Fișiere text

Fișiere text

Un fișier text este o secvență de caractere organizată pe linii și stocată pe un suport de memorie extern.

Lucrul cu fișiere într-un limbaj de programare presupune următoarele etape:

- deschiderea fișierului - prin care asociem o variabilă unui fișier fizic (identificat prin calea sa), variabilă cu ajutorul căreia putem lucra cu fișierul folosind funcții puse la dispoziție de limbaj
- prelucrarea fișierului, adică citirea și scrierea de date în fișier, permise în funcție de modul în care am deschis fișierul
- închiderea fișierului când am terminat lucrul cu el

Deschiderea unui fișier text

În Python un fișier se poate deschide folosind metoda open:

```
open("cale_fisier"[, "mod_de_deschidere"])
```

care returnează un obiect asociat fișierului fizic cu calea specificată prin parametrul **`cale_fisier`** (dacă fișierul este în directorul de lucru se poate da doar numele fișierului), deschis cu modul de prelucrare specificat în parametrul **`mod_de_deschidere`** (dacă nu se specifică acest parametru, modul implicit de deschidere este **`r`**), unde **`mod_de_deschidere`** poate fi **de exemplu**:

- **`"r"` = read = pentru citire;**
  dacă fișierul indicat în primul parametru nu există, atunci se va genera eroarea **`FileNotFoundError`**
- **`"w"` = write = pentru scriere (suprascrie dacă există deja)**
  dacă fișierul indicat în primul parametru nu există se va crea unul, altfel se va suprascrie cel existent (conținutul celui existent se va pierde)
- **`"a"` = append = pentru scriere la sfârșit (dacă nu există se creează)**
  dacă fișierul indicat în primul parametru nu există, se va crea un fișier nou; datele se vor scrie la fișierului

Obiectul asociat fișierului (de tipul TextIOWrapper) conține mai multe informații despre starea fișierul respectiv.
Metoda open are mai mulți parametri opționali:
https://docs.python.org/3/library/functions.html#open

Exemplu – Pentru a trata și cazul în care un fișier deschis pentru citire nu există (caz în care se va genera eroare **`FileNotFoundError`**) putem folosi instrucțiuni **`try...except`**

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

Un fișier se poate deschide și folosind o instrucțiune **`with... as`**, caz în care fișierul **nu mai trebuie închis explicit:**

```
with open("exemplu.txt") as f:
    print(f.closed)
print(f.closed)
```

La terminarea execuției blocului with..as fișierul se va închide chiar dacă a apărut o eroare în timpul execuției acestui bloc (erorile legate de deschiderea fișierului, de exemplu **`FileNotFoundError`**, trebuie tratate însă explicit).

Închiderea unui fișier text

După ce terminăm lucru cu un fișier, acesta trebuie închis (amintim că dacă fișierul este deschis cu instrucțiunea **`with...as`** se închide automat la terminarea execuției corpului instrucțiunii). Un fișier se poate închide cu metoda **`close`**(). Este importantă închiderea fișierului pentru o bună gestionare a memoriei (! dacă un fișier pentru scriere nu este închis, există riscul ca ultimele informații scrise în fișier să nu fie salvate).

Exemplu:

```
f = open("exemplu.txt")
print("Fișier deschis cu succes!")
f.close()
```

Citirea din fișiere text

Metodele pentru citirea din fișier text permit citirea uneia sau tuturor liniilor din fișier, informațiile fiind **returnate ca șiruri de caractere**. Aceste metode sunt:

- **`f.read()`** – **returnează tot conținutul** fișierul într-un șir de caractere

  Exemplu – Rulați următoarea secvență de cod pentru un fișier de intrare **`date.in`** cu cel puțin două lini:

  ```
  f = open('date.in')
  linie = f.read()
  print(linie)
  print(repr(linie)) #si \n
  f.close()
  ```

