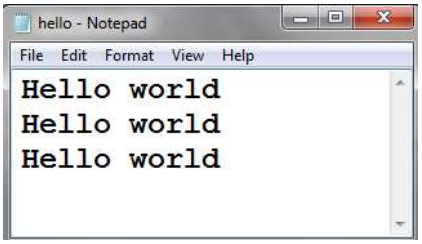
Berikut ini contoh program untuk membuat file dengan nama 'hello.txt' yang diletakkan di direktori D: . File tersebut berisi beberapa string di dalamnya.

```
myfile = open('d:\hello.txt', 'w')
myfile.write('Hello world\n')
myfile.write('Hello world\n')
myfile.write('Hello world\n')
myfile.close()
```

Perhatikan contoh program tersebut. Variabel `myfile` merupakan nama variabel yang menunjukkan pointer file yang akan dibuka.

Karena kita akan membuat file baru dan mau menuliskan beberapa isi di dalamnya, maka digunakan mode `w` (*write*). Selanjutnya untuk menuliskan isi ke dalam file, digunakan perintah `write()`. Tambahan karakter `\n` digunakan supaya isi file yaitu '`Hello World`' sebanyak tiga buah bisa tersusun ke bawah (ingat kembali topik tentang *escape characters* di Bab 13). Setelah file berhasil dibuat, jangan lupa untuk menutup file dengan memberikan perintah `close()` pada `myfile`. Apa yang akan terjadi jika file tidak ditutup setelah dilakukan pemrosesan? File yang tidak ditutup terkadang bisa menjadi *corrupted* atau strukturnya rusak.

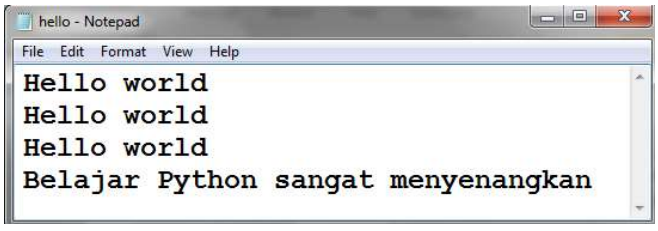
Hasil dari program tersebut akan diperoleh sebuah file bernama hello.txt yang berisi beberapa string dengan tampilan berikut ini (jika dibuka dengan teks editor seperti Notepad atau sejenisnya).



Selanjutnya akan diberikan contoh untuk melakukan proses appending yaitu menambah isi file. Dalam contoh ini, kita akan mencoba menambahkan beberapa baris baru ke dalam file hello.txt yang telah dibuat sebelumnya. Perhatikan program berikut ini

```
myfile = open('d:\hello.txt', 'a')
myfile.write('Belajar Python sangat menyenangkan\n')
myfile.close()
```

Apabila program tersebut dijalankan, maka akan dihasilkan tampilan isi dari file hello.txt sebagai berikut



Dari tampilan hasil program setelah dijalankan dapat kita simpulkan bahwa mode *append* (`a`) akan menambahkan isi ke dalam file teks, tanpa menimpa semua isi file yang ada sebelumnya. Hal ini yang

menjadi perbedaan antara mode *append* (`a`) dengan *write* (`w`). Jika mode yang digunakan adalah write (w), maka jika sebelumnya sudah ada isinya maka isi yang lama akan ditimpa dengan yang baru atau dengan kata lain isi file sebelumnya akan hilang dan diganti dengan isi yang baru.

Contoh berikutnya adalah bagaimana cara membaca isi dari sebuah file teks. Berikut ini adalah contoh bagaimana membaca isi dari file hello.txt yang sudah berhasil kita buat dengan program sebelumnya. Isi dari file hello.txt ini akan dibaca oleh program kemudian ditampilkan isinya. Perhatikan contoh program berikut ini.

```
myfile = open('d:\hello.txt', 'r')
teks = myfile.read()
print(teks)
myfile.close()
```

```
Hello world
Hello world
Hello world
Belajar Python sangat menyenangkan
```

Karena kita ingin membuat program yang bisa untuk membaca isi file, maka digunakan mode read (r) pada `open()`. Selanjutnya untuk membaca isi file, digunakan `read()`. Pada program tersebut, hasil pembacaan disimpan ke dalam variabel bernama teks yang selanjutnya nanti akan dicetak ke layar.

Berdasarkan contoh program yang baru saja diberikan, dapat disimpulkan bahwa perintah `read()` langsung membaca semua isi dari file. Pertanyaan berikutnya adalah, bagaimana jika kita hanya ingin beberapa karakter dari depan saja, misalkan 10 karakter awal dari isi file? Caranya adalah dengan memberikan banyaknya karakter yang ingin dibaca tersebut ke dalam `read()`. Sehingga untuk membaca 10 karakter terdepan dari isi file digunakan perintah `read(10)`.

Untuk membaca secara keseluruhan isi file, dapat pula kita gunakan looping WHILE. Dengan looping tersebut kita baca karakter per karakter secara satu per satu sampai selesai. Berikut ini contoh programnya:

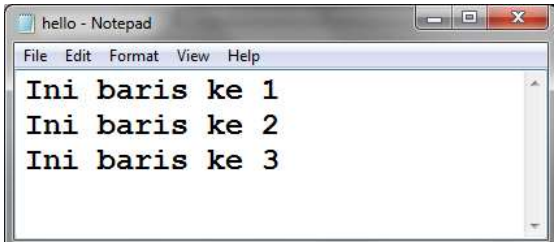
```
myfile = open('d:\hello.txt', 'r')

while True:
    # baca karakter satu per satu
    char = myfile.read(1)
    # jika sudah tidak ada yg dibaca maka looping selesai
    if not char:
        break
    # mencetak karakter per karakter
    print(char, end='')

myfile.close()
```

```
Hello world
Hello world
Hello world
Belajar Python sangat menyenangkan
```

Berdasarkan program yang diberikan, pembacaan karakter demi karakter menggunakan `read(1)`. Proses pembacaan akan selesai apabila sudah tidak ada lagi karakter yang dibaca. Untuk mengidentifikasi tidak adanya karakter yang dibaca lagi, di sini menggunakan statement `if not char`.

