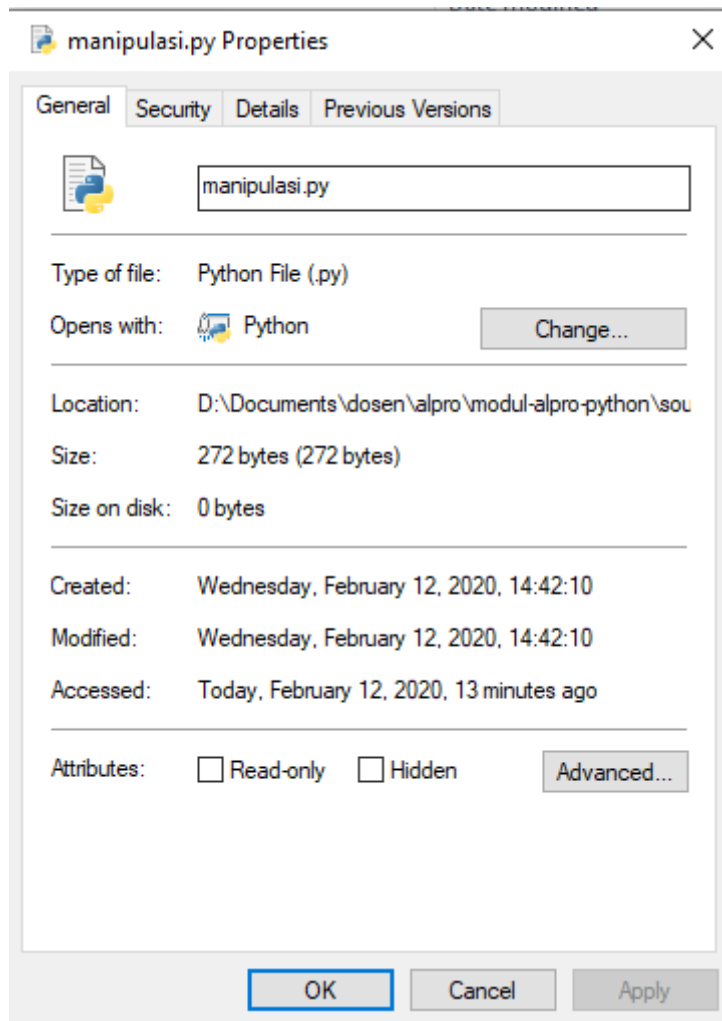
Gambaran handle file dapat dilihat pada 8.3

Program Python dapat dibuat sebagai berikut:

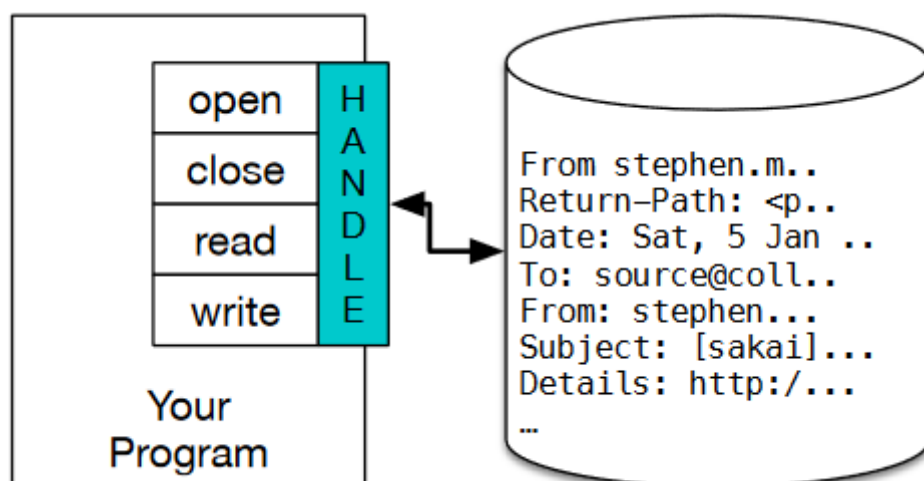
```
1 handle = open('mbox-short.txt')
2 print(handle)
```

Hasil adalah:

```
1 <_io.TextIOWrapper name='mbox-short.txt' mode='r' encoding='UTF-8'>
```



Gambar 8.2: Contoh Property File



Gambar 8.3: Ilustrasi Handle File

Jadi hasilnya berupa tampilan nama file, modenya (r = read), dan encoding yang digunakan yaitu unicode UTF-8 dari sistem io pada Python. Jika nama file tidak ada / tidak ditemukan, maka output akan error:

```
1 Traceback (most recent call last): File "main.py", line 1, in <module>
2 handle = open('tidak-ada.txt')
3 FileNotFoundError: [Errno 2] No such file or directory: 'tidak-ada.txt'
```

