INFORMATICĂ · PYTHON · CLASA A IX-A · MEMORAREA DATELOR ȘI ORGANIZAREA CODULUI · LECȚIA 13

Fișiere text: citire, scriere și pregătirea datelor

Durată: 100 min Nivel: mate-info · Python Înainte: lecțiile py9-10, py9-11

CE VEI ȘTI DUPĂ LECȚIE

- Explici cele trei etape ale lucrului cu un fișier (deschidere, transfer, închidere) și alegi modul potrivit: "r", "w" sau "a".
- Folosești with open(...) și metodele read, readline, readlines, write ale unui obiect TextIOWrapper.
- Prelucrezi un fișier linie cu linie, pregătind datele cu strip, split și conversii.
- Recunoști și tratezi erorile tipice: fișier inexistent, fișier gol, conversie uitată.

1. De ce fișiere și cum se lucrează cu ele

Ce scrii cu `input()` dispare când se termină programul. Un **fișier text** păstrează datele pe disc: le poți citi de câte ori vrei, poți lucra cu mii de valori fără să le tastezi, iar alt program le poate folosi mai departe.

Lucrul cu un fișier are mereu **trei etape**:

Etapa	În Python	Ce se întâmplă
1. deschiderea	`f = open("note.txt", "r", encoding="utf-8")`	se leagă un obiect de fișierul de pe disc
2. transferul	`f.read()`, `f.readline()`, `f.write(…)`	datele trec din fișier în program sau invers
3. închiderea	`f.close()`	se termină scrierea pe disc și se eliberează fișierul

`open()` întoarce un obiect din clasa **`TextIOWrapper`** (`print(type(f))` afișează `<class '_io.TextIOWrapper'>`). Ca la orice obiect (lecția py9-10), îl folosești prin metodele lui.

Modul	Înseamnă	Dacă fișierul nu există	Dacă există
`"r"` (implicit)	citire	`FileNotFoundError`	se citește de la început
`"w"`	scriere	se creează	**conținutul vechi se șterge**
`"a"`	adăugare	se creează	se scrie la sfârșit

SFAT

Scrie mereu `encoding="utf-8"`. Fără el, pe unele calculatoare (de exemplu pe Windows) diacriticele (ă, î, ș, ț) pot apărea greșit sau pot da erori. Numele fără cale (`"note.txt"`) se caută în **folderul din care rulează programul**.

2. with: închiderea automată

Varianta recomandată este `with`: fișierul se închide singur la ieșirea din bloc, chiar dacă apare o eroare.

```
with open("note.txt", "w", encoding="utf-8") as g:
    g.write("Ana 9\nBogdan 7\nCristina 10\n")
# aici g este deja închis
```

Fișierul `note.txt` are acum trei rânduri. Îl folosim în exemplele următoare.

3. Metodele de citire și „sfârșitul de fișier”

Obiectul fișier ține minte **unde a ajuns** (un cursor). Fiecare citire pornește de acolo și mută cursorul mai departe.

```
with open("note.txt", encoding="utf-8") as f:
    print(repr(f.readline()))      # 'Ana 9\n'
    print(repr(f.readline()))      # 'Bogdan 7\n'
    print(repr(f.read()))          # 'Cristina 10\n'   (restul fișierului)
    print(repr(f.readline()))      # ''                (sfârșitul fișierului)
print(f.closed)                    # True
```

Metoda	Întoarce
`f.read()`	tot restul fișierului, ca un singur `str`
`f.readline()`	următorul rând, **cu** `\n` la final; `''` la sfârșitul fișierului
`f.readlines()`	lista rândurilor rămase: `['Ana 9\n', 'Bogdan 7\n', 'Cristina 10\n']`
`for linie in f:`	parcurge rândurile, unul câte unul

Sfârșitul fișierului se recunoaște după textul gol: `readline()` dă `''`, iar `read()` dă `''`. Un rând gol din mijlocul fișierului nu se confundă cu sfârșitul, pentru că el este `'\n'`, nu `''`.

4. Pregătirea datelor citite: strip, split, conversii

Fișierul conține **text**. Ca să calculezi, tai rândul în bucăți și convertești:

```
with open("note.txt", encoding="utf-8") as f:
    for linie in f:                    # 'Ana 9\n', apoi 'Bogdan 7\n', ...
        nume, nota = linie.split()     # split() elimină și \n-ul
        print(f"{nume:<10}{int(nota):>3}")
```

```
Ana         9
Bogdan      7
Cristina   10
```

Metoda	Exemplu	Rezultat
`strip()`	`" 7\n".strip()`	`'7'` (taie spațiile și `\n` de la margini)
`split()`	`"3 8 5\n".split()`	`['3', '8', '5']` (taie la orice spații, inclusiv `\n`)
`split(",")`	`"Ana,9".split(",")`	`['Ana', '9']` (pentru fișiere CSV)
`int()`, `float()`	`int("12")`	`12`

Lista dintr-o singură linie. Forma `[expresie for x in colecție]` construiește o listă nouă: pentru fiecare `x` din colecție calculează expresia și o adaugă la rezultat. Este scrierea scurtă a unei bucle cu `append`:

```
texte = ["3", "8", "5"]

a = []                                  # varianta cu buclă
for x in texte:
    a.append(int(x))

b = [int(x) for x in texte]             # aceeași listă, într-o linie
c = [int(x) for x in texte if x != "8"] # cu `if` păstrezi doar o parte din elemente
print(a, b, c)                          # [3, 8, 5] [3, 8, 5] [3, 5]
```

Citești de la stânga la dreapta: „`int(x)`, pentru fiecare `x` din `texte`". În această lecție o folosim ca să convertim dintr-un foc toate textele obținute cu `split()`.

