42(,195) km dát

Zadanie 1 – deadline: 20. 3. 2026

Predstavte si cieľovú rovinku maratónu. Niekto šprintuje posledné metre, niekto už len silou vôle kráča k cieľovej čiare. Na veľkej obrazovke svieti oficiálny čas, bežci dostávajú medaily a v zákulisí sa okamžite začína analyzovať: kto mal najlepšie tempo? Kto prepálil začiatok? A koľko by ste museli bežať vy, aby ste skončili o jednu priečku vyššie?
Za každým výsledkom sa skrýva množstvo dát – medzičasy na jednotlivých kilometroch, oficiálny čas od štartu (*gun time*), čistý čas od prebehnutia štartovej brány (*net time*), poradie v kategórii či rozdiely medzi bežcami. V tomto zadaní sa pozrieme na maratón nie ako diváci, ale ako dátoví analytici. Budete pracovať so súborom skutočných výsledkov z košického Medzinárodného maratónu mieru, ktoré načítate, spracujete a následne z nich získate rôzne zaujímavé informácie.
Pri riešení si precvičíte prácu so súbormi, spracovanie CSV dát, používanie zoznamov a slovníkov (vrátane vnorených štruktúr), triedenie dát, výpočty a prevody medzi rôznymi reprezentáciami času. Budete pracovať s podmienkami, cyklami, funkciami s viacerými parametrami aj návratovými hodnotami a naučíte sa premýšľať nad tým, ako vhodne modelovať dáta v Pythone, aby sa s nimi dalo efektívne pracovať.

Poznámka: Uvedené funkcie viete implementovať nezávisle, avšak pre úplné pochopenie úlohy odporúčame riešiť úlohy v uvedenom poradí. Okrem týchto funkcií môžete implementovať aj ďalšie pomocné funkcie, hodnotené však budú iba tie uvedené v tomto dokumente.

Úloha 1: `load_data` (2 body)

Vašou úlohou je implementovať funkciu `load_data(file_path)`, ktorá načíta údaje zo vstupného CSV súboru a prevedie ich do zoznamu slovníkov.
Ukážkový vstupný súbor nájdete v projekte zadania (`data.csv`). V súbore každý riadok obsahuje dáta o bežcovi, bežci sú zoradení podľa štartového čísla (*bib*). Jednotlivé stĺpce obsahujú štartové číslo, meno, klub, pohlavie, medzičasy na vybraných kilometroch a na konci oficiálny čas (*gun time*) a čistý čas (*net time*). Medzičasy sú namerané po 5, 10, 15, 20, 21, 26, 31, 36 a 41 kilometroch. Všetky časy sú uvedené vo formáte H:MM:SS, pričom niektoré hodnoty môžu byť prázdne (napríklad ak bežec nedokončil preteky).

Funkcia teda načíta tieto dáta do zoznamu slovníkov, ponechajte poradie bežcov ako sú uvedení vo vstupnom súbore. Každý prvok zoznamu bude reprezentovať jedného bežca a bude obsahovať všetky dôležité informácie o jeho výkone v nasledovnom tvare:

```
{
    "bib": 101,                # štartové číslo (int)
    "name": "Novak Jan",       # meno bežca ako v súbore (string)
    "gender": "M",             # kategória – "M" alebo "F" (string)
    "splits": {                # medzičasy v sekundách
        "5": 1092,
        "10": 2215,
        "21": 5710,
        ...
    },
    "gun_time": 11140,          # oficiálny čas od štartu v sekundách (int
alebo -1)
    "net_time": 10999          # čistý čas v sekundách (int alebo -1)
}
```

V slovníku `splits` budú uložené medzičasy na jednotlivých kilometroch. Kľúčom je vzdialenosť (ako string) a hodnotou je čas v sekundách od štartu. Ak medzičas pre daný kilometer v súbore chýba, do slovníka ho nepridávajte.
Časy v súbore sú uvedené vo formáte H:MM:SS. Vašou úlohou je tieto časy skonvertovať na počet sekúnd (ako celé číslo). Ak je finálny čas (`gun_time` alebo `net_time`) prázdny alebo neplatný, zapíšte hodnotu -1.

