Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230978**
Nama Lengkap	**Jonathan Satriani Gracio Andrianto**
Minggu ke / Materi	**09 / 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.

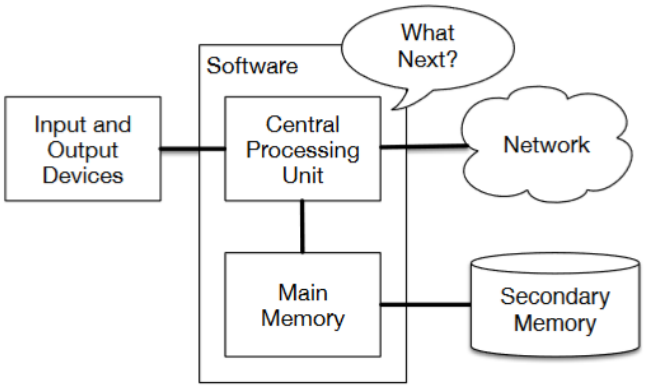
SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

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

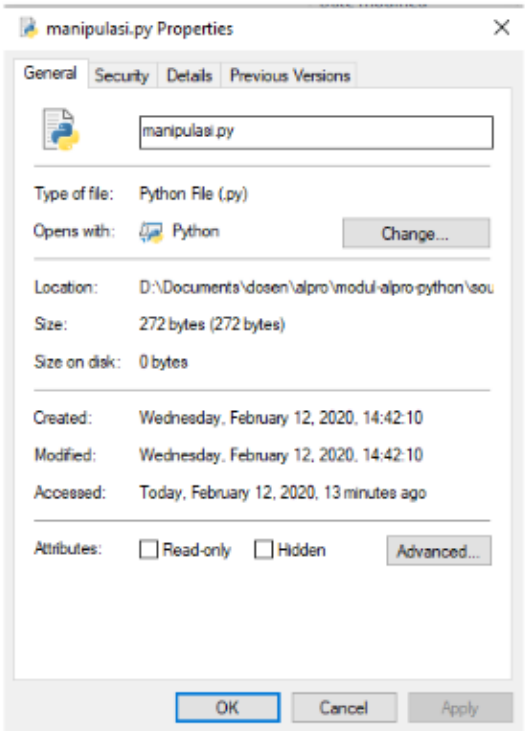
BAGIAN 1: MATERI MINGGU INI (40%)

Pengantar File

Program membutuhkan memory primer untuk berjalan, tetapi data di dalamnya hanya tersimpan sementara dan akan hilang setelah program dimatikan karena sifat volatile memory. Untuk menyimpan data secara permanen, diperlukan penyimpanan tetap seperti secondary memory. File disimpan di secondary memory dan dapat digunakan untuk menyimpan data antar-sesi program serta tidak akan hilang saat komputer dimatikan. Secondary memory dapat dilihat di gambar dibawah :



File adalah kumpulan data yang tersimpan secara permanen di secondary memory, seperti file system, file program (binary), multimedia, teks, dll. Setiap file memiliki properti seperti nama, ukuran, lokasi di harddisk, owner, hak akses, dan tanggal akses.



Property file

Contoh penggunaan file dalam python :

```
handle = open("pert9.txt", "w")
handle.write("hello ges")

handle = open("pert9.txt", "w+")
handle.write("Hello World\n")
handle.write("Pertemuan 9")
```

Penggunaan file dalam python

Pengaksesan File

Jika kita ingin mengakses file pada python, maka kita harus menyiapkan file dan path yang akan diakses. Kemudian kita open file dengan fungsi open(). Dengan fungsi open(), kita dapat membaca (read), menulis (write) dan manambahkan isi ke file. Dan ketika selesai bekerja dengan file, maka kita harus close file dengan close().

