Python File I/O

Unit input adalah (masukan) suatu data yang berbentuk documet bait itu data data foto, data data hurup ataupun data tanggal ke dalam sebuah system yang terkomputerisasi.

unit output (keluaran) biasanya digunakan untuk menampilkan data, atau dengan kata lain untuk menangkap data yang telah diinputkan terlebih dahulu dalam sebuah penyimpanan media elektronic , contohnya data yang akan ditampilkan pada layar monitor atau printer.

I/O Input/Ouput Read [IOR] dan untuk tulis I/O Input/Output Write [IOW].

File adalah lokasi bernama pada disk untuk menyimpan informasi terkait. Ini digunakan untuk menyimpan data secara permanen dalam memori non-volatile (misalnya hard disk).

Karena, random access memory (RAM) bersifat volatile sehingga kehilangan datanya saat komputer dimatikan, kita menggunakan file untuk penggunaan data masa depan.

Bila kita ingin membaca dari atau menulis ke file kita perlu membukanya terlebih dahulu. Bila sudah selesai, perlu ditutup, agar sumber yang diikat dengan file tersebut dibebaskan.

Oleh karena itu, dengan Python, sebuah operasi file berlangsung dengan urutan sebagai berikut.

1. Buka file
2. Membaca atau menulis (melakukan operasi)
3. Tutup file tersebu

Membuka sebuah file

Python memiliki built-in function open () untuk membuka file. Fungsi ini mengembalikan objek file, juga disebut handle, karena digunakan untuk membaca atau memodifikasi file yang sesuai.

»> f = open("test.txt") # open file in current directory
»> f = open("C:/Python33/README.txt") # specifying full path

Kita bisa menentukan mode saat membuka file. Dalam mode, kami menentukan apakah kita ingin membaca ’r’, menulis ’w’ atau menambahkan ’a’ ke file. Kita juga menentukan apakah kita ingin membuka file dalam mode teks atau mode biner.

Defaultnya adalah membaca dalam mode teks. Dalam mode ini, kita mendapatkan string saat membaca dari file.

Di sisi lain, mode biner mengembalikan byte dan ini adalah mode yang akan digunakan saat berhadapan dengan file non-teks seperti file gambar atau exe.

’r’ Buka file untuk dibaca. (default)

’w’ Buka file untuk menulis. Membuat file baru jika tidak ada atau memotong

file jika file tersebut ada.

’x’ Buka file untuk pembuatan eksklusif. Jika file sudah ada, operasi gagal

’a’ Buka untuk menambahkan di akhir file tanpa memotongnya. Membuat file baru jika tidak ada.

’t’ Buka dalam mode teks. (default)

’b’ Buka dalam mode biner.

’+’ Buka file untuk mengupdate (membaca dan menulis)

```
f = open("test.txt")    # equivalent to ’r’ or ’rt’
f = open("test.txt",’w’) # write in text mode
f = open("img.bmp",’r+b’) # read and write in binary mode
```

Tidak seperti bahasa lain, karakter ’a’ tidak menyiratkan angka 97 sampai dikodekan menggunakan ASCII (atau pengkodean setara lainnya).

Apalagi, pengkodean default bergantung pada platform. Di jendela, itu adalah ’cp1252’ tapi ’utf-8’ di Linux.

Jadi, kita juga tidak harus bergantung pada pengkodean default atau kode kita akan berperilaku berbeda di berbagai platform.

Oleh karena itu, ketika bekerja dengan file dalam mode teks, sangat disarankan untuk menentukan jenis pengkodean.

```
f = open("test.txt",mode = ’r’,encoding = ’utf-8’)
```

Menutup sebuah File

Ketika kita selesai dengan operasi ke file, kita perlu menutupinya dengan benar.

Menutup file akan membebaskan sumber daya yang terkait dengan file dan dilakukan dengan menggunakan metode close ().

Python memiliki pengumpul sampah untuk membersihkan benda yang tidak difermentasi tapi, kita tidak boleh bergantung padanya untuk menutup file.

```
f = open("test.txt",encoding = ’utf-8’)
# perform file operations
f.close()
```

Metode ini tidak sepenuhnya aman. Jika pengecualian terjadi saat kita melakukan operasi dengan file, kode keluar tanpa menutup file.

Cara yang lebih aman adalah dengan menggunakan try ... akhirnya blok.

```
try:
```

```
  f = open("test.txt",encoding = 'utf-8')
  # perform file operations
finally:
  f.close()
```

Dengan cara ini, kita dijamin bahwa file tersebut benar tertutup bahkan jika pengecualian dinaikkan, menyebabkan aliran program berhenti.

Cara terbaik untuk melakukannya adalah dengan menggunakan pernyataan. Ini memastikan file ditutup saat blok di dalam dengan keluar.

