

CURS 6

Fișiere text

Un *fișier text* este o secvență de caractere organizată pe linii și stocată în memoria externă (SSD, HDD, DVD, CD etc.). Practic, fișierele text reprezintă o modalitate foarte simplă prin care se poate asigura *persistența datelor*.

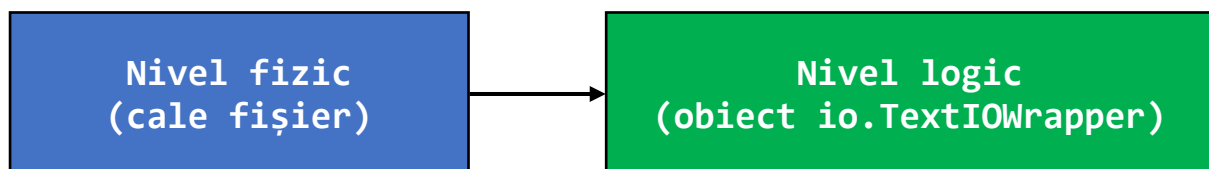
Liniile dintr-un fișier text sunt demarcate folosind caracterele **CR** (*carriage return* sau `'\r'` sau `chr(13)`) și **LF** (*line feed* sau `'\n'` sau `chr(10)`), astfel:

- în sistemele de operare *Windows* și *MS-DOS* se utilizează combinația **CR+LF**;
- în sistemele de operare de tip *Unix/Linux* și începând cu versiunea *Mac OS X* se utilizează doar caracterul **LF**;
- în sistemele de operare până la versiunea *Mac OS X* se utilizează doar caracterul **CR**.

Indiferent de modalitatea utilizată pentru demarcarea liniilor într-un anumit sistem de operare, în momentul în care se citesc linii dintr-un fișier text, caracterele utilizate pentru demarcare vor fi transformate automat într-un singur caracter LF (`'\n'`)!

Deschiderea unui fișier text

Operația de deschidere a unui fișier (text sau binar!) presupune asocierea fișierului, considerat la nivel fizic (i.e., identificat prin calea sa), cu o variabilă/un obiect dintr-un program, ceea ce va permite manipularea sa la nivel logic (i.e., prin intermediul unor funcții sau metode dedicate):



Practic, după deschiderea unui fișier, obiectul asociat fișierului va conține mai multe informații despre starea fișierului respectiv (dimensiunea în octeți, modalitatea de codificare utilizată, poziția curentă a pointerului de fișier etc.), iar programatorul va putea acționa asupra fișierului într-un mod transparent, folosind metodele puse la dispoziție de obiectul respectiv.

Deschiderea unui fișier se realizează folosind următoarea funcție (reamintim faptul că parantezele drepte indică faptul că parametrul este opțional):

```
open("cale fișier", ["mod de deschidere"])
```

Primul parametru al funcției reprezintă calea fișierului, sub forma unui șir de caractere, iar în cazul în care fișierul se află în directorul curent se poate indica doar numele său. Modul de deschidere este utilizat pentru a preciza cum va fi prelucrat fișierul text și se indică prin următoarele litere:

- **"r"** pentru *citire din fișier* (read)
 - fișierul va putea fi utilizat doar pentru a citi date din el;
 - este modul implicit de deschidere a unui fișier text, i.e. se va folosi dacă se omite parametrul "mod de deschidere" în apelul funcției open;
 - dacă fișierul indicat prin calea respectivă nu există, atunci se va genera eroarea `FileNotFoundError`;
- **"w"** pentru *scriere în fișier* (write)
 - fișierul va putea fi utilizat doar pentru a scrie date în el;
 - dacă fișierul indicat prin calea respectivă nu există, atunci se va crea unul nou, altfel fișierul existent va fi șters automat și va fi înlocuit cu unul nou;
- **"x"** pentru *scriere în fișier creat în mod exclusiv* (exclusive write)
 - fișierul va putea fi utilizat doar pentru a scrie date în el;
 - dacă fișierul indicat prin calea respectivă nu există, atunci se va crea unul nou, altfel se va genera eroarea `FileExistsError`;
- **"a"** pentru *adăugare în fișier* (append)
 - fișierul va putea fi utilizat doar pentru a scrie informații în el;
 - dacă fișierul există, atunci noile datele se vor scrie imediat după ultimul său caracter, altfel se va crea un fișier nou și datele se vor scrie de la începutul său.

Pentru a trata excepțiile care pot să apară în momentul deschiderii unui fișier, se va folosi o instrucțiune de tipul `try...except`, așa cum se poate observa în exemplele următoare:

```
try:
    f = open("exemplu.txt")
    print("Fișier deschis cu succes!")
except FileNotFoundError:
    print("Fișier inexistent!")
```

```
try:
    f = open("C:\Test Python\exemplu.txt", "x")
except FileNotFoundError:
    print("Calea fișierului este greșită!")
except FileExistsError:
    print("Fișierul deja există!!")
```

Din cel de-al doilea exemplu se poate observa faptul că eroarea `FileNotFoundError` va fi generată și în cazul modurilor de deschidere pentru scriere (i.e., modurile "w", "x" și "a") în cazul în care calea fișierului este greșită!