Parameter funkcie:

- `file_path` – cesta k CSV súboru so vstupnými dátami (string)

Návratová hodnota:

- zoznam slovníkov (list of dict), kde každý slovník reprezentuje jedného bežca v štruktúre uvedenej vyššie

Poznámka: Môžete predpokladať, že vstupný súbor má konzistentný formát a neobsahuje chybné hodnoty (iba chýbajúce). Spracovanie chýb teda nepotrebujete riešiť.

Úloha 2: `find_fastest` (0,5 boda)

Keď už máme dáta načítané a prehľadne uložené, môžeme sa konečne pozrieť na to, čo väčšinu divákov zaujíma najviac – kto bol najrýchlejší. V maratóne sa však výsledky vždy vyhodnocujú osobitne pre mužov a ženy, preto budeme pracovať s jednotlivými kategóriami zvlášť.
Vašou úlohou je implementovať funkciu `find_fastest(all_results, category, n)`, ktorá nájde najrýchlejších bežcov v zadanej kategórii na základe ich čistého času (`net_time`).

Parametre funkcie:

- `all_results` – zoznam slovníkov reprezentujúcich bežcov, tak ako ho vracia funkcia `load_data`. Každý prvok zoznamu obsahuje informácie o jednom bežcovi.
- `category` – reťazec určujúci kategóriu, v ktorej chceme vyhľadávať najrýchlejších. Očakáva sa hodnota `"M"` pre mužov alebo `"F"` pre ženy.
- `n` – celé číslo určujúce, koľko najrýchlejších bežcov má funkcia vrátiť. Predvolená hodnota má byť `100`.

Funkcia má vrátiť zoznam mien (stringov) najrýchlejších bežcov v danej kategórii, zoradených od najrýchlejšieho po pomalšieho podľa hodnoty `net_time`. Do výberu zahrňte iba bežcov, ktorí maratón dokončili, teda majú `net_time` rôzny od -1. Ak je v kategórii menej než *n* bežcov, ktorí dobehli do cieľa, vráťte iba toľko mien, koľko je dostupných.

Úloha 3: `get_position` (0,5 boda)

V maratóne nerozhoduje iba konečný čas. Často je zaujímavé sledovať aj priebežné poradie – kto bol vpredu na 10. kilometri, kto sa posúval dopredu a kto naopak strácal. V tejto úlohe sa pozrieme práve na takéto medzivýsledky. Implementujte funkciu `get_position(all_results, station, bib)`, ktorá určí priebežné poradie konkrétneho bežca na zadanom medzičase v jeho kategórii.

Parametre funkcie:

- `all_results` – zoznam slovníkov s výsledkami všetkých bežcov (výstup z `load_data`).

- `station` – vzdialenosť medzičasu (napr. `"10"`, `"21"`, `"31"`), pre ktorý chceme zistiť poradie. Ide o hodnotu, ktorá sa používa ako kľúč v slovníku `splits`.
- `bib` – štartové číslo bežca (int), ktorého priebežnú pozíciu chceme určiť.

Pri výpočte pozície bežca berte do úvahy jeho kategóriu (pohlavie), a rátajte iba s bežcami, ktorým bol nameraný medzičas na danej stanici. Týchto bežcov zoraďte podľa času na danom medzičase (od najrýchlejšieho po najpomalšieho) a určte poradie hľadaného bežca v tomto zozname. Poradie začína od 1.
Funkcia má vrátiť celé číslo predstavujúce priebežnú pozíciu bežca na zadanom medzičase v rámci jeho kategórie.

Úloha 4: `save_results` (1 bod)

Implementujte funkciu `save_results(all_results, category, station)`, ktorá vytvorí CSV súbor s poradím bežcov na zadanom medzičase v rámci vybranej kategórie.

