Órai feladatmegoldás

Rendelkezésre álló idő: 75 perc

I. Feladat

Írj programot, amely megkérdezi a felhasználótól egy hónap számát (pl. 3 = március), és utána kiírja, hány napos az a hónap!

A megoldás terve

- A gép nem tudja magától, hogy melyik hónap hány napos, ezt nekünk kell megadni.
- Hozzunk létre egy listát, tároljuk el a hosszakat!

Figyelni kell itt, hogy mindig a **hónap - 1** indexet használjuk a listán, mivel a január, vagyis az 1-es hónap adata a 0. indexű elemben van, tehát **hosszak[hónap - 1] = hónap hossza** módon használhatjuk az adatszerkezetünket
.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
31	28	31	30	31	30	31	31	30	31	30	31

Ezt úgy is meg lehetne oldani, ha a 0. indexű elembe betennénk egy helytartót, pl. egy 0 értéket. Mert akkor a januárhoz tartozó 31-es szám az 1. indexre kerülne, a február 28-a a 2. indexre stb. A példamegoldás nem ilyen.

```
honapok = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
honap = int(input("Kerem a hónap számát: "))
print(f"A kért hónap {honapok[honap-1]} db napból áll.")
```

Írj programot, amelyik egy adott dátumról (év, hónap, nap) kiírja, hogy az év hányadik napja! Az év paraméterre a szökőévek miatt van szükség. Használd fel az előző feladatban megírt függvényed!

Végül tedd be az év napját kiszámító algoritmusod is egy függvénybe! Vagyis az **evnapja(ev, honap, nap)** függvényhívás visszatérési értékében adja meg a nap sorszámát 1 és 365 (szökőév esetén 366) között.

A megoldás terve

- Össze kell adni az eltelt hónapokat, és az adott napot. Például március 5. = 31 (január) + 28 (február) + 5 (ötödike) = az év 64. napja.
- A hónapok hosszait az előző programban már eltároltuk egy listában. Annak az elejét kell majd összegezni.
- Szökőéveknél a február 29 napos. Ezt akkor kell figyelembe venni, ha a február eltelt.

II. Feladat

Írj egy szoveget_elemez nevű függvényt, amely egy szöveget vár paraméterben! A függvény számolja meg, hogy a szövegben mennyi betű, számjegy és egyéb karakter szerepel, majd adja ezt vissza egy dictionary-ben, a példában látható formátumban!

Példa:

```
Input: 'Python4Life!!!'

Return: {'betu': 10, 'szamjegy': 1, 'egyeb': 3}



Input: '12345'

Return: {'betu': 0, 'szamjegy': 5, 'egyeb': 0}
```

III. Feladat

Informatikus szakon többféle kurzust is felvehetünk. Az informatikával kapcsolatos tárgyak kurzuskódja I betűvel, a matekos tárgyak kurzuskódja M betűvel, míg a szabadon választható tárgyak kurzuskódja X betűvel kezdődik.

Írj egy kurzuskodot_csoportosit nevű függvényt, amely egy olyan szöveget kap paraméterül, ami pontosvesszővel elválasztott kurzuskódokat tartalmaz!

A függvény csoportosítsa a szövegben szereplő kurzuskódokat "infós", "matekos" és "szabvál" kategóriákba, majd adja vissza az így kapott csoportosítást egy dictionary-ben, a példában látható formátumban

Ha a paraméterben kapott érték az üres string, akkor a függvény egy üres dictionary-vel térjen vissza!

Példa:

```
Input: 'IB370G;MBNXK114E;MBNXK114G;XA0021-GTK-MM1;IB370E'

Return: {'infos': ['IB370G', 'IB370E'], 'matekos': ['MBNXK114E', 'MBNXK114G'],
'szabval': ['XA0021-GTK-MM1']}



Input: 'ITN714G;IB402g;XN0011-01317;IB202g'

Return: {'infos': ['ITN714G', 'IB402g', 'IB202g'], 'matekos': [], 'szabval':
['XN0011-01317']}



Input: ''

Return: {}
```

IV. Feladat

Írj egy statisztika nevű függvényt, amely egy listát kap paraméterül! A lista fájlok neveit tartalmazza, kiterjesztéssel együtt. A fájlnévben a kiterjesztés alatt a legutolsó pont karakter után lévő szöveget értjük. A függvény számolja meg, hogy az egyes kiterjesztések hányszor fordulnak elő a listában, és az eredményt adja vissza egy dictionary-ben, a példában látható formátumban!

A feladatot úgy oldd meg, hogy a kiterjesztés vizsgálata során ne különböztesd meg a kis- és nagybetűket (tehát pl. hello.py és WORLD.PY egyaránt py kiterjesztésűek).

Példa:

```
Input: ['feladat.py', 'Bolygo.java', 'HELLOFRIENDS.MP4', 'TEST.PY',
'biro.gib.maxpont.py', 'russian-driving-fails.mp4']

Return: {'py': 3, 'java': 1, 'mp4': 2}
```