Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230970**
Nama Lengkap	**Gregorius Daniel Jodan Perminas**
Minggu ke / Materi	**08 / Membaca dan Menulis File**

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2024

BAGIAN 1: MATERI MINGGU INI (40%)

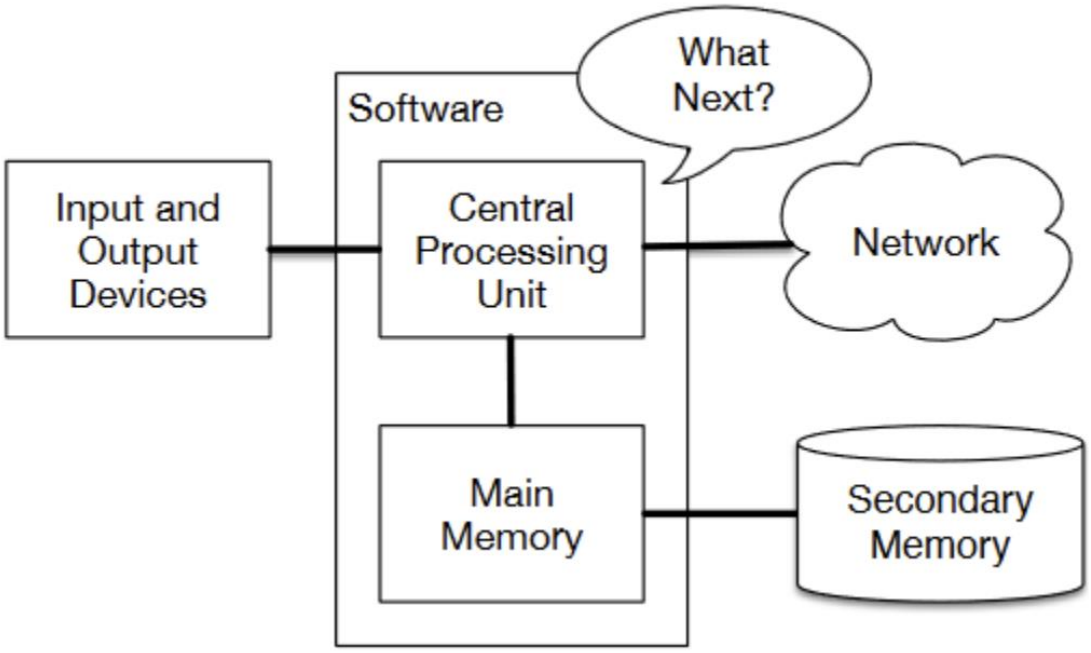
Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

Pengantar File

Dalam komputasi, primary memory atau memori utama adalah tempat di mana komputer menyimpan data yang sedang diproses atau digunakan. Ini termasuk data yang dibutuhkan oleh program yang sedang berjalan, serta hasil dari operasi yang dilakukan oleh program tersebut. Memori utama seringkali volatile, yang berarti data yang disimpan di dalamnya akan hilang ketika komputer dimatikan atau program selesai dijalankan.

Sebaliknya, secondary memory atau memori sekunder adalah tempat di mana komputer menyimpan data secara permanen. Ini termasuk hard drive, SSD, dan media penyimpanan lainnya. Data yang disimpan di memori sekunder tidak akan hilang ketika komputer dimatikan, sehingga memungkinkan kita untuk menyimpan hasil dari program kita untuk digunakan di kemudian hari.

Dalam konteks pemrograman, kita sering kali perlu menyimpan hasil dari program kita ke dalam file yang disimpan di memori sekunder. Ini memungkinkan kita untuk mempertahankan hasil dari program kita meskipun program tersebut sudah selesai dijalankan. Misalnya, kita mungkin ingin menyimpan hasil dari perhitungan yang kompleks, atau kita mungkin ingin menyimpan data yang dihasilkan oleh pengguna.