Citirea datelor dintr-un fișier text

În limbajul Python, datele citite dintr-un fișier text vor fi furnizate întotdeauna sub forma unor șiruri de caractere!

Pentru citirea datelor dintr-un fișier text sunt definite următoarele metode:

- **read():** furnizează tot conținutul fișierului text într-un singur șir de caractere

Exemplu:

Program	Fișier de intrare	Ecran
<pre>f = open("exemplu.txt") s = f.read() print(s) f.close()</pre>	<p>Ana are multe pere,</p> <p>mere rosii,</p> <p>si mai multe prune!!!</p>	<p>Ana are multe pere,</p> <p>mere rosii,</p> <p>si mai multe prune!!!</p>

Observați faptul că șirul `s` conține și caracterele LF (`'\n'`) folosite pentru delimitarea liniilor!

- **readline():** furnizează conținutul liniei curente din fișierul text într-un șir de caractere sau un șir vid când se ajunge la sfârșitul fișierului

Exemplu:

Program	Fișier de intrare	Ecran
<pre>f = open("exemplu.txt") s = f.readline() while s != "": print(s, end="") s = f.readline() f.close()</pre>	<p>Ana are multe pere,</p> <p>mere rosii,</p> <p>si mai multe prune!!!</p>	<p>Ana are multe pere,</p> <p>mere rosii,</p> <p>si mai multe prune!!!</p>

Observați faptul că, de fiecare dată, șirul `s` conține și caracterul LF (`'\n'`) folosit pentru delimitarea liniilor, mai puțin în cazul ultimei linii!

O modalitate echivalentă de parcurgere a unui fișier text linie cu linie constă în iterarea directă a obiectului `f` asociat:

Exemplu:

Program	Fișier de intrare	Ecran
<pre>f = open("exemplu.txt") for linie in f: print(linie, end="") f.close()</pre>	<p>Ana are multe pere,</p> <p>mere rosii,</p> <p>si mai multe prune!!!</p>	<p>Ana are multe pere,</p> <p>mere rosii,</p> <p>si mai multe prune!!!</p>

- **readlines():** furnizează toate liniile din fișierul text într-o listă de șiruri de caractere

Exemplu:

Program	Fișier de intrare	Ecran
<pre>f = open("exemplu.txt") s = f.readlines() print(s) f.close()</pre>	<pre>Ana are multe pere, mere rosii, si mai multe prune!!!</pre>	<pre>['Ana are multe pere,\n', '\n', 'mere rosii,\n', 'si mai multe prune!!!']</pre>

Observați faptul că fiecare șir de caractere din listă, corespunzător unei linii din fișier, conține și caracterul LF (' \n ') folosit pentru delimitarea liniilor, mai puțin în cazul ultimei linii!

Cele 3 modalități de citire a datelor dintr-un fișier text sunt echivalente, dar, în cazul în care dimensiunea fișierului este mare, se preferă citirea linie cu linie a conținutului său, deoarece necesită mai puțină memorie.

Deoarece toate metodele folosite pentru citirea datelor dintr-un fișier text furnizează șiruri de caractere, dacă vrem să citim datele respective sub forma unor numere trebuie să utilizăm funcții pentru manipularea șirurilor de caractere.

Exemplu: Fișierul text *numere.txt* conține, pe mai multe linii, numere întregi separate între ele prin spații. Scrieți un program care afișează pe ecran suma numerelor din fișier.

Soluția 1 (folosind metoda read):

Construim o listă *numere* care să conțină numerele din fișier, împărțind șirul corespunzător întregului conținut al fișierului în subșiruri cu metoda *split* și transformând fiecare subșir într-un număr cu metoda *str*:

```
f = open("numere.txt")
numere = [int(x) for x in f.read().split()]
print("Suma numerelor din fisier:", sum(numere))
f.close()
```

Soluția 2 (folosind metoda readline):

Construim tot o listă care să conțină toate numerele din fișier, dar împărțind în subșiruri doar șirul corespunzător liniei curente:

```
f = open("numere.txt")
numere = []
linie = f.readline()
while linie != "":
    numere.extend([int(x) for x in linie.split()])
    linie = f.readline()
print("Suma numerelor din fisier:", sum(numere))
f.close()
```

Evident, o soluție mai eficientă din punct de vedere al memoriei utilizate constă în calculul sumei numerelor de pe fiecare linie și adunarea sa la suma tuturor numerelor din fișier:

```
f = open("numere.txt")
total = 0
linie = f.readline()
while linie != "":
    numere = [int(x) for x in linie.split()]
    total = total + sum(numere)
    linie = f.readline()
print("Suma numerelor din fisier:", total)
f.close()
```

Soluția 3 (folosind metoda readlines):

Vom proceda într-un mod asemănător cu soluția 2, dar parcurgând lista cu liniile fișierului furnizată de metoda readlines:

```
f = open("numere.txt")
total = 0
for linie in f.readlines():
    numere = [int(x) for x in linie.split()]
    total = total + sum(numere)
print("Suma numerelor din fisier:", total)
f.close()
```

Atenție, în oricare dintre soluțiile prezentate, metoda split trebuie apelată fără parametri, pentru a împărți șirul respectiv în subșiruri considerând toate spațiile albe (care includ caracterele '\n' și '\r')! Altfel, dacă am apela metoda split doar cu parametrul " " (un spațiu) ar fi generată eroarea ValueError în momentul în care am încerca să transformăm în număr ultimul subșir de pe o linie (mai puțin ultima din fișier), deoarece acesta ar conține și caracterul '\n'!