Kita tidak perlu secara eksplisit memanggil metode close (). Hal itu dilakukan secara internal.

```
with open("test.txt",encoding = 'utf-8') as f:
  # perform file operations
```

Menulis ke File

Untuk menulis ke file kita perlu membukanya dalam mode write 'w', tambahkan 'a' atau exclusive creation 'x'.

Kita harus berhati-hati dengan mode 'w' karena akan menimpa file jika sudah ada. Semua data sebelumnya terhapus.

Menulis string atau urutan byte (untuk file biner) dilakukan dengan menggunakan metode write (). Metode ini mengembalikan jumlah karakter yang ditulis ke file.

```
with open("test.txt",'w',encoding = 'utf-8') as f:
  f.write("my first file \n")
  f.write("This file \n \n")
  f.write("contains three lines \n")
```

Program ini akan membuat file baru bernama 'test.txt' jika tidak ada. Jika memang ada, itu akan ditimpa.

Kita harus menyertakan karakter newline sendiri untuk membedakan garis yang berbeda.
Membaca Dari File

Untuk membaca isi sebuah file, kita harus membuka file dalam mode baca.

Ada berbagai metode yang tersedia untuk tujuan ini. Kita bisa menggunakan metode read (size) untuk membaca dalam jumlah ukuran data. Jika parameter ukuran tidak ditentukan, bunyinya dan kembali ke akhir file.

```
»> f = open("test.txt",'r',encoding = 'utf-8')
»> f.read(4)   # read the first 4 data
```

’This’

»> f.read(4) # read the next 4 data
’ is ’

»> f.read() # read in the rest till end of file
’my first file \nThis file \ncontains three lines \n’

»> f.read() # further reading returns empty sting

Kita dapat melihat, metode read () mengembalikan baris baru sebagai ’ \ n’. Begitu akhir file tercapai, kita mendapatkan string kosong untuk dibaca lebih lanjut.

Kita bisa mengubah kursor file kita saat ini (posisi) dengan menggunakan metode seek (). Demikian pula metode tell () mengembalikan posisi kita saat ini (dalam jumlah byte).

»> f.tell() # get the current file position
56

»> f.seek(0) # bring file cursor to initial position
0

»> print(f.read()) # read the entire file
This is my first file
This file
contains three lines

Kita bisa membaca file line-by-line menggunakan for loop. Ini efisien dan cepat.

»> for line in f:
... print(line, end = ”)
...
This is my first file
This file
contains three lines

Baris dalam file itu sendiri memiliki karakter baris baru ’ \ n’.

Terlebih lagi, print () parameter akhir untuk menghindari dua baris baru saat mencetak.

Sebagai alternatif, kita dapat menggunakan metode readline () untuk membaca setiap baris file. Metode ini membaca sebuah file sampai newline, termasuk newline character.

»> f.readline()
’This is my first file \n’

»> f.readline()
’This file \n’

»> f.readline()
’contains three lines \n’

»> f.readline()
”

Terakhir, metode readlines () mengembalikan daftar baris yang tersisa dari keseluruhan file. Semua metode membaca ini mengembalikan nilai kosong saat akhir file (EOF) tercapai.

»> f.readlines()
[’This is my first file \n’, ’This file \n’, ’contains three lines \n’]

Metode Berkas Python

Ada berbagai metode yang tersedia dengan objek file. Beberapa di antaranya telah digunakan pada contoh di atas.

Berikut adalah daftar lengkap metode dalam mode teks dengan deskripsi singkat.

Atribut Objek file

Setelah file dibuka dan Anda memiliki satu file objek, Anda bisa mendapatkan berbagai informasi yang berkaitan dengan file tersebut.

Berikut adalah daftar semua atribut yang terkait dengan objek file:

#!/usr/bin/python

# Open a file
fo = open("foo.txt", "wb")
print "Name of the file: ", fo.name
print "Closed or not : ", fo.closed
print "Opening mode : ", fo.mode
print "Softspace flag : ", fo.softspace

Membaca dan Menulis File

Objek file menyediakan seperangkat metode akses untuk membuat hidup kita lebih mudah. Kita akan melihat bagaimana menggunakan metode read () dan write () untuk membaca dan menulis file.
Metode tulis ()

Metode write () menulis string apapun ke file yang terbuka. Penting untuk dicatat bahwa string Python dapat memiliki data biner dan bukan hanya teks.

