Parkujeme

Prvé zadanie sa sústreďuje na spracovanie záznamov z parkovacieho systému. Precvičíte si pritom prácu so súbormi a základnými údajovými typmi v Pythone, a získate cenné skúsenosti so spracovaním dát v rôznych formátoch. Okrem základných operácií sa pozrieme na problém prevedenia medzi rôznymi reprezentáciami, na štatistické výpočty, a na optimalizáciu.

Vaše riešenie bude pracovať s údajmi, ktoré si načítate z dvoch súborov – ukážky nájdete v priečinku `samples`. Prvý typ súboru nesie názov `parking_logs_XX.csv`, kde `XX` je číslo ukážky, a obsahuje údaje vo formáte hodnôt oddelených čiarkou (*comma separated values*). V týchto súboroch každý riadok reprezentuje parkovanie jedného vozidla, pričom riadok vyzerá nasledovne:

```
TV333XI,6,2,16,46
```

kde prvý stĺpec obsahuje ŠPZ vozidla, ďalšie dva stĺpce reprezentujú čas príchodu (v našom prípade 6:02) a posledné dva stĺpce čas odchodu z parkoviska (v našom prípade 16:46). Stĺpce sú oddelené čiarkou a posledný riadok v súbore je prázdny. Záznamy sú zoradené podľa príchodu vozidla na parkovisko.

Druhý súbor obsahuje cenník parkovania a nesie názov `prices_XX.txt`, kde `XX` je číslo ukážky. Súbor obsahuje riadky vo formáte:

```
30m: 0.5
```

kde hodnota pred dvojbodkou vyjadruje dĺžku parkovania a hodnota po dvojbodke cenu (v eurách). Každý súbor obsahuje šesť kľúčov, na základe ktorých sa vypočítava cena parkovania za daný čas: `30m` – parkovanie do pol hodiny; `1h` – parkovanie do hodiny; `3h` – parkovanie do troch hodín; `6h` – parkovanie do šiestich hodín; `1d` – parkovanie za celý deň; `h+` – príplatok za každú začatú hodinu nad rámec tarifného intervalu. Spôsob výpočtu ceny je popísaný nižšie pri príslušnej metóde.

Vo vašom riešení implementujete funkcie pre načítanie údajov zo súborov, pre výpočet ceny parkovania, pre zistenie rôznych štatistických ukazovateľov, a pre optimalizáciu ceny pre maximálny príjem prevádzkovateľa.

Úloha 1 – 0,75 bodov

Implementujte metódu `load_parking_records(file_path)`, ktorá zo súboru dostupného na zadanej ceste (parameter `file_path`) načíta parkovacie záznamy vo formáte CSV, ako to bolo popísané vyššie. Metóda vracia načítané údaje ako zoznam n-tíc (*list of tuples*), pričom každá n-tica obsahuje päť hodnôt: ŠPZ, hodina a minúta príchodu, hodina a minúta odchodu. ŠPZ nech je reprezentovaná ako reťazec (`string`), ostatné hodnoty majú byť celé čísla (`integer`).

Pri hodnotení získate 0,25 bodov za správny formát a 0,5 za správne načítané hodnoty.

Úloha 2 – 0,75 bodov

Implementujte metódu `load_prices(file_path)`, ktorá zo súboru dostupného na adrese `file_path` načíta cenník parkoviska, pričom súbor bude `txt` súbor s formátom uvedeným vyššie. Metóda vracia jednu hodnotu – slovník (`dictionary`) s načítanými hodnotami. Kľúče budú krátke reprezentácie jednotlivých časových pásiem (`30m`, `1h`, `3h`, `6h`, `1d`, `h+`) a hodnoty budú tarify prislúchajúce k týmto intervalom. Kľúče nech majú typ `string` a hodnoty `float`.

Pri hodnotení získate 0,25 bodov za správny formát a 0,5 za správne načítané hodnoty.

Úloha 3 – 0,5 bodov

Implementujte metódu `calculate_parking_time(start_h, start_m, end_h, end_m)`, ktorá vypočíta dĺžku parkovania v minútach na základe času príchodu a odchodu, ktoré dostane ako parametre. Metóda má teda jednu návratovú hodnotu: celé číslo reprezentujúce dĺžku parkovania v minútach.

Napríklad volanie `calculate_parking_time(7, 45, 11, 23)` vráti hodnotu 218 (15 + 3 * 60 + 23).

Úloha 4 – 1 bod

Implementujte metódu `get_parking_fee(time_in_minutes, prices)`, ktorá má dva parametre:

- `time_in_minutes` – dĺžka parkovania v minútach (ako bolo vypočítané metódou `calculate_parking_time()`)
- `prices` – slovník s cenníkom parkoviska (z metódy `load_prices()`).

Metóda vráti cenu parkovania ako desatinné číslo, pričom parkovanie do 15 minút je bezplatné (vracia sa `0.0`). Ak niekto parkuje menej ako pol hodinu, tak si zaplatí polhodinový lístok (`30m`). V ostatných prípadoch najprv zaplatí základnú cenu za parkovanie a následne priplatí za každú začatú hodinu. Napríklad ak parkoval celkovo 128 minút (2h a 18m), tak zaplatí najprv za parkovanie do hodiny (`1h`), a následne dvakrát zaplatí príplatok za začatú hodinu (`h+`). Pritom však hľadá výhodnejší variant, čiže ak cena za parkovanie do troch hodín (`3h`) je nižšia ako pred tým vypočítaná suma, kúpi si parkovací lístok na tri hodiny. Obdobne to funguje aj pri ďalších kategóriách (`3h` vs `6h` a `6h` vs `1d`).