Scrierea datelor într-un fișier text

În limbajul Python, într-un fișier text se pot scrie doar șiruri de caractere. Pentru scrierea datelor într-un fișier text sunt definite următoarele metode:

- **write(șir)**: scrie șirul respectiv în fișier, fără a adăuga automat o linie nouă

Exemplu:

Program	Fișier de ieșire
<pre>f = open("exemplu.txt", "w") cuvinte = ["Ana", "are", "mere!"] for cuv in cuvinte: f.write(cuv) f.close()</pre>	Anaaremere!

Dacă dorim să scriem fiecare cuvânt pe o linie nouă, atunci trebuie să modificăm `f.write(cuv)` în `f.write(cuv + "\n")`.

- **writelines(colecție de șiruri)**: scrie toate șirurile din colecție în fișier, fără a adăuga automat linii noi între ele

Exemplu:

Program	Fișier de ieșire
<pre>f = open("exemplu.txt", "w") cuvinte = ["Ana", "are", "mere!"] f.writelines(cuvinte) f.close()</pre>	Anaaremere!

Dacă dorim să scriem fiecare cuvânt pe o linie nouă, atunci fie trebuie să adăugăm câte un caracter `"\n"` la sfârșitul fiecărui cuvânt din listă, fie să modificăm `f.writelines(cuvinte)` în `f.writelines("\n".join(cuvinte))`, deci să concatenăm toate șirurile din listă într-unul singur, folosind caracterul `"\n"` ca separator.

Închiderea unui fișier text

Indiferent de modalitatea în care a fost deschis un fișier și, implicit, accesat conținutul său, acesta trebuie închis după terminarea prelucrării sale, folosind metoda `close()`. Închiderea unui fișier constă în golirea eventualei zone tampon (buffer) alocate fișierului și ștergerea din memorie a obiectului asociat cu el. Dacă un fișier deschis într-unul dintre modurile de scriere nu este închis, atunci există riscul ca ultimele informații scrise în fișier să nu fie salvate! În mod implicit, un fișier este închis dacă referința obiectului asociat este atribuită unui alt fișier (i.e., se utilizează, din nou, funcția `open`).

Putem elimina necesitatea închiderii explicite a unui fișier putem folosi o instrucțiune `with...as`, astfel:

```
with open("exemplu.txt") as f:
    s = f.readlines()
    print(f.closed)      #False

print(f.closed)         #True
print(s)
```

În acest caz, fișierul va fi închis automat imediat după ce se va termina prelucrarea sa, așa cum se poate observa din exemplul de mai sus (data membră `closed` permite testarea stării curente a unui fișier, respectiv dacă acesta a fost închis sau nu). Mai mult, închiderea fișierului respectiv se va efectua corect chiar și în cazul apariției unei erori în timpul prelucrării sale. Atenție, chiar dacă se utilizează o instrucțiune

with...as, erorile care pot să apară în momentul deschiderii unui fișier trebuie să fie tratate explicit!

În încheiere, vom prezenta un exemplu în care vom utiliza toate cele 3 moduri de deschidere ale unui fișier text.

Mai întâi, vom implementa un program care să genereze un fișier text care va conține pe prima linie o sumă formată din n numere naturale aleatorii (de exemplu, 12 + 300 + 9 + 1234):

```
import random

numef = input("Numele fisierului: ")
n = int(input("Numarul de valori aleatorii: "))

with open(numef, "w") as f:
    # initializam generatorul de numere aleatorii
    random.seed()
    for k in range(n - 1):
        # generam un numar aleatoriu
        # cuprins intre 1 si 1000
        x = random.randint(1, 1000)
        f.write(str(x) + " + ")
    f.write(str(random.randint(1, 1000)))
```

În continuare, vom implementa un program care să calculeze suma numerelor din fișier și apoi să o scrie la sfârșitul primei linii, după un semn "=" (de exemplu, 12 + 300 + 9 + 1234 = 1555):

```
numef = input("Numele fisierului: ")

# deschidem fisierul pentru citire si
# calculam suma numerelor de pe prima linie
f = open(numef, "r")
nr = [int(x) for x in f.readline().split("+")]
s = sum(nr)

# deschidem fisierul pentru adaugare
# si scriem suma la sfarsitul primei linii

# fisierul deschis anterior pentru citire
# va fi inchis implicit
f = open(numef, "a")
f.write(" = " + str(s))
# inchidem explicit fisierul
f.close()
```