Metode write () tidak menambahkan karakter baris baru (’ \ n’) ke akhir string -
Sintaksis

```
#!/usr/bin/python

# Open a file
fo = open("foo.txt", "wb")
fo.write( "Python is a great language. \nYeah its great!! \n");

# Close opend file
fo.close()
#!/usr/bin/python

# Open a file
fo = open("foo.txt", "r+")
str = fo.read(10);
print "Read String is : ", str

# Close opend file
fo.close()
```

```
Read String is : Python is
```

```
#!/usr/bin/python

# Open a file
fo = open("foo.txt", "r+")
str = fo.read(10);
print "Read String is : ", str

# Check current position
position = fo.tell();
print "Current file position : ", position

# Reposition pointer at the beginning once again
position = fo.seek(0, 0);
str = fo.read(10);
print "Again read String is : ", str
# Close opend file
fo.close()
```

```
Read String is : Python is
Current file position : 10
Again read String is : Python is
```

```
os.rename(current _file _name, new _file _name)

#!/usr/bin/python
import os

# Rename a file from test1.txt to test2.txt
os.rename( "test1.txt", "test2.txt" )
```

```
#!/usr/bin/python
import os

# Delete file test2.txt
os.remove("text2.txt")
```

```
os.mkdir("newdir")
```

```
#!/usr/bin/python
import os

# Create a directory "test"
os.mkdir("test")
```

```
#!/usr/bin/python
import os

# Changing a directory to "/home/newdir"
os.chdir("/home/newdir")
```

```
#!/usr/bin/python
import os

# This would give location of the current directory
os.getcwd()
```

```
#!/usr/bin/python
import os

# This would remove "/tmp/test" directory.
os.rmdir( "/tmp/test" )
```

Membuka sebuah file

Untuk membuka file teks yang Anda gunakan, well, open () function. Sepertinya masuk akal. Anda melewatkan parameter tertentu untuk membuka () untuk memberitahukannya di mana file harus dibuka - 'r' untuk dibaca saja, 'w' untuk tulisan saja (jika ada file lama, akan dituliskan), 'a 'Untuk menambahkan (menambahkan sesuatu ke akhir file) dan' r + 'untuk membaca dan menulis. Tapi kurang bicara, mari kita buka file untuk membaca (Anda bisa melakukan ini dengan mode idle python Anda). Buka file teks biasa. Kami kemudian akan mencetak apa yang kita baca di dalam file:

Contoh Kode 1 - Membuka sebuah file

```
Openfile = open ('pathtofile', 'r')
Openfile.read ()
```

Itu menarik. Anda akan melihat banyak simbol ' \ n'. Ini merupakan baris baru (di mana Anda menekan enter untuk memulai baris baru). Teksnya benar-benar tidak diformat, tapi

jika Anda melewati keluaran openfile.read () untuk mencetak (dengan mengetikkan print openfile.read ()) akan diformat dengan baik.

Carilah dan Anda Temukan

Apakah Anda mencoba mengetik di print openfile.read ()? Apakah itu gagal? Kemungkinan besar, dan alasannya adalah karena 'kursor' telah mengubah tempatnya. Kursor Kursor apa Nah, kursor yang sebenarnya tidak bisa kamu lihat, tapi tetap kursor. Kursor tak terlihat ini memberitahukan fungsi baca (dan banyak fungsi I / O lainnya) dari mana mulai. Untuk mengatur di mana kursor berada, Anda menggunakan fungsi seek (). Ini digunakan dalam bentuk seek (offset, dari mana).

Mana yang opsional, dan menentukan mana yang harus dicari. Jika darimana 0, byte / huruf dihitung dari awal. Jika 1, byte dihitung dari posisi kursor saat ini. Jika 2, maka byte dihitung dari akhir file. Jika tidak ada yang diletakkan di sana, 0 diasumsikan.

Offset menggambarkan seberapa jauh dari mana kursor bergerak. sebagai contoh:

openfile.seek (45,0) akan memindahkan kursor ke 45 byte / huruf setelah permulaan file.

Openfile.seek (10,1) akan memindahkan kursor ke 10 byte / huruf setelah posisi kursor saat ini.

openfile.seek (-77,2) akan memindahkan kursor ke 77 byte / huruf sebelum akhir file (perhatikan - sebelum angka 77)

Cobalah sekarang juga. Gunakan openfile.seek () untuk pergi ke tempat manapun di file dan kemudian mencoba mengetik print openfile.read (). Ini akan dicetak dari tempat yang Anda inginkan. Tapi sadari bahwa openfile.read () memindahkan kursor ke akhir file - Anda harus mencari lagi.

Fungsi I / O Lainnya

Ada banyak fungsi lain yang membantu Anda dalam berurusan dengan file. Mereka memiliki banyak kegunaan yang memberdayakan Anda untuk berbuat lebih banyak, dan membuat hal-hal yang dapat Anda lakukan lebih mudah. Mari kita lihat kirim (), readline (), readlines (), tulis () dan close ().

Kirim () mengembalikan tempat kursor berada dalam file. Tidak memiliki parameter, ketik saja (seperti contoh di bawah ini yang akan ditampilkan). Ini sangat berguna, untuk mengetahui apa yang Anda maksud, di mana letaknya, dan kontrol kursor yang sederhana. Untuk menggunakannya, ketik fileobjectname.tell () - dimana fileobjectname adalah nama dari file objek yang Anda buat saat Anda membuka file (di openfile = open ('pathtofile', 'r') nama objek file adalah openfile).