Gambar 1 - Secondary Memory di Komputer

Pengaksesan File

Istilah lain pengaksesan File pada Python adalah File Handling. Keduanya merujuk pada proses membaca, menulis, memperbarui, dan menghapus file. File handling adalah bagian penting dari aplikasi web atau program lainnya. File Handling memungkinkan kita untuk menyimpan output dari program tertentu dalam file dan melakukan operasi tertentu pada file tersebut. Beberapa hal yang perlu disiapkan untuk mengakses file antara lain:

1. Menyiapkan File dan Path yang akan diakses
2. Menggunakan perintah Open File
3. Melakukan modifikasi pada File (read, write, append, dll)
4. Close File

Ada empat metode atau *modes* yang bisa digunakan untuk *open file*:

"r" - Read - Default value. Membuka file untuk dibaca, akan terjadi error apabila file tidak ditemukan

"w" - Write – Membuka file untuk ditulis (diganti isinya), membuat file baru apabila file tidak ditemukan

"a" - Append - Membuka file untuk ditambah isinya, membuat file baru apabila file tidak ditemukan

"x" - Create – Membuat file baru, dan akan terjadi error apabila file sudah ada

Kita juga bisa menentukan apakah file harus di *handle* dalam bentuk *binary* atau *text modes:*

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (Images)

Demonstrasi Menggunakan Python:

- Membaca File

```
1  f = open("mbox-short.txt", "r")
2  print(f.read())
```

- Menulis File

```
1  f = open("mbox-short.txt", "w")
2  f.write("Hello There!")
```

- Menambah Isi File

```
1  f = open("mbox-short.txt", "a")
2  f.write("\nThis is a new line!")
```

- Menutup File

```
1   f = open("mbox-short.txt", "a")
2   f.write("\nThis is a new line!")
3   f.close()
```

Proses menutup file ketika sudah digunakan sangat diperlukan untuk menjaga *data integrity* dan mengurangi penggunaan *system resource* yang tidak perlu.

Manipulasi File

Untuk bisa memanipulasi file maka file harus dibaca terlebih dahulu

Berikut langkah-langkahnya:

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

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

```
Line Count: 1910
```

Gambar 2 – Metode Manipulasi File dengan Looping

Digunakan looping untuk membaca file baris per baris. Hal ini dilakukan untuk mencegah terjadinya *crash* ketika membuka file dengan ukuran yang sangat besar.

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

```
Ukuran: 94626 bytes
Huruf dari belakang sendiri mundur 16 huruf adalah: > Preferences.
```

Gambar 3 - Metode Manipulasi File dengan Fungsi Read

Program di atas akan membuka file yang bernama 'mbox-short.txt', kemudian fungsi len() akan menghitung jumlah karakter yang ada dalam variabel hasil. Variabel hasil ini berisi sekumpulan karakter yang diperoleh dari fungsi handle.read(). Penting untuk diingat bahwa fungsi read() dapat menggunakan memori secara signifikan, sehingga saat bekerja dengan file berukuran besar, lebih disarankan untuk menggunakan teknik iterasi atau looping. Terakhir, program ini akan menampilkan 16 karakter terakhir dari string, dimulai dari karakter paling belakang dan bergerak maju.

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

```
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)
```

Gambar 4 - Metode Manipulasi File dengan Loop 2

Selama melakukan looping kita juga dapat melakukan manipulasi terhadap file tersebut, misalnya menangkap dan menampilkan bagian dari suatu string. Seperti pada contoh file mbox-short, saat looping kita dapat menampilkan hanya kalimat yang diawali dengan "tanggal" saja, yaitu "Date :"

Penyimpanan File

Dalam Python cara untuk menulis ke file cukup mirip dengan cara membuka (open) file pada sub bab sebelumnya, hanya perlu mengubah metodenya saja yang tadinya ’r’ menjadi ’w’. Untuk menuliskan isi string ke dalam file langsung saja digunakan perintah write dan jangan lupa tutup file dengan close().