Numere puse oricum în fișier. Dacă `numere.txt` conține `3 8 5`, `12`, un rând gol și `7 6`, atunci `f.read().split()` le adună pe toate, indiferent de rânduri:

```
with open("numere.txt") as f:
    valori = [int(x) for x in f.read().split()]
print(valori, sum(valori))          # [3, 8, 5, 12, 7, 6] 41
```

Pentru fișiere foarte mari, citește **rând cu rând** și prelucrează din mers, fără să păstrezi toate valorile. Exemplu: cel mai mare număr par.

```
mx = -1                             # santinelă: numerele sunt naturale
with open("numere.txt") as f:
    for linie in f:
        for t in linie.split():
            x = int(t)
            if x % 2 == 0 and x > mx:
                mx = x
print(mx if mx != -1 else "nu exista")   # 12
```

LA BAC

La examenul de bacalaureat (în varianta C++ de până acum), Subiectul III conține o problemă care citește un șir lung de numere dintr-un fișier, de tipul `bac.txt`, și cere o soluție eficientă. Tiparul de mai sus (o parcurgere, prelucrare din mers, puțină memorie) este exact ce se urmărește.

5. Scrierea: write și print(..., file=...)

`write()` primește **doar text** și **nu** adaugă singur `\n`. `print(…, file=g)` scrie ca pe ecran: convertește singur, pune spații și trece la rând nou.

```
with open("rezultat.txt", "w", encoding="utf-8") as g:
    g.write("Media: ")
    g.write(str(26 / 3))            # write(26 / 3) ar da TypeError
    g.write("\n")
    print("Maxim:", 10, file=g)
with open("rezultat.txt", "a", encoding="utf-8") as g:
    g.write("Linie adăugată\n")
```

Fișierul conține: `Media: 8.666666666666666`, `Maxim: 10`, `Linie adăugată`, pe trei rânduri.

6. Erori tipice și cum le tratezi

Ce faci	Ce se întâmplă
deschizi pentru citire un fișier care nu există	`FileNotFoundError: [Errno 2] No such file or directory: 'lipsa.txt'`
`linie + 1` (ai uitat conversia)	`TypeError: can only concatenate str (not "int") to str`
`int(f.readline())` pe rândul `3 8 5`	`ValueError: invalid literal for int() with base 10: '3 8 5\n'` : trebuie `split()`
`g.write(5)`	`TypeError: write() argument must be str, not int`
citești după ieșirea din `with`	`ValueError: I/O operation on closed file.`
deschizi cu `"w"` un fișier pe care voiai să-l citești	conținutul se pierde, fără niciun mesaj
fișier gol	`read()` și `readline()` dau `''`, `readlines()` dă `[]` : verifică înainte să împarți la numărul de valori

Fișierul inexistent se poate trata cu `try` / `except` (vezi lecția py9-11):

```
try:
    with open("date.txt", encoding="utf-8") as f:
        valori = [int(x) for x in f.read().split()]
except FileNotFoundError:
    print("Nu găsesc fișierul date.txt.")
```

> **DE ȘTIUT**
>
> Pe pbinfo, la problemele cu fișiere, folosești **exact** numele de fișiere din enunț. Evaluarea se face pe Linux, unde `Date.in` și `date.in` sunt fișiere diferite.

7. Verifică-te

1. Care sunt cele trei etape ale lucrului cu un fișier? Ce metodă corespunde fiecăreia?
2. Ce diferență este între modurile `"w"` și `"a"`?
3. Fișierul are rândurile `10`, `20`, `30`. Ce afișează `print(f.readline() + f.readline())`, imediat după deschidere?
4. Cum recunoști sfârșitul fișierului când citești cu `readline()`?
5. Fișierul conține `5 7` pe primul rând. De ce `a, b = f.readline()` nu merge? Scrie varianta corectă.
6. De ce este mai bine `with open(...) as f:` decât `f = open(...)` urmat de `f.close()`?

Răspunsuri

1. Deschiderea (`open`), transferul (`read`, `readline`, `readlines`, `write`), închiderea (`close`, sau automat cu `with`).
2. `"w"` șterge conținutul vechi și scrie de la început; `"a"` păstrează conținutul și adaugă la sfârșit.
3. `10` și `20` pe rânduri separate: textul citit este `'10\n20\n'` (fiecare rând își păstrează `\n`-ul, iar `print` mai adaugă unul).
4. `readline()` întoarce textul gol `''`.
5. `f.readline()` dă textul `'5 7\n'`, care are 4 caractere, nu 2 valori. Corect: `a, b = map(int, f.readline().split())`.
6. Fișierul se închide automat, chiar dacă apare o eroare în bloc, și nu poți uita `close()`.

Probleme de lucru în clasă (pbinfo)

Le rezolvăm împreună în ședință, în ordinea de mai jos.

- [ ] **#1 sum** — ușoară
  https://www.pbinfo.ro/probleme/1/sum
- [ ] **#57 n-suma** — ușoară
  https://www.pbinfo.ro/probleme/57/n-suma
- [ ] **#7 maxim2** — ușoară
  https://www.pbinfo.ro/probleme/7/maxim2

Temă pentru acasă (pbinfo)

Trimite soluțiile pe pbinfo și notează punctajul obținut. Bifează ce ai terminat.

- [ ] **#8 maxim3** — ușoară
  https://www.pbinfo.ro/probleme/8/maxim3

- [ ] #2269 **maximpar** — ușoară
  https://www.pbinfo.ro/probleme/2269/maximpar
- [ ] #3391 **sirmaxim** — ușoară
  https://www.pbinfo.ro/probleme/3391/sirmaxim
