Ideme na teambuilding

2. zadanie – deadline: 12. 4. 2024

Každý dobrý manažér vie, že kľúčom k úspechu je dobrý pracovný kolektív. A čo by bolo lepšie na vybudovanie takejto súdržnosti ako teambuilding? Rozhodli ste sa teda zorganizovať víkend v Tatrách, avšak po zarezervovaní ubytovania a rôznych zaujímavých aktivít ste narazili na ďalší problém, a to na problém transferu.

Keďže celý rozpočet ste minuli na kvalitné aktivity, museli ste poprosiť vašich zamestnancov, aby na teambuilding prišli autom a doniesli aj svojich kolegov. Vzhľadom na to, že na teambuilding sa stále len chystáte, bohužiaľ sa v radoch zamestnancov nájdu ľudia, ktorí sa nemajú radi a práve preto by neradi cestovali tým istým autom. Vašou úlohou je teda nájsť čo najlepšie rozdelenie zamestnancov v autách tak, aby v priemere boli vaši zamestnanci čo najviac spokojní.

Pri riešení tohto problému využijete základné objektovo orientované riešenie, ako aj rôzne optimalizačné metódy. Konkrétne implementujete dve triedy: `Car` pre reprezentáciu auta so šoférom a maximálne tromi ďalšími pasažiermi (aby sa bez problémov zmestili); a triedu `Employee` pre reprezentáciu zamestnancov, ktorí sú definovaní preferovanými sedadlami a zoznamom svojich obľúbených a menej obľúbených kolegov. Následne implementujete niekoľko metód, ktoré vám pomôžu pri práci so zadanou úlohou a pri hľadaní riešenia.

Poznámka: Triedy viete implementovať nezávisle, avšak pre implementáciu musíte pochopiť ich vzájomné prepojenia. Funkcie uvedené v súbore `seating.py` sú tiež navzájom prepojené, viete ich teda využívať pri implementácii ďalších funkcií. Odporúčame začať s implementáciou tried `Car` a `Employee`, a následne sa zamerať na samotné riešenie problému. Pri odovzdaní riešenia nahrajte všetky tri súbory (`car.py`, `employee.py`, `seating.py`) na Google Drive.

Trieda `Car` – 1 bod

Trieda `Car` reprezentuje auto, ktorým pôjdu na teambuilding maximálne štyria zamestnanci. Auto bude reprezentované ako zoznam štyroch hodnôt, pričom postupne tieto hodnoty naplníte pasažiermi, ktorí si sadli do daného auta. V zozname platia nasledovné zásady:

- na prvej pozícii sa nachádza vodič;
- pod indexom jedna nájdete pasažiera sediaceho vpredu vedľa vodiča;
- posledné dve miesta predstavujú zadné sedadlá (poradie nepodstatné).

V triede implementujte nasledovnú funkcionalitu:

- (0,1b) v konštruktore (`__init__`) inicializujte členskú premennú s názvom `seats`, ktorá bude zoznam štyroch hodnôt, pri vytvorení nového auta nech je to zoznam štyroch `None` hodnôt.
- (0,1b) metóda `has_driver()` má vrátiť hodnotu `True` alebo `False` na základe toho, či už auto má prideleného šoféra.
- (0,1b) metóda `has_empty_front()` vracia `True` alebo `False` na základe toho, či je ešte predné sedadlo vedľa vodiča voľné.
- (0,1b) metóda `has_empty_back()` vracia `True` alebo `False` na základe toho, či je ešte voľné niektoré zo zadných sedadiel.

- (0,1b) metóda `get_number_of_empty()` vracia počet voľných sedadiel v aute (maximálne 4).
- (0,3b) metóda `add_passenger(p, pos)` pridá pasažiera `p` (objekt typu `Employee`) na pozíciu `pos` v zozname pasažierov. Pričom ak pozícia má neplatný index, vygenerujte `ValueError` so správou: *Cannot add passenger at index X*, kde namiesto *X* napíšete hodnotu vstupného parametra. Pre účely nášho riešenia budeme považovať všetky záporné indexy za neplatné. Okrem toho sa `ValueError` vygeneruje aj v prípade, ak dané miesto je už obsadené, chybu vygenerujte so správou *Position X already occupied*. Ak hodnota prvého parametra nebude typu `Employee`, vygenerujte `TypeError` s ľubovoľnou správou. Ak sú všetky vstupné hodnoty platné, uložte pasažiera na danú pozíciu.
- (0,2b) metóda `get_car_satisfaction()` má vrátiť priemernú spokojnosť pasažierov v aute so svojimi spolucestujúcimi. Návratová hodnota je teda číslo, obdobnú funkciu nájdete aj v triede `Employee`, ktorú môžete využiť aj tu.