Ada banyak mode dalam mengakses file. Antara lain :

a. “r” (Read) : Mode read adalah mode default. Fungsinya untuk membaca file.
b. “w” (Write) : Fungsinya untuk menulis sesuatu ke dalam file. Jika file tidak ada, maka akan otomatis dibuat. Sedangkan jika file sudah ada, maka isinya akan dihapus dan digantikan dengan pesan dari mode write.
c. “a” (Append) : Fungsinya untuk menambahkan isi di akhir file.
d. “r+” (Read and Write) : Fungsinya untuk membaca dan menulis.

Contoh-contoh pengaksesan file dalam python :

a. Open file

```
with open("mbox.txt", “r”) as file :
    isi = file.read()
    print(isi)
```

b. Menulis ke file

```
with open("mbox.txt", "w") as file :
    file.write("hello ges")
```

c. Menambahkan file

```
with open("mbox.txt", "a") as file :
    file.write("\nini baris baru")
```

d. Membaca isi file

```
with open("mbox.txt", "r") as file :
    isi = file.read()
    print(isi)
```

```
    baris = file.readline()
    print(baris)
```

e. Mengedit isi file

```
with open("mbox.txt", "r") as file :
    ganti_kata = file.read().replace("Hello", "Hi")
    with open("mbox.txt", "w") as file2 :
        file2.write(ganti_kata)
```

f. Menutup file

```
file.close()
```

Manipulasi File

Untuk memanipulasi file dalam python, pertama kita harus siapkan file, kemudian open file. Setelah itu kita harus melakukan perulangan pada setiap baris di file. Dan terakhir close file. Cara membaca file :

```
handle = open('mbox-short.txt')
count = 0
for line in handle:
    count = count + 1
    print('Line Count:', count) #Line Count = 1910
```

#Harus dibaca baris-perbaris untuk mencegah crash saat membuka file yang ukurannya besar

Cara menampilkan ukuran text file dalam bytes dengan menggunakan fungsi len() :

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

Kode tersebut menghitung jumlah huruf dalam satu file. Jika satu karakter dihitung sebagai satu byte maka bisa dibilang kode ini menghitung jumlah byte dalam file. Kode tersebut juga menampilkan string dari huruf paling belakang maju 16 huruf kedepan.

#Tidak disarankan menggunakan perintah read() pada file berukuran besar, karena perintah read() boros memory. Lebih baik jika menggunakan perulangan atau looping.

Contoh manipulasi file dengan perulangan :

```
handle = open('mbox-short.txt')
count = 1
for line in handle:
    if line.startswith("Date:") and count <= 20:
```

```
        count += 1
        print(line)
```

Program tersebut menghasilkan output 20 baris yang berawalan “Date:” atau yang tanggal saja.

Penyimpanan File

Dalam Python cara untuk menulis ke file adalah sama dengan cara membuka (open) file, hanya perlu mengubah metodenya saja yang tadinya “r” menjadi “w” sebagai berikut: handle=open(“output.txt”,’w’) Untuk menuliskan isi string ke dalam file output.txt langsung saja digunakan perintah write() dan jangan lupa tutup file dengan close(). Contohnya :

```
handle = open('output.txt','w')
isi = "teks ini akan dituliskan ke file\n"
handle.write(isi)
handle.close()
```

Kegiatan Praktikum

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.

Source Code

```
filename = input("nama file: ") #Meminta user menginput nama file
handle = open(filename) #membuka file
c = 0 #counter untuk jumlah domain berakhiran ac.uk
for line in handle: #perulangan
    if line.find("ac.uk") != -1: #mencari baris yang mengandung domain ac.uk
        c += 1
        print("Web domain 'ac.uk' ditemukan di \"" + line.strip() + "\"") #output
print("Jumlah: ",c) #menampilkan jumlah baris yang mengandung domain ac.uk
```