- **`f.readline()`** – **returnează** linia curentă într-un șir de caractere sau șirul vid dacă s-a ajuns la finalul fișierului; dacă linia din fișier se termină cu un delimitator (delimitatorul de sfârșit de linie poate depinde de sistemul de operare), șirul returnat conține la final și marcajul de sfârșit de linie transformat în caracterul '**\n' (LF)**

  Exemplu – Rulați următoarea secvență de cod pentru un fișier de intrare **`date.in`** cu **cel puțin două lini**:

  ```
  f = open('date.in')
  linie = f.readline()
  print(linie)
  print(repr(linie)) #include \n daca mai urmeaza o linie dupa in fisier
  f.close()
  ```

- **`l = f.readlines()`** – returnează toate liniile din fișier sub formă de listă de șiruri de caractere, fiecare element din listă fiind o linie din fișier (incluzând și caracterul de sfârșit de linie, mai puțin în cazul ultimei linii)

  Exemplu – Rulați următoarea secvență de cod pentru un fișier de intrare **`date.in`** cu cel puțin două lini:

```
f = open('date.in')
linie = f.readlines()
print(linie)
print(repr(linie))
f.close()
```

Parcurgerea unui fișier text linie cu line

Pentru a citi un fișier linie cu linie (nu toate liniile odată, pentru a nu le memora pe toate simultan), putem apela repetat instrucțiunea **`readline`**(), până când rezultatul întors este șirul vid:

```
#citire linie cu linie
with open('date.in', 'r') as f:
    linie = f.readline()
    while linie != '':
        print(linie, end='') #linie contine \n
        linie = f.readline()
```

Putem itera un fișier linie cu line folosind for:

```
#citire linie cu linie
with open('date.in', 'r') as f:
    for linie in f:
        print(linie, end='') #linie contine \n
```

Scrierea în fișier text

Pentru a scrie într-un fișier text deschis într-unul dintre modurile care permit scrierea, există metodele:

- **`f.write(sir_de_caractere)`** – scrie în fișier șirul de caractere primit ca parametru (și atât, nu și un delimitator de linie)

  Exemplu:

```
f = open('date.out',"w")
x = 5
y = 4
#f.write(x) #TypeError: write() argument must be str, not int
f.write(str(x)) #nu si sfarsit de linie
f.write(str(y))
f.write("\n"+str(x)+" "+str(y)+"\n")
f.write(f"{x} {y}\n")
f.close()
```

- **`f.writelines(colectie_de_siruri)`** – primește ca parametru o colecție de șiruri de caractere și le scrie în fișier (unul după altul, nu adaugă automat sfârșit de linie sau separatori – putem folosi spre exemplu "\n".join(lista_siruri de caractere) pentru a le scrie câte unul pe linie)

  Exemplu

```
with open("date1.out","w") as g:
    g.writelines(["linia 1","linia 2", "3"])
with open("date2.out","w") as g:
    g.writelines(["linia 1\n","linia 2\n", "3"])
with open("date3.out","w") as g:
    ls = ["linia 1","linia 2", "3"]
    g.write("\n".join(ls))
```

Exemple

1. Suma numerelor dintr-un fișier

Se consideră fișierul text **`numere.txt`** care conține, pe mai multe linii, numere naturale separate între ele prin spații. Să se scrie în fișierul **`numere.out`** suma numerelor din fișier.

```
# varianta 1 - fisierul tot citit cu read()
f = open("numere.txt")
s = sum([int(x) for x in f.read().split()])  # caracter alb - si spatiu si \n
f.close()
g = open("numere.out", "w")
g.write(str(s))
g.close()

# varianta 2- citit linie cu linie - mai buna, memorie mai putina
# 2.1 - linie cu linie cu readline()
f = open("numere.txt")
s = 0
linie = f.readline()
while linie != '':
    s += sum([int(x) for x in linie.split()])
    linie = f.readline()
f.close()
g = open("numere.out", "w")
g.write(str(s))
g.close()

# 2.2 - linie cu linie cu iterator
with open("numere.txt") as f, open("numere.out", "w") as g:
    s = 0
    for linie in f:
        s += sum([int(x) for x in linie.split()])
    #s = sum((sum(int(x) for x in linie.split()) for linie in f))
    g.write(str(s))
```

Temă – același enunț, dar în ipoteza că pe fiecare linie este un singur număr, fără a folosi split.