Trieda Employee – 3 body

Trieda `Employee` reprezentuje zamestnanca, ktorí sa zúčastní na teambuildingu. Trieda bude obsahovať informácie o preferovanom sedadle daného zamestnanca, a bude zabezpečovať funkcionalitu súvisiacu s výberom zo zoznamu dostupných vozidiel.

Do triedy doplňte nasledovnú funkcionalitu:
- (0,25b) konštruktor (metóda `__init__`) nech vytvorí nový objekt zamestnanca, pričom zadefinujte členské premenné `will_drive` (typu `boolean`), ktorá vyjadruje, či zamestnanec je šoférom alebo nie; `sits_in_front`, ktorá vyjadruje, či zamestnanec chce sedieť vpredu (ak nie je zadaný druhý parameter, automaticky je to `True`, samozrejme aj v prípade, ak zamestnanec je šoférom); a členskú premennú `contacts`, ktorá je typu `dictionary` a po inicializácii bude prázdny.
- (0,25b) v metóde `set_contacts()` pridajte preferovaných spolucestujúcich zamestnanca, pričom ako vstup dostanete zoznam n-tíc (*list of tuples*). Každá n-tica obsahuje dve hodnoty, prvá z nich je zamestnanec, s ktorým aktuálny zamestnanec (ne)chce spolucestovať, a druhá hodnota je desatinné číslo vyjadrujúce váhu preferencie (hodnota medzi -1 a 1, čím vyššia, tým väčšia preferencia). V metóde aktualizujte členskú premennú `contacts`, pričom ak prvá hodnota v n-tici nie je typu `Employee`, vygenerujte `TypeError`. V opačnom prípade pridajte nový záznam do `dictionary contacts`, avšak ak pre daného zamestnanca ste už mali zadanú váhu, vygenerujte `ValueError`.
- (0,5b) v metóde `get_car_weight()` vypočítajte a vráťte celkovú váhu preferencií všetkých spolucestujúcich, ktorí už sedia v aute `car`, ktoré dostanete ako vstupný parameter. Ak daný pasažier nebol definovaný medzi preferenciami zamestnanca, preňho rátajte s váhou 0.
- (0,5b) v metóde `get_satisfaction()` vypočítajte spokojnosť pasažiera, ktorý už sedí v aute, ktoré dostanete ako vstupný parameter. Spokojnosť vypočítate ako súčin celkových váh všetkých spolucestujúcich z pohľadu aktuálneho zamestnanca a konštanty, ktorá vyjadruje spokojnosť so sedadlom. Ak sa pasažier nenachádza v aute, vráťte hodnotu -1. V opačnom prípade vypočítajte spokojnosť, pričom konštanta je definovaná ako 2 ak pasažier sedí na svojom preferovanom mieste a 0.5, ak sedí na ktoromkoľvek inom mieste. Pravidlá sú teda nasledovné:
    1. šofér musí sedieť za volantom, aby bol spokojný so svojím miestom;
    2. zamestnanec, ktorý chce sedieť vpredu, musí sedieť za volantom alebo vedľa aby bol spokojný so svojím miestom;

3. zamestnanec, ktorý chce sedieť vzadu, musí sedieť na jednom zo zadných sedadiel, aby bol spokojný so svojím miestom.

- (0,5b) v metóde `choose_car()` si zamestnanec vyberie auto, do ktorého si sadne zo zoznamu `cars`, ktorý dostane metóda ako vstupný parameter. Tento zoznam teda obsahuje objekty typu `Car`, z ktorých sa vyberá na základe nasledovných pravidiel:
  1. zamestnanec si môže sadnúť iba do auta, ktoré ešte obsahuje voľné miesto;
  2. zamestnanec preferuje autá, v ktorom celková váha spolucestujúcich je čo najväčšia;
  3. následne vyberá iba z áut, v ktorých je voľné jeho preferované miesto. Ak také auto nenájde, vyberá z tých, v ktorých sú voľné miesta.
  4. metóda na konci vráti referenciu na auto, ktoré zamestnancovi vyhovuje najviac. Pri riešení môžete rátať s tým, že aspoň jedno auto v zozname bude mať voľné miesto.
- (1b) metóda `take_seat()` vracia dve hodnoty, pričom prvá hodnota je index auta, do ktorého si zamestnanec sadne, a druhá návratová hodnota je index sedadla, na ktoré si sadne – samozrejme to musí byť voľné miesto. Funkciu implementujte podľa nasledovných pravidiel:
  1. zamestnanec si najprv vyberie auto
  2. následne si vyberie sedadlo, pričom
     - ak je šofér a auto nemá ešte šoféra, sadne si na miesto šoféra; ak sa v aute šofér už nachádza, zamestnanec si sadne vedľa neho (ak je miesto voľné), v opačnom prípade si sadne na náhodné voľné miesto.
     - ak zamestnanec chce sedieť vpredu a je miesto vedľa vodiča voľné, sadne si na dané miesto, v opačnom prípade si vyberie ľubovoľné voľné miesto.
     - ak zamestnanec chce sedieť vzadu, a je voľné miesto vzadu, sadne si na ľubovoľné z nich. V opačnom prípade vyberie ľubovoľné voľné miesto.