Parametre funkcie:
- `all_results` – zoznam slovníkov s výsledkami všetkých bežcov (výstup z `load_data`).
- `category` – kategória, pre ktorú chceme generovať výsledky (`"M"` alebo `"F"`).
- `station` – vzdialenosť medzičasu (napr. `"5"`, `"10"`, `"21"`), podľa ktorej sa bude určovať poradie.

Funkcia má najskôr vybrať všetkých bežcov danej kategórie, ktorí majú zaznamenaný medzičas na zadanom kilometri. Týchto bežcov následne zoradí podľa času na danom medzičase (od najrýchlejšieho po najpomalšieho). Na základe tohto zoradenia určí ich priebežné poradie (počítané od 1). Výsledok zapíšte do CSV súboru s názvom v tvare: *results_<category>_<station>K.csv* (napríklad *results_F_15K.csv*).

Každý riadok výstupného súboru má obsahovať štyri hodnoty oddelené čiarkou:
`poradie,štartové_číslo,meno,čas_na_medzičase`

Čas na medzičase zapíšte v klasickom formáte H:MM:SS. Funkcia nič nevracia (jej návratová hodnota je `None`), jej úlohou je vytvoriť správne naformátovaný súbor. Ukážkový výstup nájdete v projektovom priečinku (*results_M_10K.csv*).

Úloha 5: `get_gender_results` (1 bod)

Samotné poradie víťazov je síce dôležité, ale o priebehu pretekov nám veľa napovie aj pohľad na základné štatistiky celej kategórie. Aký bol najlepší čas? Aký najhorší? Aký je priemerný výkon všetkých, ktorí dobehli do cieľa? A koľko bežcov napokon maratón nedokončilo?
Implementujte funkciu `get_gender_results(all_results, category)`, ktorá pre zadanú kategóriu vygeneruje prehľad základných štatistík.

Parametre funkcie:
- `all_results` – zoznam slovníkov s výsledkami všetkých bežcov (výstup z `load_data`).
- `category` – kategória, pre ktorú chceme vypočítať štatistiky (`"M"` alebo `"F"`).

Funkcia má pracovať iba s bežcami danej kategórie. Z nich vyberte tých, ktorí maratón dokončili (teda majú hodnotu `net_time` rôznu od -1), a na základe ich čistého času určte:

- meno víťaza (bežca s najmenším `net_time`),
- najlepší čas (vo formáte H:MM:SS),
- priemerný čas všetkých, ktorí dobehli do cieľa (zaokrúhľujte smerom nadol na celé sekundy),
- najhorší čas (najväčší `net_time`, vo formáte H:MM:SS),
- počet bežcov, ktorí nedokončili maratón (ktorí majú neplatný `net_time`).

Funkcia má vrátiť slovník v tvare:

```
{
    "Winner": "Meno Priezvisko",
    "Best": "H:MM:SS",
    "Average": "H:MM:SS",
    "Worst": "H:MM:SS",
    "DNF": 3
}
```

Úloha 6: `find_most_consistent` (2 body)

Skúsení bežci vedia, že kľúčom k dobrému výsledku je rovnomerné tempo. Niekto začne príliš rýchlo a v druhej polovici výrazne spomalí, iný si drží stabilné tempo od štartu až do cieľa. V tejto úlohe sa pokúsite nájsť práve takého bežca – toho, ktorý bežal najkonzistentnejšie. Implementujte preto funkciu `find_most_consistent(all_results, category=None)`, ktorá nájde bežca s najvyrovnanejším tempom počas celého maratónu.

Parametre funkcie:

- `all_results` – zoznam slovníkov s výsledkami všetkých bežcov (výstup z `load_data`).
- `category` – voliteľný parameter určujúci kategóriu (`"M"` alebo `"F"`). Ak je zadaný, funkcia pracuje iba s bežcami danej kategórie. Ak je `None`, zohľadňujú sa všetci bežci.

Do výberu zahrňte iba bežcov, ktorí majú platný cieľový čas (`gun_time` rôzny od -1), keďže práve ten budeme používať pri výpočte celkového priemerného tempa.