2. Intersecția a două/n mulțimi – fiecare mulțime se dă pe o linie din fișier

Se consideră fișierul text **`multimi.in`** care conține elementele a două mulțimi de numere naturale, elementele unei mulțimi fiind date pe o linie separate prin spații. Să se scrie în fișierul **`multimi.out`** intersecția acestor mulțimi.

```
f = open("multimi.in")
m1 = {int(x) for x in f.readline().split()}
m2 = {int(x) for x in f.readline().split()}
f.close()
g = open("multimi.out", "w")
g.write(" ".join([str(x) for x in m1 & m2]))
g.close()
```

3. Adăugarea unei informații la finalul unui fișier

a) Să se adauge la finalul fișierului **`numere.out`** creat la 1 mesajul " este suma numerelor din fisierul numere.txt"

```
g = open("numere.out", "a")
g.write(" este suma numerelor din fisierul numere.txt")
g.close()
```

b) Modificați soluția de la 1 astfel încât suma să fie scrisa la finalul fișierului **`numere.txt`**, pe o linie nouă

```
f = open("numere.txt")
#s = sum((sum(int(x) for x in linie.split()) for linie in f))
s = 0
for linie in f:
    s += sum([int(x) for x in linie.split()])

f.close()
g = open("numere.txt","a")
g.write("\n"+str(s))
g.close()
```

4. Lista cu numele elevilor dintr-un fișier

În fișierul **`nume.txt`** sunt numele elevilor dintr-o clasă, câte unul pe linie. Să se memoreze numele elevilor din fișier într-o listă.

```
#varianta 1 - cu read() si split dupa \n:
f = open("nume.txt")
ls = f.read().split("\n")
print(ls)
f.close()

#varianta 2 -cu readlines() - fiecare sir are \n la final cu exceptia
ultimului
f = open("nume.txt")
ls = f.readlines()
for i in range(len(ls)-1):
    ls[i] = ls[i].strip("\n")
    #ls[i] = ls[i][:-1]
print(ls)
f.close()
```

```
#varianta 3 - linie cu linie - fiecare linie are \n la final cu exceptia
ultimei linii
f = open("nume.txt")
ls = [linie.strip("\n") for linie in f]
print(ls)
f.close()
```

5. Copierea conținutului unui fișier în alt fișier

```
#Varianta 1 – linie cu linie
with open("date.in", 'r') as f, open("date.out", 'w') as g:
    for linie in f:
        g.write(linie)
#Varianta 2 – citit tot fisierul in memorie
with open("date.in", 'r') as f, open("date.out", 'w') as g:
    linii = f.read()
    g.write(linii)
#Varianta 3
with open("date.in", 'r') as f, open("date.out", 'w') as g:
    linii = f.readlines()
    g.writelines(linii)
```

Fișierele standard de intrare sys.stdin

Fișierul standard de intrare este în modulul **`sys`** și se numește **`stdin`** (fiind tot de tipul **`TextIOWrapper`**), folosind metodele descrise anterior putem citi un șir dat pe mai multe linii de la tastatură (oprire cu CTR+D).

Exemple

1. Citirea unui șir care conține mai multe linii

```
import sys
strofa = sys.stdin.read()
print(strofa)
```

2. Citirea linie cu linie de la tastatură

Varianta 1 – folosind **`input`**()

```
linie = input()
while len(linie)>0:
    print(linie)
    linie = input()
```

Varianta 2 - iterând **`sys.stdin`**

```
for linie in sys.stdin:
    #print(linie,end="")
    sys.stdout.write(linie)
```