Poznámka: Pri výpočtoch môžete rátať s tým, že dve polhodinové lístky vychádzajú cenovo rovnako ako hodinový lístok, a že príplatok za začatú hodinu je menší ako hodinový lístok. Ak klient začne novú hodinu, musí si ju zaplatiť celú, nemôže si kúpiť iba polhodinový lístok.

Úloha 5 – 0,5 bodov

Implementujte metódu `calculate_average_parking_fee(records, prices)`, ktorá na základe načítaných záznamov `records` a cenníku `prices` vypočíta a vráti priemernú sumu zaplatenú za parkovanie za celý deň. Metóda má iba jednu návratovú hodnotu typu `float`. Parameter `records` má štruktúru podľa návratovej hodnoty `load_parking_records()` a `prices` podľa metódy `load_prices()`.

Úloha 6 – 0,5 bodov

Implementujte metódu `calculate_average_parking_time(records)`, ktorá na základe načítaných záznamov `records` vypočíta a vráti priemernú dĺžku parkovania. Metóda má iba jednu návratovú hodnotu typu `float`. Parameter `records` má štruktúru podľa návratovej hodnoty `load_parking_records()`.

Úloha 7 – 0,5 bodov

Implementujte metódu `calculate_average_stays(records)`, ktorá na základe načítaných záznamov `records` vypočíta a vráti priemerný počet návštev parkoviska vozidlom. V niektorých prípadoch zistíte, že to isté auto parkovalo na parkovisku viackrát, táto metóda zohľadňuje túto skutočnosť. Metóda má iba jednu návratovú hodnotu typu `float`. Parameter `records` má štruktúru podľa návratovej hodnoty `load_parking_records()`.

Úloha 8 – 1 bod

Implementujte metódu `get_most_common_region(records)`, ktorá na základe načítaných záznamov `records` nájde a vráti najčastejší kód okresu, pričom opakované návštevy toho istého vozidla zaráta viackrát. Metóda vráti jeden reťazec – dvojpísmenový kód okresu, z ktorého parkovisko navštívia autá najviac krát. Parameter `records` má štruktúru podľa návratovej hodnoty `load_parking_records()`.

Poznámka: Ak viac okresov má rovnaké zastúpenie v dátach, metóda vráti kód toho okresu, ktorý sa vyskytol prvýkrát v záznamoch.

Úloha 9 – 0,5 bodov

Implementujte metódu `get_busiest_hour(records)`, ktorá na základe načítaných záznamov `records` nájde a vráti hodinu, kedy parkovisko je najviac vyťažené. Parameter `records` má štruktúru podľa návratovej hodnoty `load_parking_records()`.

Metóda prechádza všetkými hodinami, kedy parkovisko je otvorené, a zistí, koľko áut využilo parkovisko v danej hodine. Auto zarátajte do počtu ak prichádzalo pred alebo v danú hodinu, a odchádzalo v alebo po danej hodine. Napríklad ak auto prišlo o 8:47 a odišlo o 15:27, tak sa zaráta do hodín 8 až 15 (vrátane). Otváraciu dobu parkoviska zistite priamo zo záznamov. Metóda vráti jedno celé číslo.

Poznámka: Ak viac hodín má rovnaký počet parkovaní, tak metóda vráti najskoršiu z týchto možností.

Úloha 10 – 2 body

Implementujte metódu `get_max_number_of_cars(records)`, ktorá na základe načítaných záznamov `records` nájde a vráti maximálny počet áut, ktoré stáli na parkovisku v tom istom čase (na úrovni minút). Parameter `records` má štruktúru podľa návratovej hodnoty `load_parking_records()`. Metóda vracia dve hodnoty – maximálny počet áut v jednom okamihu, a zoznam počtov parkujúcich áut za každú minútu. Otváraciu dobu parkoviska zistite priamo zo záznamov.

Pri spočítavaní áut dodržujeme zásady indexovania v Pythone: ak auto prichádzalo o 8:02 a odchádzalo o 10:39, tak ho zarátate do počtu za každú minútu od 8:02 (vrátane) až po 10:38. V čase 10:39 už auto neberiete do úvahy pri určení počtu áut stojacich na parkovisku.

Poznámka: Pre efektívne riešenie môžete využiť skutočnosť, že záznamy sú zoradené podľa príchodu áut na parkovisko. Nezabudnite tiež zohľadniť, že autá parkovisko už možno opustili.

Úloha 11 – 2 body

Implementujte metódu `optimize_hourly_fee(records, prices)`, ktorá na základe načítaných záznamov `records` a cenníku `prices` určí takú hodnotu príplatku za začatú hodinu, ktorá bude maximalizovať celkový príjem prevádzkovateľa parkoviska. Parameter `records` má štruktúru podľa návratovej hodnoty `load_parking_records()` a `prices` podľa metódy `load_prices()`. Metóda vracia jednu hodnotu: optimálnu cenu za začatú hodinu ako `float`.

Hľadaná optimálna hodnota musí byť vyššia ako cena polhodinového lístka, a menšia ako cena hodinového lístka. Cena takisto musí byť násobok 10 centov, čiže X.X0 EUR.

Vaše riešenia môžete otestovať aj pomocou sady testov v súbore `tests_1a.py`. Pri hodnotení vášho riešenia sa použijú podobné testy, avšak ich bude viac.

Približná dĺžka riešenia: *cca. 180 riadkov formátovaného kódu bez komentárov*.