MODUL 10

FILE dan DIREKTORI

A. Tujuan

1. Memahami teknik mengakses dan mengubah file
2. Memahami teknik mengakses dan mengubah direktori

B. Pendahuluan

Membaca dan menulis file adalah teknik dasar yang harus dipahami dalam pemrograman Python, karena banyak digunakan untuk pengolahan dan pemrosesan file. Memahami cara membaca dan menulis file dengan Python akan membuat kita mampu membuat aplikasi yang bisa mengambil dan menyimpan data ke file. Selain itu juga, kita akan lebih mudah memahami beberapa materi Python lanjutan, seperti baca dan *parsing* file JSON, XML, CSV, XLS, dan sebagainya.

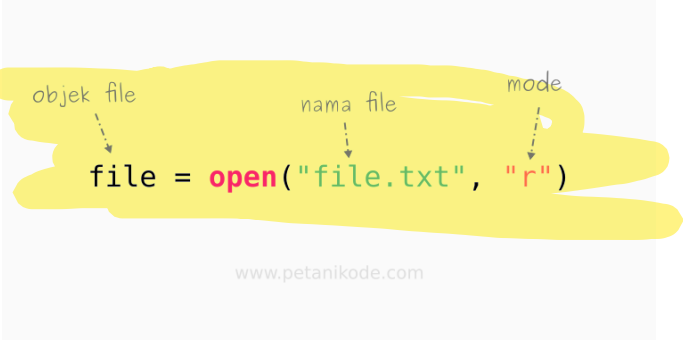
Ada banyak sekali tipe file pada komputer, seperti dokumen, video, gambar, audio, arsip, dll. Pada Python, file hanya dikelompokkan menjadi dua tipe:

1. **File Teks**: File yang berisi teks. Setiap baris teks memiliki EOL (End of Line). Contoh: TXT, MD, CSV, JSON, dsb.
2. **File Binary**: File yang bukan teks, hanya bisa diproses oleh program tertentu yang memahami strukturnya. Contohnya: EXE, JPG, MKV, M4A, 3GP, dsb.

Pada modul ini kita hanya akan belajar cara membaca dan menulis file teks saja.

C. File di Python

Python sudah menyediakan fungsi Open () untuk membaca dan menulis file, Fungsi tersebut memiliki 2 parameter, yaitu nama file dan mode.



Gambar 10.1 Penulisan Kode untuk Membuka File

Objek file adalah variabel objek yang menampung isi file. Kita bisa melakukan pemrosesan file dari variabel tersebut.

Nama file bisa kita isi langsung apabila file-nya terletak dalam satu direktori dengan skrip python. Namun, apabila terletak di direktori yang berbeda, maka kita harus memberikan alamat path file-nya.

```
obj_file = open("/path/ke/file.txt","r")
```

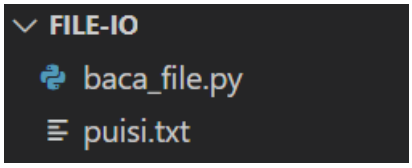
Kemudian untuk parameter mode yang fungsinya untuk menentukan hak akses terhadap file. Ada beberapa mode yang tersedia:

Tabel 10.1 Parameter Mode

Mode	Keterangan
"r"	Hanya baca
"w"	Akses menulis file, jika file sudah ada, maka file akan di replace dan diganti dengan yang baru ditulis
"a"	Digunakan untuk *append* atau menambah data ke file, artinya jika sudah ada data dalam file, amaka akan ditambahkan dan tidak di-replace
"r+"	Digunakan untuk membaca sekaligus menulis data ke file

Example 10.1

Terdapat direktori bernama 'file-io', lalu didalamnya terdapat file 'puisi.txt' dan 'baca_file.py'.



Didalam file 'puisi.txt' terdapat isi teks sebagai berikut:

```
Engkau bahkan bukan lagi variabel bagiku
Engkau adalah konstanta abadi yang tak tergantikan...
```

1. Membaca File per Baris

Didalam file ‘baca_file.py’ terdapat kode program sebagai berikut:

```
# buka file
file_puisi = open("puisi.txt", "r")
# baca isi file
print(file_puisi.readlines())
# tutup file
file_puisi.close()
```

Pertama, kita membuka file dengan fungsi open(), selanjutnya membaca isinya per baris dengan readlines(), dan terakhir menutup file dengan close() agar dihapus dari dalam memori.

Saat dieksekusi, kode di atas akan menghasilkan output berupa *list*, karena kita menggunakan metode readlines().

```
PS C:\Users\MSI-GAMMING\Documents\vscode\file-io> & "C:/Program Files/Python39/python.exe" c:/Users/MSI-GAMMING/Documents/vscode/file-io
/baca_file.py
['Engkau bahkan bukan lagi variabel bagiku\n', 'Engkau adalah konstanta abadi yang tak tergantikan...']
```

2. Membaca Semua Teks dalam File

Kalau tadi kita membaca isi file per baris, sekarang kita coba baca semua teks menggunakan method read (). Berikut adalah kode yang akan dibuat:

```
# buka file
file_puisi = open("puisi.txt", "r")
# baca isi file
puisi = file_puisi.read()
# cetak isi file
print(puisi)
# tutup file
file_puisi.close()
```

Hasilnya adalah sebagai berikut:

```
PS C:\Users\MSI-GAMMING\Documents\vscode\file-io> & "C:/Program Files/Python39/python.exe" c:/Users/MSI-GAMMING/Documents/vscode/file-io
/baca_file.py
Engkau bahkan bukan lagi variabel bagiku
Engkau adalah konstanta abadi yang tak tergantikan...
```

“Apa perbedaan metode read() dengan readlines()?”