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

Gambar 5 - Metode Penyimpanan File pada Python

BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

LINK GITHUB: https://github.com/danieljodan/PrakAIPro8_A_71230970.git

SOAL 1

```python
# Program Membandingkan Isi Dua File
def bandingkan_teks(file1, file2):
    with open(file1, 'r') as f1, open(file2, 'r') as f2:
        handle1 = f1.readlines()
        handle2 = f2.readlines()
    ada_perbedaan = False
    i = 0
    for line1, line2 in zip(handle1, handle2):
        if line1 != line2:
            print(f"Perbedaan pada baris {i+1}:")
            print(f"File 1: {line1.strip()}")
            print(f"File 2: {line2.strip()}")
            print()
            ada_perbedaan = True
        i += 1
    if ada_perbedaan == False:
        print("Tidak ada perbedaan")
    f1.close()
    f2.close()

file1 = "fileSatu.txt"
file2 = "fileDua.txt"
bandingkan_teks(file1, file2)
```

```
Perbedaan pada baris 3:
File 1: Daniel
File 2: Danielo

Perbedaan pada baris 7:
File 1: Kampus Gemink
File 2: Kampus
```

Gambar 6 - Program Untuk Soal Latihan Mandiri 8.1

Pertama terdapat fungsi bandingkan_teks yang menerima dua argumen, yaitu file1 dan file2, yang merupakan nama dari dua file teks yang ingin dibandingkan.

Kemudian fungsi open digunakan untuk membuka kedua file tersebut dalam mode baca (‘r’). Isi dari kedua file tersebut dibaca dan disimpan dalam variabel ”handle1” dan ”handle2” menggunakan metode readlines(), yang mengembalikan list dari setiap baris dalam file.

Variabel ada_perbedaan diinisialisasi sebagai False. Variabel ini akan digunakan untuk melacak apakah ada perbedaan antara kedua file.

Fungsi zip digunakan untuk menggabungkan ”handle1” dan ”handle2” menjadi pasangan baris yang sesuai dari kedua file. Kemudian, untuk setiap pasangan baris, program memeriksa apakah baris tersebut sama. Jika tidak, program mencetak baris ke berapa yang berbeda dan apa isi dari kedua baris tersebut. Variabel ada_perbedaan diatur menjadi True.

Jika setelah memeriksa semua baris dan tidak ada perbedaan yang ditemukan (ada_perbedaan = False), program akan mencetak “Tidak ada perbedaan”.

Akhirnya, fungsi bandingkan_teks dipanggil dengan file1 dan file2 sebagai argumen. Ini akan membandingkan kedua file tersebut dan mencetak perbedaan, jika ada.

SOAL 2

```python
# Program Menampilkan Soal dan Mengecek Jawaban
Comment Code
def tampilkan_soal(nama_file):
    with open(nama_file, 'r') as f:
        soal_jawaban = f.readlines()
    for line in soal_jawaban:
        soal,jawaban = line.strip().split('||')
        print(soal)
        jawaban_user = input("Jawab: ")
        if jawaban_user.strip().lower() == jawaban.strip().lower():
            print("Jawaban benar!")
        else:
            print("Jawaban salah!")
    f.close()

nama_file = "soaljawab.txt"
tampilkan_soal(nama_file)
```

```
1+1 =
Jawab: 2
Jawaban benar!
Bendera Indonesia?
Jawab: merah putih
Jawaban benar!
Kota gudeg adalah:
Jawab: yogya
Jawaban salah!
```

Gambar 7 - Program Untuk Soal Latihan Mandiri 8.2

Fungsi tampilkan_soal menerima satu argumen, yaitu nama_file, yang merupakan nama dari file teks yang berisi soal dan jawaban.

Fungsi open digunakan untuk membuka file tersebut dalam mode read atau ‘r’. Konten dari file tersebut dibaca dan disimpan dalam variabel soal_jawaban menggunakan metode readlines(), yang mengembalikan list dari setiap baris dalam file.

Untuk setiap baris dalam soal_jawaban, program melakukan hal berikut:

- Menghapus spasi di awal dan akhir baris dengan strip().
- Memisahkan soal dan jawaban berdasarkan pemisah ‘||’ dengan split('||').
- Menampilkan soal dengan print(soal).
- Menerima jawaban dari pengguna dengan input("Jawab: ").
- Membandingkan jawaban pengguna dengan jawaban yang ada di file. Jika sama, program mencetak “Jawaban benar!”. Jika berbeda, program mencetak “Jawaban salah!”.

Setelah selesai menampilkan semua soal dan mengecek jawaban, file ditutup dengan f.close().

Akhirnya, fungsi tampilkan_soal dipanggil dengan nama_file sebagai argumen. Ini akan menampilkan soal dari file tersebut dan mengecek jawaban pengguna.