Readline () membaca dari mana kursor sampai akhir baris. Ingat bahwa akhir baris bukan tepi layar Anda - garis berakhir saat Anda menekan enter untuk membuat baris baru. Ini berguna untuk hal-hal seperti membaca log peristiwa, atau mengalami sesuatu yang progresif untuk mengolahnya. Tidak ada parameter yang harus Anda lewati ke readline (), meskipun secara opsional Anda dapat memberi tahu jumlah maksimal byte / huruf untuk dibaca dengan meletakkan nomor di tanda kurung. Gunakan dengan fileobjectname.readline

().

Readlines () sama seperti readline (), namun readlines () membaca semua baris dari kursor dan seterusnya, dan mengembalikan sebuah daftar, dengan setiap elemen daftar memegang satu baris kode. Gunakan dengan fileobjectname.readlines (). Misalnya, jika Anda memiliki file teks:

Contoh Kode 2 - contoh file teks

Baris 1

Baris 3
Baris 4

Baris 6

Fungsi write (), menulis ke file. Bagaimana kamu menebak nya??? Ini menulis dari mana kursor berada, dan menimpa teks di depannya - seperti di MS Word, di mana Anda menekan 'insert' dan menulis di atas teks lama. Untuk memanfaatkan fungsi yang paling penting ini, letakkan string di antara tanda kurung untuk ditulis mis. fileobjectname.write ('ini adalah string').

Dekat, Anda bisa mencari, menutup file sehingga Anda tidak dapat lagi membaca atau menulis sampai Anda membuka kembali lagi. Cukup sederhana Untuk menggunakan, Anda akan menulis fileobjectname.close (). Sederhana!

Dengan mode siaga Python, buka file uji (atau buat yang baru ...) dan mainkan fungsi ini. Anda bisa melakukan penyuntingan teks yang sederhana (dan sangat merepotkan).

Mmm, acar

Pickles, dengan Python, harus dimakan. Rasa mereka hanya untuk membiarkan pemrogram meninggalkan mereka di lemari es.

Ok, hanya bercanda disana. Pickles, dengan Python, adalah objek yang disimpan ke sebuah file. Objek dalam kasus ini bisa berupa variabel, instance dari kelas, atau daftar, kamus, atau tupel. Hal lain juga bisa acar, tapi dengan batas. Objek kemudian dapat dipulihkan, atau tidak dicemari, nanti. Dengan kata lain, Anda 'menyimpan' benda Anda.

Jadi bagaimana kita acar? Dengan fungsi dump (), yang ada di dalam modul acar - jadi pada awal program Anda, Anda harus menulis acar impor. Cukup sederhana Kemudian buka file kosong, dan gunakan pickle.dump () untuk menjatuhkan objek ke file itu. Mari kita coba itu:

Contoh Kode 3 - pickletest.py

pickletest.py
PICKLE AN OBJECT

mengimpor modul acar
Acar impor

```
# Mari membuat sesuatu untuk menjadi acar
# Bagaimana dengan daftar?
picklelist = ['one', 2, 'three', 'four', 5, 'dapatkah kamu menghitung?']

# Sekarang buat file
# ganti nama file dengan file yang ingin Anda buat
File = open ('filename', 'w')

# Sekarang mari kita pilih picklelist
pickle.dump (picklelist, file)

# Tutup file, dan pengawetan Anda sudah selesai
file.close ()
```

Kode untuk melakukan ini diletakkan seperti pickle.load (object _to _pickle, file _object) di mana:

object _to _pickle adalah objek yang ingin Anda acar (yaitu simpan ke file)
file _object adalah objek file yang ingin Anda tulis (dalam kasus ini, objek file adalah 'file')

Setelah Anda menutup file tersebut, buka di notepad dan lihat apa yang Anda lihat. Seiring dengan beberapa kue gibblygook lainnya, Anda akan melihat potongan daftar yang kami buat.

Sekarang untuk membuka kembali, atau unpickle, file Anda. Untuk menggunakan ini, kita akan menggunakan pickle.load ():

Contoh kode 4 - unpickletest.py

```
# # # unpickletest.py
# # # unpickle file

# mengimpor modul acar
Acar impor

# sekarang buka file untuk dibaca
# ganti nama file dengan path ke file yang Anda buat di pickletest.py
unpicklefile = open ('filename', 'r')

# sekarang muat daftar yang kita acar ke objek baru
unpickledlist = pickle.load (unpicklefile)

# Tutup file, hanya untuk keamanan
unpicklefile.close ()

# Mencoba menggunakan daftar
untuk item dalam unpickledlist:
   Item cetak
```