Pada file teks, biasanya file akan terdiri dari baris demi baris string. Cara pembacaan file teks biasanya juga menggunakan model baca baris demi baris untuk setiap string yang ditemukan sampai dengan EOF (End of File)

Contoh tampilan baris-baris dari sebuah file teks mbox-short.txt adalah pada gambar 8.4

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Date: Sat, 5 Jan 2008 09:12:18 -0500
To: source@collab.sakaiproject.org
From: stephen.marquard@uct.ac.za
Subject: [sakai] svn commit: r39772 - content/branches/
Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772
...
```

Gambar 8.4: Contoh File Teks dan Barisnya

8.3.3 Manipulasi File

Untuk bisa memanipulasi file maka harus dimulai dari membaca file tersebut terlebih dahulu. Cara membaca file pada Python adalah:

1. Siapkan file
2. Open file
3. Loop setiap baris pada file
4. Close file

Cara pembacaan file adalah:

```
1 handle = open('mbox-short.txt')
2 count = 0
3 for line in handle:
4     count = count + 1
5 print('Line Count:', count)
```

Hasilnya adalah Line Count: 1910




Mengapa harus dibaca baris-perbaris? Untuk mengatasi kemungkinan sistem membuka file yang besar sekali dalam satu waktu dan akhirnya hang / crash karena ukuran file yang sangat besar.

Cara menampilkan ukuran file teks dalam bytes, dapat digunakan fungsi len dari string yang ada pada file.

```
1     handle = open('mbox-short.txt')
2     hasil = handle.read()
3     print("Ukuran: " + len(hasil) + "bytes")
4     print("Huruf dari belakang sendiri mundur 16 huruf adalah: " + hasil[-16::1])
```

Program di atas akan membuka file mbox-short.txt, menampilkan ukuran berapa banyak huruf yang ada pada file tersebut (catatan: kalau dianggap 1 karakter = 1 byte, maka bisa disebut juga ukuran berapa banyak karakter = ukuran file tersebut dalam byte), dan terakhir menampilkan string dari huruf paling belakang maju 16 huruf kedepan.

 **Hati-hati!** Perintah read() pada file sangat boros memory, sehingga jika ukuran file begitu besar maka lebih baik tidak menggunakan read(), melainkan menggunakan teknik loop seperti pada contoh di atas sebelumnya.

Selama dilakukan looping kita juga dapat melakukan manipulasi terhadap file tersebut, seperti misalnya menangkap / menampilkan bagian dari string. Seperti pada contoh file mbox, saat looping kita dapat menampilkan hanya kalimat yang diawali dengan "tanggal" saja, yaitu "Date :". Perhatikan contoh program berikut:

```
1     handle = open('mbox-short.txt')
2     count = 1
3     for line in handle:
4         if line.startswith("Date:") and count <= 10:
5             count += 1
6             print(line)
```

Program di atas menghasilkan output 10 baris yang berupa tanggal saja.

Date: Sat, 5 Jan 2008 09:12:18 -0500

Date: 2008-01-05 09:12:07 -0500 (Sat, 05 Jan 2008)

Date: Fri, 4 Jan 2008 18:08:57 -0500

Date: 2008-01-04 18:08:50 -0500 (Fri, 04 Jan 2008)

Date: Fri, 4 Jan 2008 16:09:02 -0500

Date: 2008-01-04 16:09:01 -0500 (Fri, 04 Jan 2008)

Date: Fri, 4 Jan 2008 15:44:40 -0500

Date: 2008-01-04 15:44:39 -0500 (Fri, 04 Jan 2008)

Date: Fri, 4 Jan 2008 15:01:38 -0500

Date: 2008-01-04 15:01:37 -0500 (Fri, 04 Jan 2008)

Mengapa ada baris kosong disetiap baris di atas? hal ini karena dari file mbox sudah terdapat newline di setiap baris dan ditambah perintah print yang kita buat, sehingga terjadi dobel newline. Silahkan modifikasi agar enter tidak dobel! Kita bisa menggunakan perintah `rstrip` pada line atau menggunakan perintah `print` tanpa `newline`.

8.3.4 Penyimpanan File

Dalam Python cara untuk menulis ke file adalah sama dengan cara membuka (`open`) file pada sub bab sebelumnya, hanya perlu mengubah metodenya saja yang tadinya `r` menjadi `w` sebagai berikut: **`fout=open('output.txt','w')`** Untuk menuliskan isi string ke dalam file `output.txt` langsung saja digunakan perintah `write(<string>)` dan jangan lupa tutup file dengan `close()`

Contoh lengkap adalah sebagai berikut:

```
1 handle = open('output.txt','w')
2 tulisan = "teks ini akan dituliskan ke file\n"
3 handle.write(tulisan)
4 handle.close()
```

8.4 Kegiatan Praktikum

Kegiatan praktikum akan dilakukan untuk memanipulasi lebih dalam lagi file teks `mbox-short.txt` sebagai berikut:

Kasus 8.1 Program harus mampu menerima nama file teks tertentu (`mbox-short.txt` atau `mbox.txt`) dan kemudian tampilkanlah **semua baris** yang mengandung string web pada file tersebut yang menggunakan domain berakhiran `*.ac.uk` dan **berapa jumlahnya**.

Pembahasan Soal 1

Untuk dapat menerima input nama file digunakan perintah `input`, kemudian masukkan nama file. File tersebut sebaiknya berada dalam satu folder yang sama agar mudah. Untuk menampilkan baris-baris file yang mengandung string URL web yang mengandung domain berakhiran: `*.ac.uk` digunakan perintah `find()`. Untuk menampilkan jumlah baris digunakan counter untuk setiap baris yang ditemukan. Berikut adalah kode programnya.

```
1 filename = input("nama file: ")
2 handle = open(filename)
3 c = 0
4 for line in handle:
5     if line.find("ac.uk") != -1:
6         c += 1
7     print("Web domain 'ac.uk' ditemukan di \"" + line.strip() + "\"")
8 print("Jumlah: ",c)
```

Silahkan lihat hasil programnya!

Kasus 8.2 Program harus mampu menerima nama file teks tertentu (`mbox-short.txt` atau `mbox.txt`) dan kemudian tampilkanlah **berapa baris** string pada file yang diawali kata "Subject" dan ubahlah semua baris tersebut menjadi "capitalized each word".

Pembahasan Soal 2

Untuk dapat menerima input nama file digunakan perintah input, kemudian masukkan nama file. File tersebut sebaiknya berada dalam satu folder yang sama agar mudah. Untuk menampilkan baris-baris file yang diawali string 'Subject:' digunakan perintah startswith(). Untuk menampilkan jumlah baris digunakan counter untuk setiap baris yang ditemukan. Berikut adalah kode programnya.

```
1 filename = input("nama file: ")
2 handle = open(filename)
3 c = 0
4 for line in handle:
5     if line.startswith("Subject:"):
6         c += 1
7         line = line.strip().title()
8         print(line)
9 print("Jumlah: ",c)
```

Silahkan lihat hasil programnya!

Kasus 8.3 Program harus mampu menampilkan ukuran file dalam KB dari sebuah file teks dan menghandle error jika file yang diinputkan tidak ditemukan.

Pembahasan Soal 3

Untuk dapat menerima input nama file digunakan perintah input, kemudian masukkan nama file. File tersebut sebaiknya berada dalam satu folder yang sama agar mudah. Untuk menghitung ukuran file dapat digunakan perintah len untuk setiap baris string yang diperoleh dari pembacaan baris file. Ukuran file yang didapat adalah berupa byte, sehingga perlu dibagi 1000 yang akan menjadi dalam KB. Untuk menghandle error, gunakan blok try catch. Berikut adalah kode programnya.

```
1 filename = input("nama file: ")
2 try:
3     handle = open(filename)
4     total = 0
5     for line in handle:
6         total += len(line)
7         kb = total / 1000
8         print("Ukuran: " + str(mb) + " KB")
9 except:
10    print("File tidak ditemukan!")
```

Silahkan lihat hasil programnya!

8.5 Latihan Mandiri

Latihan 8.1 Buatlah sebuah program yang dapat membandingkan 2 buah file teks dan kemudian menampilkan perbedaan antar kedua teks per barisnya jika ada perbedaan! ■

Latihan 8.2 Buatlah sebuah program untuk menampilkan soal sederhana yang diambil dari file teks soal.txt yang memiliki format sebagai berikut:

```
1+1 = || 2
Bendera Indonesia? || Merah Putih
Kota gudeg adalah: || Yogyakarta
Komponen PC untuk penyimpanan file adalah... || harddisk
50 * 20 = || 1000
```

Dari soal tersebut tampilkan sbb:

```
nama file1: soal.txt
1+1 =
Jawab: 2
Jawaban benar!
Bendera Indonesia?
Jawab: merah putih
Jawaban benar!
Kota gudeg adalah:
Jawab: yogya
Jawaban salah!
Komponen PC untuk penyimpanan file adalah...
Jawab: HARDDISK
Jawaban benar!
```