Súbor `seating.py` – 6 bodov

(2b) Implementujte metódu `load_example()`, ktorá dostane ako parameter cestu k csv súboru s informáciami o zamestnancoch, ktorí sa zúčastnia na teambuildingu. Metóda vracia zoznam načítaných zamestnancov, teda bude to zoznam objektov typu `Employee`. V csv súbore každý riadok je reprezentovaný nasledovne:

```
False,True,7:-0.03,9:0.91,6:-0.22
```

kde prvá hodnota vyjadruje, či daný človek je alebo nie je šofér, druhá hodnota či chce sedieť vpredu, a ďalšie hodnoty (ich počet sa mení) vyjadruje váhy iných zamestnancov, pričom pred dvojbodkou nájdete index zamestnanca (podľa poradia v súbore, indexovanie sa začína od 0) a za dvojbodkou je hodnota vyjadrujúca váhu, teda preferenciu ako spolucestujúceho (vždy zaokrúhlené na dve desatinné miesta, môže byť aj kladná, aj záporná).

(0,5b) Ďalej zadefinujte metódu `get_avg_satisfaction()`, ktorý nasimuluje nástup zamestnancov, ktorých dostanete ako prvý parameter `employees` (zoznam objektov typu `Employee`), do `car_no` počet áut. Zamestnanci si vyberajú miesto v aute podľa ich poradia v zozname (takže najprv si sadne prvý zamestnanec, potom druhý atď.). Po nástupe vypočítajte a vráťte priemernú spokojnosť všetkých zamestnancov v jednotlivých autách.

(2b) Implementujte metódu `get_all_seatings()`, ktorá vygeneruje všetky možné jedinečné kombinácie sedenia v rôznych autách. Ako parameter metóda dostane zoznam zamestnancov a počet áut. Môžete rátať s tým, že počet zamestnancov bude deliteľný 4, počet áut zodpovedá minimálnemu počtu áut potrebných pre daný počet zamestnancov, a zoznam zamestnancov bude obsahovať toľko šoférov, koľko je áut – v každom aute musí sedieť šofér.

Poradie zamestnancov v zozname bude náhodné. Metóda vracia zoznam zoznamov zoznamov, kde každý prvok reprezentuje jednu možnú kombináciu. Kombinácia bude reprezentovaná ako zoznam `car_no` zoznamov, kde vo vnútorných zoznamoch budete mať štyroch zamestnancov.

Poznámka: Pri riešení jedinečnej kombinácie neberte do úvahy poradie áut ani poradie zamestnancov sediacich vzadu. Napríklad pri ôsmych zamestnancoch kombinácie `[[Z1, Z2, Z3, Z4], [Z5, Z6, Z7, Z8]]` **a** `[[Z5, Z6, Z7, Z8], [Z1, Z2, Z3, Z4]]` **sa považujú za rovnaké (menilo sa iba poradie áut) ako aj kombinácie** `[[Z1, Z2, Z3, Z4], [Z5, Z6, Z7, Z8]]` **a** `[[Z1, Z2, Z4, Z3], [Z5, Z6, Z7, Z8]]` **– menilo sa iba poradie zamestnancov sediacich vzadu v prvom aute.**

(0,5b) Doplňte metódu `get_optimal_satisfaction()` s rovnakými parametrami ako `get_all_seatings()`, pričom táto metóda vráti maximálnu možnú priemernú spokojnosť zamestnancov pri danom zozname zamestnancov a danom počte áut. Návratová hodnota je teda jedno číslo – maximálna priemerná spokojnosť zo všetkých možných jedinečných kombinácií.

(1b) Implementujte metódu `get_seating_order()` s rovnakými parametrami ako predošlé metódy. Metóda rieši poradie nástupu pasažierov tak, aby bola dosiahnutá maximálna možná spokojnosť zamestnancov s ich spolucestujúcimi. Keďže ste dobrý manažér, nechcete im kázať, kam si majú sadnúť, radšej im prenechávate slobodnú voľbu, poradie nástupu však určíte vy. Pozor, nemusí existovať poradie nástupu, ktoré by garantovalo maximálnu možnú spokojnosť spomedzi všetkých kombinácií sedenia (z funkcie `get_optimal_satisfaction()`). Návratová hodnota metódy je zoznam zamestnancov, pričom ich poradie je nastavené tak, aby na konci nástupu bola spokojnosť čo najvyššia.