Sedangkan untuk menampilkan tiap karakter yang sudah dibaca, digunakan `print(char, end='')` supaya tiap karakter yang dicetak bisa berlanjut posisi di sebelah karakter yang dicetak sebelumnya. Apabila tanpa menggunakan parameter end='' maka setiap karakter yang dicetak akan tampil di bawah karakter sebelumnya.

Selain menggunakan `read()`, Python juga terdapat function untuk membaca isi file baris per baris yaitu menggunakan `readlines()`. Adapun hasil dari `readlines()` berupa list yang isinya adalah teks tiap baris file. Sebagai contoh misalkan di dalam file hello.txt berisi teks sebagai berikut:



Selanjutnya apabila kita tulis program berikut ini

```
myfile = open('d:\hello.txt', 'r')

line = myfile.readlines()
print(line)

myfile.close()
```

```
['Hello world\n', 'Hello world\n', 'Hello world\n', 'Belaj
ar Python sangat menyenangkan\n']
```

Dalam hal ini, isi dari variabel line adalah berupa list. Isi list tersebut merupakan teks pada tiap baris yang merupakan isi file hello.txt. Jika diperhatikan pada isi list hasil `readlines()` maka terdapat karakter \n di setiap akhir barisnya. Karakter \n ini merupakan *escape character* yang bermakna ganti baris (*new line*).

Dari contoh-contoh program yang sudah diberikan sebelumnya, kesemuanya hanya menggunakan satu jenis mode open saja. Bagaimana jika mode yang digunakan bersifat *dual mode*, misalnya membaca dan *append* (`a+`) ? Bagaimana contoh programnya. Perhatikan program berikut ini

```
myfile = open('d:\hello.txt', 'a+')

# menambahkan isi ke dalam file
myfile.write('I love Python\n')

# memposisikan kembali pointer file ke awal file
myfile.seek(0, 0)

#membaca isi file dan menampilkan
isi = myfile.read()
print(isi)

myfile.close()
```

```
Hello world
Hello world
Hello world
Belajar Python sangat menyenangkan
I love Python
```

Program yang diberikan tersebut, bermaksud untuk menambahkan teks 'I love Python' ke dalam file hello.txt yang sudah ada isinya sebelumnya. Selain menambahkan isi file, program juga akan membaca isi file setelah penambahan isinya dan selanjutnya akan dicetak ke layar. Dikarenakan ada dua proses yang akan dilakukan, yaitu append dan membaca file maka digunakan mode `a+` (*append and read*).

Perlu diketahui bahwa setelah proses *append* selesai, pointer file berada di paling akhir isi file. Sehingga apabila dilakukan proses pembacaan isi file, maka akan berakibat tidak ada data yang bisa dibaca (*blank*). Oleh karena itu diperlukan proses reposisi pointer ke awal file sebelum membaca isi menggunakan perintah `seek(0, 0)`. Tanpa adanya perintah `seek(0, 0)` ini hasil pembacaan isi file akan kosong.

Mengubah Nama File

Python juga dilengkapi dengan adanya function untuk mengubah nama file yang ada di direktori komputer. Untuk mengubah nama file ini, digunakan function:

```
os.rename(nama file lama, nama file baru)
```

Namun, untuk bisa menjalankan perintah os.rename() perlu dilakukan import modul os. Perhatikan contoh berikut ini:

```
import os
os.rename('D:\hello.txt', 'D:\hello2.txt')
```

Perintah tersebut dimaksudkan untuk mengubah nama file hello.txt menjadi hello2.txt yang ada di direktori d:

Menghapus File

Sebuah file dalam direktoripun dapat juga dihapus melalui program Python yaitu menggunakan perintah:

```
os.remove(nama file)
```

Adapun contohnya adalah sebagai berikut:

```
import os
os.remove('D:\hello2.txt')
```

Program yang diberikan tersebut digunakan untuk menghapus file hello.txt yang terletak di root direktori d:

Membuat Direktori

Tidak hanya membuat file, Python juga mampu untuk membuat direktori baru dengan menggunakan perintah

```
os.mkdir(nama direktori)
```

Contoh penggunaannya dalam program adalah:

```
import os
os.mkdir('d:\mydir')
```

Program yang dicontoh tersebut digunakan untuk membuat direktori dengan nama 'mydir' di dalam root direktori d:

Di sistem operasi tertentu, misalnya Linux terdapat mode tertentu pada direktorinya. Secara default, mode direktori yang dibuat dengan `mkdir()` adalah 777. Bagaimana jika ingin membuat direktori dengan mode yang lain, misalnya 755?. Caranya adalah menambahkan mode pada parameter ke duanya. Misalnya:

```
import os
os.mkdir('d:\mydir', 0755)
```

Menghapus Direktori

Sebuah direktori dapat pula dihapus melalui Python menggunakan perintah

```
os.rmdir(nama direktori)
```

Contoh berikut ini digunakan untuk menghapus direktori bernama 'mydir' yang letaknya di root direktori d:

```
import os
os.rmdir('d:\mydir')
```