CURS 6
Fişiere text

Un *fişier text* este o secvenţă de caractere organizată pe linii şi stocată în memoria externă (SSD, HDD, DVD, CD etc.). Practic, fişierele text reprezintă o modalitate foarte simplă prin care se poate asigura *persistenţa datelor*.

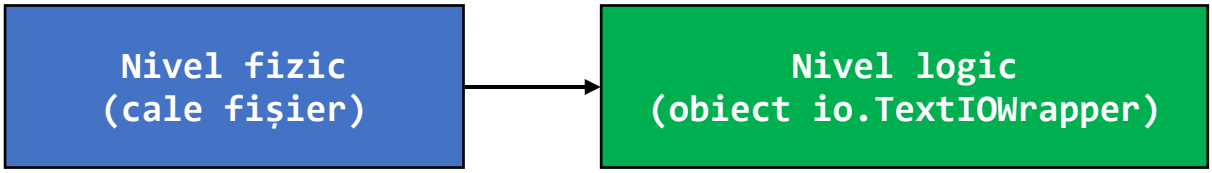
Liniile dintr-un fişier text sunt demarcate folosind caracterele **CR** (*carriage return* sau `'\r'` sau `chr(13)`) şi **LF** (*line feed* sau `'\n'` sau `chr(10)`), astfel:

- în sistemele de operare *Windows* şi *MS-DOS* se utilizează combinaţia **CR+LF**;
- în sistemele de operare de tip *Unix/Linux* şi 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 fişier text, caracterele utilizate pentru demarcare vor fi transformate automat într-un singur caracter `LF ('\n')`!

Deschiderea unui fişier text

Operaţia de deschidere a unui fişier (text sau binar!) presupune asocierea fişierului, 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 funcţii sau metode dedicate):



Practic, după deschiderea unui fişier, obiectul asociat fişierului va conţine mai multe informaţii despre starea fişierul respectiv (dimensiunea în octeţi, modalitatea de codificare utilizată, poziţia curentă a pointerului de fişier etc.), iar programatorul va putea acţiona asupra fişierului într-un mod transparent, folosind metodele puse la dispoziţie de obiectul respectiv.

Deschiderea unui fişier se realizează folosind următoarea funcţie (reamintim faptul că parantezele drepte indică faptul că parametrul este opţional):

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

Primul parametru al funcţiei reprezintă calea fişierului, sub forma unui şir de caractere, iar în cazul în care fişierul se află în directorul curent se poate indica doar numele său.
Modul de deschidere este utilizat pentru a preciza cum va fi prelucrat fişierul text şi 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 fişier text

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

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

- **`read()`**: furnizează tot conţinutul fişierului text într-un singur şir de caractere

  Exemplu:

Program	Fişier de intrare	Ecran

`f = open("exemplu.txt")`
`s = f.read()`
`print(s)`
`f.close()`
	`Ana are multe pere,`
	
	`mere rosii,`
	`si mai multe prune!!!`
		`Ana are multe pere,`
		
		`mere rosii,`
		`si mai multe prune!!!`

  Observaţi faptul că şirul `s` conţine şi caracterele LF (`'\n'`) folosite pentru delimitarea liniilor!

- **`readline()`**: furnizează conţinutul liniei curente din fişierul text într-un şir de caractere sau un şir vid când se ajunge la sfârşitul fişierului

  Exemplu:

Program	Fişier de intrare	Ecran

`f = open("exemplu.txt")`
`s = f.readline()`
`while s != "":`
`  print(s, end="")`
`  s = f.readline()`
`f.close()`
	`Ana are multe pere,`
	
	`mere rosii,`
	`si mai multe prune!!!`
		`Ana are multe pere,`
		
		`mere rosii,`
		`si mai multe prune!!!`

  Observaţi faptul că, de fiecare dată, şirul `s` conţine şi caracterul LF (`'\n'`) folosit pentru delimitarea liniilor, mai puţin în cazul ultimei linii!

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

  Exemplu:

Program	Fişier de intrare	Ecran

`f = open("exemplu.txt")`
`for linie in f:`
`  print(linie, end="")`
`f.close()`
	`Ana are multe pere,`
	
	`mere rosii,`
	`si mai multe prune!!!`
		`Ana are multe pere,`
		
		`mere rosii,`
		`si mai multe prune!!!`

- **`readlines()`**: furnizează toate liniile din fişierul text într-o listă de şiruri de caractere

Exemplu:

Program	Fişier de intrare	Ecran

`f = open("exemplu.txt")`
`s = f.readlines()`
`print(s)`
`f.close()`
	`Ana are multe pere,`
	
	`mere rosii,`
	`si mai multe prune!!!`
		`['Ana are multe pere,\n', '\n', 'mere rosii,\n', 'si mai multe prune!!!']`

Observaţi faptul că fiecare şir de caractere din listă, corespunzător unei linii din fişier, conţine şi caracterul LF (`'\n'`) folosit pentru delimitarea liniilor, mai puţin în cazul ultimei linii!

Cele 3 modalităţi de citire a datelor dintr-un fişier text sunt echivalente, dar, în cazul în care dimensiunea fişierului este mare, se preferă citirea linie cu linie a conţinutului său, deoarece necesită mai puţină memorie.

Deoarece toate metodele folosite pentru citirea datelor dintr-un fişier text furnizează şiruri de caractere, dacă vrem să citim datele respective sub forma unor numere trebuie să utilizăm funcţii pentru manipularea şirurilor de caractere.

Exemplu: Fişierul text *numere.txt* conţine, pe mai multe linii, numere întregi separate între ele prin spaţii. Scrieţi un program care afişează pe ecran suma numerelor din fişier.

Soluţia 1 (folosind metoda `read`):
Construim o listă `numere` care să conţină numerele din fişier, împărţind şirul corespunzător întregului conţinut al fişierului în subşiruri cu metoda `split` şi transformând fiecare subşir 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()
```

Soluţia 2 (folosind metoda `readline`):
Construim tot o listă care să conţină toate numerele din fişier, dar împărţind în subşiruri doar şirul 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

`f = open("exemplu.txt", "w")`
`cuvinte = ["Ana", "are", "mere!"]`
`for cuv in cuvinte:`
`    f.write(cuv)`
`f.close()`
	`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 şirurile din colecţie în fişier, fără a adăuga automat linii noi între ele

Exemplu:

Program	Fișier de ieșire

`f = open("exemplu.txt", "w")`
`cuvinte = ["Ana", "are", "mere!"]`
`f.writelines(cuvinte)`
`f.close()`
	`Anaaremere!`

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

Închiderea unui fişier text

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

Putem elimina necesitatea închiderii explicite a unui fişier putem folosi o instrucţiune `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, fişierul va fi închis automat imediat după ce se va termina prelucrarea sa, aşa cum se poate observa din exemplul de mai sus (data membră `closed` permite testarea stării curente a unui fişier, respectiv dacă acesta a fost închis sau nu). Mai mult, închiderea fişierului respectiv se va efectua corect chiar şi în cazul apariţiei unei erori în timpul prelucrării sale. Atenţie, chiar dacă se utilizează o instrucţiune

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

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

Mai întâi, vom implementa un program care să genereze un fişier text care va conţine 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 fişier şi apoi să o scrie la sfârşitul 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()
```