Pri implementácii si dajte pozor najmä na tieto časté chyby:

- Nevytvárajte tempo delením kumulatívneho času celkovou vzdialenosťou – vždy počítajte tempo po jednotlivých úsekoch (viď poznámku).
- Pred výpočtom sektorov je potrebné zabezpečiť, aby boli medzičasy spracované v správnom poradí podľa vzdialenosti.
- Ošetrite prípad chýbajúcich medzičasov (ak niektorý split neexistuje, nemožno ho použiť do výpočtu).
- Pri porovnávaní bežcov používajte priemernú absolútnu odchýlku tempa v jednotlivých úsekoch od celkového priemerného tempa.

Pre každého bežca teda:

1. Vypočítajte tempo jednotlivých úsekov trate (v sekundách na kilometer).
2. Vypočítajte jeho celkové priemerné tempo ako `gun_time` / 42,195.
3. Určite priemernú absolútnu odchýlky jednotlivých úsekov od tohto priemeru. Funkcia vracia pretekára s najmenšou priemernou absolútnou odchýlkou.

Funkcia má vrátiť trojicu hodnôt:

- meno najkonzistentnejšieho bežca (string),
- hodnotu vypočítanej priemernej absolútnej odchýlky (float),
- jeho priemerné tempo (float, v sekundách na kilometer).

Poznámka: Pri výpočte konzistentnosti je dôležité postupovať správne. Medzičasy v dátach predstavujú kumulatívne časy od štartu (napr. čas na 5 km, 10 km, 21 km atď.). Preto nestačí jednoducho vydeliť čas na 21 km hodnotou 21 – tým by ste získali iba priemerné tempo od štartu po daný bod. Správny postup je počítať tempo jednotlivých úsekov, teda rozdiel času medzi dvoma po sebe idúcimi medzičasmi vydelený rozdielom vzdialeností. Rovnako treba zohľadniť aj posledný úsek od posledného zaznamenaného medzičasu až do cieľa (42,195 km).

Pri kontrole hodnôt typu float, môže dôjsť k drobným rozdielom spôsobeným spôsobom reprezentácie desatinných čísel v počítači. Z tohto dôvodu budú výsledné hodnoty porovnávané s očakávaným riešením s toleranciou 0,001, teda na presnosť troch desatinných miest. Nie je preto potrebné výsledok explicitne zaokrúhľovať.

Úloha 7: `get_target_pace_for_position` (1 bod)

V cieli často počuť vetu: „Keby som bol o minútu rýchlejší, bol by som o dve miesta vyššie.“ Ale koľko presne treba zrýchliť, aby ste sa posunuli v poradí? Implementujte funkciu `get_target_pace_for_position(all_results, pos, category=None)`, ktorá vypočíta cieľové tempo potrebné na dosiahnutie určitej pozície v celkovom poradí (alebo v rámci kategórie).

Parametre funkcie:

- `all_results` – zoznam slovníkov s výsledkami všetkých bežcov (výstup z `load_data`).
- `pos` – cieľová pozícia (int), ktorú chceme dosiahnuť. Pozícia sa počíta od 1 (teda `pos=1` znamená víťaza).
- `category` – voliteľný parameter určujúci kategóriu (`"M"` alebo `"F"`). Ak je zadaný, uvažujú sa iba bežci danej kategórie. Ak je `None`, berú sa do úvahy všetci bežci.

Postup výpočtu je nasledovný: najskôr vyberte všetkých bežcov, ktorí majú platný cieľový čas (`gun_time` rôzny od -1), prípadne ich obmedzte na zadanú kategóriu. Vypočítajte cieľový čas, ktorý vám zaistí vysnívané umiestnenie (musíte byť aspoň o jednu sekundu rýchlejší ako bežec, ktorý končil na danom mieste), a určte potrebné priemerné tempo na kilometer (v sekundách na kilometer), pričom celková vzdialenosť maratónu je 42,195 km.

Funkcia má vrátiť tempo vo formáte "MM:SS" (minúty a sekundy na kilometer), pričom hodnoty sekúnd majú byť vždy dvojciferné.

