1. feladat (fájlkezelés – alap): Sorok szűrése és összesítések

Készíts programot, amely egy `adatok.txt` nevű szövegfájlban szereplő sorokat feldolgozza!
Adatok:

- Bemeneti fájl: `adatok.txt` (UTF-8, tetszőleges, vegyes tartalmú sorok)

A program végezze el:

- Olvassa be a fájl összes sorát `with open(..., encoding="utf-8")` használatával.
- Számolja meg a sorok számát, a nem üres sorok számát, és a teljes karakterek számát (újsorok nélkül).
- Szűrje ki azokat a sorokat, amelyek tartalmaznak legalább egy számjegyet (0–9), és írja ki ezeket egy új fájlba: `szamjegyes.txt`.
- Írjon a képernyőre egy rövid összefoglalót (f-string): sorok száma, nem üres sorok száma, összkarakter.

Programozási elemek:

- `with open(..., encoding="utf-8")`, `strip()`, `any(ch.isdigit() for ch in sor)`
- f-string formázás: pl. `f"Nem ures sorok:  {db} db"`

Kimenet példa:

- Sorok száma: 120, nem üres: 96, összkarakter: 5342
- A `szamjegyes.txt` csak a számjegyet tartalmazó sorokat tartalmazza, eredeti sorrendben.

2. feladat (fájlkezelés – CSV): Csavarok összesítése és rendezése

Olvasd be a `file/csavarok.csv` állományt, és készíts összesítést a méret szerinti darabszámokról!
Adatok:

- Bemeneti CSV: pontosvesszővel tagolt; soronként `Meret;Darab` (pl. `M4;5`)

A program számolja ki:

- Összes darabszám: $\sum$ `Darab`.
- Méretenkénti összegzés (azonos méretek összevonása).
- Rendezett lista: először darabszám szerint csökkenően, azon belül méret szerint növekvően.
- Eredmény kiírása új fájlba: `osszefoglalo.csv` (fejléc: `Meret;Darab`).

Programozási elemek:

- `csv` modul vagy `split(';')` használata
- `dict` az aggregáláshoz, `sorted(..., key=...)` a rendezéshez
- Típuskonverzió: `int(darab)`

Kimenet példa (konzol):

- Összes darab: 54
- Top 3: M4: 18 db, M6: 12 db, M3: 9 db

3. feladat (szövegfeldolgozás): Szógyakoriság és stop-szavak

Készíts szógyakorisági elemzést egy szövegfájlról, és írd ki a leggyakoribb szavakat!

Adatok:

- Bemeneti fájl: `szoveg.txt` (UTF-8)
- Stop-szavak: `stop.txt` – soronként egy szó (kisbetűvel)

Feladat:

- Olvasd be a teljes szöveget, alakítsd kisbetűssé, és távolítsd el az alap írásjeleket (vessző, pont, pontosvessző, kettőspont, felkiáltójel, kérdőjel, zárójelek, idézőjelek).
- Tördeld szavakra, szűrd ki az 1–2 karakteres zajt (len(szo) ≥ 3).
- Szűrd ki a stop-szavakat (ha a szó szerepel a `stop.txt`-ben).
- Számold meg az előfordulásokat és írd ki a `top_20.txt` fájlba a 20 leggyakoribb szót és gyakoriságukat.

Programozási elemek:

- `str.lower()`, `translate()` (vagy karakterenkénti csere)
- `collections.Counter` vagy `dict` alapú számlálás
- Rendezés és szeletelés a top 20-hoz

Kimenet példa:

- `top_20.txt`: soronként `szo;gyakorisag`
- Konzol: Egyedi szavak száma (stop nélkül): 1837

4. feladat (logelemzés): Napló szűrése, csoportosítása és statisztikák

Dolgozz fel egy alkalmazásnaplót és készíts statisztikákat időbélyeg és szint szerint!

Adatok (példa formátum, soronként):

- `2025-09-01 08:15:02 [INFO] Rendszer indult`
- `2025-09-01 08:15:05 [ERROR] Adatbazis kapcsolat megszakadt`

Feladat:

- Olvasd be az `app.log` fájlt; minden sorból parsold ki a dátum-időt (`%Y-%m-%d %H:%M:%S`) és a log-szintet (`INFO/WARNING/ERROR`).
- Szűrd ki az `ERROR` és `WARNING` sorokat külön fájlokba: `errors.log`, `warnings.log`.
- Készíts összesítést: darabszám szintenként; darabszám óránként (00–23), csak az `ERROR` sorokra.
- Írd ki a legrégebbi és legfrissebb bejegyzés időbélyegét.

Programozási elemek:

- `datetime.strptime`, `split()` vagy `re` a formátumhoz
- Szótár alapú csoportosítás, `defaultdict(int)` vagy `Counter`
- f-string-es, táblázatos kiírás (oszlopok igazítása)

Kimenet példa (konzol):

- INFO = 152, WARNING = 17, ERROR = 9
- ERROR óránként: 01: 0, 02: 1, . . . , 14: 3, . . .
- Időtartomány: 2025-09-01 08:15:02 – 2025-09-07 21:44:59

5. feladat (bináris fájl): Rekordok írása/olvasása és ellenőrzés

Tervezd meg egy egyszerű bináris állomány formátumát mérési rekordok tárolására, és készíts író–olvasó programot!

Formátum (rekordonként):

- `id` – 32 bites egész (`int32`)
- `value` – 32 bites lebegőpontos (`float32`)
- `status` – 8 bites egész (`uint8`), 0=OK, 1=WARN, 2=ERR

Feladat:

- Olvasd be a `meresek.txt` fájlt (soronként: `id;value;status`), konvertáld, majd írd ki binárisba: `meresek.bin` (little-endian).
- Olvasd vissza a `meresek.bin`-t, és ellenőrizd rekordról rekordra, hogy az adatok egyeznek.
- Készíts összefoglalót: rekordok száma, `status` szerinti darabszámok, `value` minimum, maximum, átlag.
- Hibakezelés: ha egy sor formátuma hibás, írd a `hibas_sorok.txt`-be az eredeti sort (a program ne álljon le).

Programozási elemek:

- `struct` modul: kis endian formátum: `<ifB`
- Típuskonverziók és kivételkezelés: `try/except ValueError`
- Ellenőrző összeg (opcionális): egyszerű bájtösszeg mod 256 a teljes fájlra

Kimenet példa (konzol):

- Rekordok: 500, OK = 472, WARN = 21, ERR = 7
- `value`: min = 0.002, max = 98.441, átlag = 12.537
- Hibás sorok: 3 (lásd: `hibas_sorok.txt`)