Perbedaannya yaitu read() membaca seluruh teks dan akan mengembalikan nilai string. Sedangkan readlines() membaca isi file per baris dan akan mengembalikan nilai berupa *list*. Perlu diketahui, metode read() dan readlines() hanya sekali pakai. Artinya, hanya dieksekusi sekali saja. Eksekusi pertama akan mengembalikan nilai berdasarkan isi filenya. Eksekusi kedua akan mengembalikan nilai kosong.

3. Cara Menulis File di Python

Seperti yang sudah kita ketahui, ada tiga mode yang digunakan bila ingin menulis file, yaitu ‘w’,’a’, dan ‘r+’.

a. Menulis dengan mode ‘w’

Kita membuat kode program sebagai berikut:

```
print("Selamat Datang di Program Biodata")
print("---------------------------------")

# Mengambil input dari user
nama = input("Nama: ")
umur = input("Umur: ")
alamat = input("Alamat: ")

# Format teks
teks = "Nama: {}\nUmur: {}\nAlamat: {}".format(nama,umur,alamat)

# Buka file untuk ditulis
file_bio = open("biodata.txt","w")

# Tulis teks ke file
file_bio.write(teks)

# Tutup file
file_bio.close()
```

Eksekusi program yang dibuat

```
PS C:\Users\MSI-GAMMING\Documents\vscode\file-io> & "C:/Program Files/Python39/python.exe" c:/Users/MSI-GAMMING/Documents/v
scode/file-io/write.py
Selamat Datang di Program Biodata
---------------------------------
Nama: Hasea
Umur: 19
Alamat: Bumi
```

Maka sekarang kita puya file baru bernama ‘biodata.txt’, file tersebut akan terbuat secara otomatis.

b. Menyisipkan data ke File

Apabila kita tidak ingin menulis ulang atau menindih file yang sudah ada, kita bisa menggunakan mode ‘a’ (*append*) untuk menulisnya.

```
print("Selamat Datang di Program Biodata")
print("---------------------------------")

# Mengambil input dari user
nama = input("Nama: ")
umur = input("Umur: ")
alamat = input("Alamat: ")

# Format teks
teks = "\nNama: {}\nUmur: {}\nAlamat: {}".format(nama,umur,alamat)

# Buka file untuk ditulis
file_bio = open("biodata.txt","a")

# Tulis teks ke file
file_bio.write(teks)

# Tutup file
file_bio.close()
```

Kemudian eksekusi dengan biodata yang berbeda:

```
PS C:\Users\MSI-GAMMING\Documents\vscode\file-io> & "C:/Program Files/Python39/python.exe" c:/Users/MSI-GAMMING/Documents/v
scode/file-io/write.py
Selamat Datang di Program Biodata
---------------------------------
Nama: Banu
Umur: 20
Alamat: Bumi
```

Setelah itu, lihat file ‘biodata.txt’

```
Nama: Hasea
Umur: 19
Alamat: Bumi
Nama: Banu
Umur: 20
Alamat: Bumi
```

D. Direktori pada Python

1. Membuat Direktori

Dalam Python, pembuatan direktori baru dilakukan dengan fungsi mkdir() dari library ‘os’. Bentuk umum penggunaannya adalah: os.mkdir(‘<nama direktori>’). Berikut adalah contoh yang menunjukkan penggunaan fungsi mkdir().

```
import os

def main():
    os.mkdir("unit")

ifname:"main"
main()
```

Jika program dijalankan, maka akan ada direktori baru bernama ‘unit’.

2. Mengubah Nama Direktori

Untuk mengubah nama direktori, kita perlu menggunakan fungsi rename(). Bentuk umum penggunaannya: os.rename.(‘<namaDirektori>’, ’<namaDirektoriBaru>’). Berikut adalah contoh yang menunjukkan penggunaan fungsi rename().

```
import os

def main():
    os.rename("unit","modul")

ifname:"main"
main()
```

Jika program dijalankan, maka direktori ‘unit’ berubah menjadi ‘modul.

3. Menghapus Direktori

Untuk menghapus nama direktori, kita perlu menggunakan fungsi rmdir(). Bentuk umum penggunaannya: os.rmdir(‘<namaDirektori>’). Berikut adalah contoh yang menunjukkan penggunaan fungsi rmdir().

```
import os

def main():
    os.rmdir("modul")

ifname:"main"
main()
```

Jika program dijalankan, maka direktori ‘modul’ akan terhapus.