Úloha 8: `calculate_target_pace` (1 bod)

Nejakým zázrakom sa niekedy stane, že po polovici trate sa bežec cíti dobre, a uvedomí si, že môže zaútočiť na určitý cieľový čas – napríklad prekonať hranicu troch hodín. Otázka potom znie: aké tempo musí držať na zvyšných kilometroch, aby to ešte stihol? Implementujte funkciu `calculate_target_pace(section, section_time, target_time)`, ktorá vypočíta potrebné priemerné tempo na zvyšnej časti trate tak, aby bežec dosiahol zadaný cieľový čas.

Parametre funkcie:

- `section` – vzdialenosť (číselná hodnota), ktorú má bežec už za sebou (napr. 20 alebo 21).
- `section_time` – čas, za ktorý túto vzdialenosť prebehol, vo formáte "H:MM:SS" (string).
- `target_time` – cieľový čas, ktorý chce bežec dosiahnuť v cieli, takisto vo formáte "H:MM:SS" (string).

Funkcia má vrátiť tempo vo formáte "MM:SS" (minúty a sekundy na kilometer), pričom sekundy musia byť vždy dvojciferné. Tempo musí byť dostatočne dobré na to, aby bežec dosiahol minimálne vysnívaný celkový čas. Ak však z výpočtu vyjde nedosiahnuteľne nízke tempo (pod 3 minúty na kilometer), funkcia má vrátiť hodnotu `None`.

Úloha 9: `find_slowest_starter` (0,5 boda)

Niektorí bežci sa snažia vybehnúť čo najrýchlejšie po štartovom výstrele, iní sa naopak na štarte zdržia – či už kvôli hustote davu, alebo preto, že štartujú z menej výhodnej pozície. Rozdiel medzi oficiálnym časom od výstrelu (*gun time*) a čistým časom od prebehnutia štartovej brány (*net time*) nám vie napovedať, kto sa na štarte najviac zdržal.
Implementujte funkciu `find_slowest_starter(all_results)`, ktorá nájde bežca s najväčším rozdielom medzi hodnotami `gun_time` a `net_time`.

Parameter funkcie:

- `all_results` – zoznam slovníkov s výsledkami všetkých bežcov (výstup z `load_data`).

Do výpočtu zahrňte iba bežcov, ktorí majú platný `gun_time` aj `net_time`. Funkcia má vrátiť dvojicu hodnôt:

- meno bežca (string),
- veľkosť rozdielu v sekundách (int).

Úloha 10: `calculate_split` (0,5 boda)

Implementujte funkciu `calculate_split(all_results, bib)`, ktorá vypočíta časový rozdiel medzi zadaným bežcom a víťazom jeho kategórie.

Parametre funkcie:

- `all_results` – zoznam slovníkov s výsledkami všetkých bežcov (výstup z `load_data`).
- `bib` – štartové číslo (int) bežca, pre ktorého chceme rozdiel vypočítať.

Najskôr nájdite bežca so zadaným štartovým číslom a zistite jeho kategóriu (`gender`). Následne určte cieľový čas (`gun_time`) víťaza tejto kategórie. Rozdiel (split) vypočítajte ako rozdiel medzi `gun_time` zadaného bežca a `gun_time` víťaza kategórie. Výsledok vyjadrite vo formáte času (návratová hodnota je typu string):

- ak je rozdiel aspoň jedna hodina, použite formát H:MM:SS,
- ak je rozdiel menší než jedna hodina, použite formát M:SS.

Ak sa v zozname nenachádza bežec so zadaným štartovým číslom, funkcia má vrátiť prázdny reťazec.

Vaše riešenia môžete otestovať aj pomocou sady testov v súbore `assignment1_test.py`. Pri hodnotení vášho riešenia sa použijú podobné testy, avšak ich bude viac. Testy budú zverejnené dva týždne pred odovzdaním (6. 3. 2026). Vaše riešenie môžete testovať aj vo funkcii `main`. V odovzdanom riešení nenechajte príkazy mimo funkcií.

Približná dĺžka riešenia: *cca. 200 riadkov formátovaného kódu s komentármi*.