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

Source Code

```
filename = input("nama file: ")
try: #untuk penanganan error
    handle = open(filename) #membuka file
    total = 0 #counter untuk ukuran file
    for line in handle: #perulangan
        total += len(line) #total akan bertambah seiring ditemukannya setiap
baris
```

```
        kb = total / 1000 #menghitung total kb
        print("Ukuran: " + str(kb) + " KB") #output
except: #penanganan error jika nama file tidak ditemukan
    print("File tiidak ditemukan!")
```

BAGIAN 2: LATIHAN MANDIRI (60%)

SOAL 1

Souce Code

```
def perbandingan_file(f1, f2) :
    with open(f1, "r") as file1, open(f2, "r") as file2:
        for i, (line1, line2) in enumerate(zip(file1, file2), start = 1) :
            if not line1 and not line2 :
                break
            if line1!= line2:
                print(f"Perbedaan ada pada baris {i}:")
                print(f"file 1: {line1.strip()}")
                print(f"file 2: {line2.strip()}")

file1 = input("masukkan nama file pertama = ")
file2 = input("masukkan nama file kedua = ")
perbandingan_file(file1, file2)
```

Output

```
masukkan nama file pertama = tes1.txt
masukkan nama file kedua = tes2.txt
Perbedaan ada pada baris 1:
file 1: hallo ges
file 2: hello world
Perbedaan ada pada baris 2:
file 1: malam
file 2: siang
```

Penjelasan

Pertama kita buka file dengan fungsi ‘with’, kemudian kita lakukan perulangan yang dikombinasi dengan fungsi ‘**enumerate(zip(file1, file2), start = 1)**’ yang berfungsi untuk mendapatkan elemen dan indeks dari beberapa daftar secara bersamaan dan dimulai atau distart indeksnya dari satu. Selama perulangan, **‘if not line1 and not line2’** selalu mengecek apakah barisnya kosong atau tidak, jika kosong maka program akan dihentikan.

Selama perulangan, **‘if line1!= line2’** juga mengecek baris yang berbeda. Jika berbeda, maka akan ditampilkan letak baris yang berbeda, dan isi baris yang berbeda dari kedua file.

Pada ketiga baris paling bawah, kita hanya meminta input dari user tentang nama kedua file yang ingin dibandingkan, dan memanggil fungsi perbandingan_file.

SOAL 2

Source Code

```
def soal(file_soal) :
    with open(file_soal, "r") as file:
        soal = file.readlines()

        for line in soal :
            q, a = line.strip().split("||")
            a = a.strip().split()
            print(q)
            jawab = input("jawab : ")
            jawab = jawab.strip().split()
            if jawab.lower() == a.lower() :
                print("BENAR")
            else :
                print("SALAH")

inp = input("nama file : ")
soal(inp)
```

Output

```
nama file : soal.txt
1+1 =
jawab : 2
BENAR
Bendera Indonesia?
jawab : merah putih
jawab : yogyakarta
BENAR
Komponen PC untuk penyimpanan file adalah...
jawab : harddisk
BENAR
50 * 20 =
jawab : 1000
BENAR
```

Penjelasan

Seperti biasa kita buka file dengan fungsi ‘with open()’. Kita hanya membaca filenya saja jadi kita gunakan mode read (‘r’). Kemudian kita gunakan **‘soal = file.readlines()’** untuk membaca file per baris. Selanjutnya kita gunakan **‘q, a = line.strip().split("||")’** untuk memisahkan soal dengan jawaban dengan pemisah “||”.

Kemudian kita minta jawaban dari user (yang sudah distrip dan dijadikan huruf kecil) dan dicocokan dengan jawaban dari file (yang sudah distrip dan dijadikan huruf kecil). Jika jawaban user sama dengan

jawaban dari file maka akan ditampilkan output “BENAR’ dan sebaliknya jika jawaba user berbeda dengan jawaban dari file maka akan ditampilkan output “SALAH”.

Pada baris terakhir kita hanya meminta nama file dari user dan memanggil fungsi.

Link Github

https://github.com/JonathanAndrianto123/laporan_alpro9_71230978.git