Mentor Klub

Python tanfolyam

7. alkalom

Fájlkezelés

Mentor Klub

- Alapvetően két féle fájlról beszélhetünk
  - Szöveges fájl:
    - Azokat a fájlokat, melyeket elejétől végéig dolgozunk fel, és szöveges formában tárolják az adatot, szöveges fájlnak nevezzük.
    - Megnyitás után szabad szemmel is olvasható.
    - *.txt, *.csv, *.py
  - Bináris fájl:
    - A fájl tartalma nem sorfolytonos szöveg, hanem valamilyen alkalmazás specifikus formátum. Önmagában nem olvasható, egy alkalmazás kell a dekódolásához.

Műveletek szöveges fájlokkal

Mentor Klub

- A programozási nyelvek általában 3 módot engedélyez szöveges fájlok kezelésére:
  - Megnyitás olvasásra /read/ (“r”): A programból ami megnyitotta a fájlt, nem lehet módositani a tartalmát
  - Megnyitás irásra /write/ (“w”): A fájlt irásra nyitja meg az alkalmazás, az irás a fájl elején fog kezdődni. Ez azt is jelenti, hogy a már lézető file-t kérdés nélkül felülírja, tartalmát “kinullázza”
  - Megnyitás hozzáfűzésre /append/: A fájlt írásra nyitja meg. Ha nem létezik a file, viselkedése megegyezik a “w” opcióval. Ha a file létezik, akkor az írási művelet a fájl végén fog történni.
- Pythonban mindhárom módnak van egy bővített változata, de azokat nem igazán használjuk. (r+, w+, a+)

Fájl megnyitása/bezárása

- Egy fájl megnyitása az `open("file_nev", "mód")` paranccsal lehetséges.
  - Az eredmény egy változó lesz, a fájlt ezen keresztül lehet elérni.
- Egy már megnyitott fájl bezárása a close() művelettel történik
  - FONTOS: A fájl bezárása garantálja, hogy a kiírt módosítások a háttértárra kerülnek.
- Példa:

```
myfile = open("ez_itt_egy_file.txt", "a")
#Itt csinálunk valamit a fájllal
myfile.close()
```

Olvasás/Írás

- Sikeres megnyitás után, a megnyitási mód függvényében hajthatunk végre műveleteket a fájlon.
- Olvasás a `read(n)`, `readline(n)` vagy a `readlines()` műveletekkel történik
  - Az eredmény mindig szöveges (str) típusú
- Írni a `write(szoveg)` vagy a `writelines(lista)` műveletekkel lehet.

Context manager használata (with)

- Használhatunk úgynevezett context manager-t, azaz a kódban indentálással is kiemelhetjük, hogy mit kell csinálni, amikor a fájl nyitva van.
- Ilyenkor nincs szükség a fájl bezárására, mert a blokkot elhagyva automatikusan bezáródik a fájl.
- Példa

```
with open("egy_file.txt", "a") as myfile:
    myfile.write("Hello világ!")
```

Gyakorló feladatok

- Írjunk programot, mely az `elso_fajlom.txt` fájlba kiírja a 0 és 100 közötti páros számokat. Használjunk függvényt, ahol lehet.

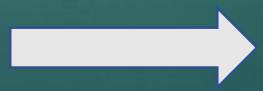
- Írjunk programot, mely az `elso_fajlom.txt` fájl tartalmát beolvassa és kiirja a standard kimenetre. Használjunk függvényt, ahol lehet.

Gyakorló feladatok

- A múltheti gyakorló házi feladatokat valósítsuk meg úgy, hogy a bemenet és a kimenet kezelése is fájlok segítségével történjen.
  - Egy számokat tartalmazó listából keressük meg a legnagyobb hárommal osztható számot.
  - Egy számokat tartalmazó listából keressük meg a legkisebb 5-tel osztható számot és adjuk vissza a lista-beli indexét
  - Egy szavakat tartalmazó listából válasszuk ki azt a szót, amiben a legtöbb, szintén paraméterként átadott betű szerepel.
    - Például a ['cica', 'kutya’, 'kakas’] szavakat tartalmazó listában mely szóban van a legtöbb ‘a’ betű?

Gyakorló- és házi feladatok

- Készíts programot, mely a Super Mario-ból is ismert két oldalú lépcsőt írja ki egy fájlba.
  - A felhasználótól a program indulásakor kérjük be, hogy hány lépcsőfokot szeretne, és azt is, hogy mi legyen a kimeneti fájl neve.
  - Az eredmény fájlt mindíg írjuk felül.
  - A lépcsőfokok a # karakterrel legyenek ábrázolva.
  - Például, ha a 4 fokos lépcsőt kell készíteni, és a kimeneti fájl neve out.txt, akkor az out.txt tartalma:



```
   #  #
  ##  ##
 ###  ###
####  ####
```