Kristers Andermanis 241RIC059

Ideja

Šī programma palīdz loģistikas speciālistam ātri aprēķināt:

- cik maksā preču piegāde ar dažādiem transporta veidiem,
- cik daudz CO_2 izmešu tā rada,
- kurš variants ir lētākais un videi draudzīgākais.

Skaidrojums

1. Vārdnīcu definēšana

```
tarifi = {"kravas_auto": 0.05, "vilciens": 0.03, "kuģis": 0.02}
emisijas = {"kravas_auto": 0.08, "vilciens": 0.04, "kuģis": 0.02}
```

- Pats sākums satur vārdnīcas ar datiem par transporta veidu tarifiem un emisijām.

2. Tukša saraksta izveide

```
kravas = []
```

- Izveidots tukšs saraksts, kurā programma vēlāk ieliks visu lietotāja ievadīto informāciju par kravām.

3. Datu ievade un cikls

```
n = int(input("Cik kravas jāaprēķina? "))
```

- Programma jautā, cik kravas jāaprēķina, un ievadīto informāciju formatē kā skaitli nevis tekstu.

```
for i in range(n):
```

- Tālāk atkarībā no ievadītā skaitļa tiek noteikts, cik reizes cikls atkārtosies. Tas nozīmē: ja ievadi “3”, tad nepieciešamie parametri jāievada 3 reizes.

```
    print(f"\nKrava #{i+1}")
    nosaukums = input("Nosaukums: ")
    attalums = float(input("Attālums (km): "))
    svars = float(input("Svars (kg): "))
    veids = input("Transporta veids (kravas_auto/vilciens/kuģis): ").lower()
```

- programma pieprasa ievadīt: kravas nosaukumu (teksts), attālumu kilometros (float – skaitlis ar komatu), svaru kilogramos (float) un transporta veidu, kur ievadītais teksts tiks izvadīts ar mazajiem burtiem (.lower()), lai nerastos kļūda, kad kāds ieraksta tektu, kas satur kādu lielo burtu.

4. Datu pārbaude

```
while True:
    veids = input("Transporta veids (kravas_auto/vilciens/kuģis): ").lower()
    if veids in tarifi:
        break
    print("❌ Nepareizs transporta veids! Mēģini vēlreiz.")
```

- Šeit notiek **ievades pārbaude**:

- while True: nozīmē "atkārto, līdz lietotājs ievada pareizi",
- if veids in tarifi: pārbauda, vai ievadītais transporta veids ir vārdnīcā tarifi,
- Ja nav — tiek izvadīts brīdinājums un transporta jautājums tiek uzdots atkārtoti.

5. Formulas

```
cena = attalums * svars * tarifi[veids]
co2 = attalums * svars * emisijas[veids]
```

- Programmai tiek dotas formulas, kuras tiks imazntotas aprēķinam.

6. Kritēriji, lai noteiktu vides draudzīgumu

```
if co2 < 1000:
        efekts = "Videi draudzīgs"
    elif co2 < 3000:
        efekts = "Vidēji efektīvs"
    else:
        efekts = "Liels CO₂ piesārņojums"
```

- Šeit tiek izmantoti loģiskie nosacījumi (if, elif, else), lai, balstoties uz aprēķināto CO_2 daudzumu, noteiktu, cik videi draudzīga ir piegāde. Ja aprēķinātie izmeši ir mazāki par 1000, programma piešķir vērtību "Videi draudzīga"; ja tie ir starp 1000 un 3000 – "Vidēji efektīva"; bet, ja pārsniedz 3000, tā piešķir "Liels CO_2 piesārņojums".

7. Saglabā visus kravas datus

```
kravas.append({
        "nosaukums": nosaukums,
        "veids": veids,
        "cena": cena,
        "co2": co2,
        "efekts": efekts
    })
```

- Šī daļa saglabā visu ievadīto informāciju par katru kravu. Funkcija append() nozīmē "pievienot sarakstam", un šajā gadījumā tā sarakstam "kravas" pievieno jaunu vārdnīcu, kurā glabājas visi dati par konkrēto kravu – tās nosaukums, transporta veids, aprēķinātā cena, CO_2 daudzums un efektivitātes novērtējums. Tas nozīmē, ka, ja lietotājs ievada vairākas kravas, katra no tām tiek saglabāta kā atsevišķs ieraksts šajā sarakstā. Beigās kravas satur visu informāciju par visām kravām, piemēram:

```
[
  {"nosaukums": "Koksne", "veids": "vilciens", "cena": 600.0, "co2": 800.0,
"efekts": "Videi draudzīga"},
  {"nosaukums": "Metāls", "veids": "kuģis", "cena": 400.0, "co2": 700.0,
"efekts": "Videi draudzīga"}
]
```

- Ja šī daļa nebūtu iekļauta, programma atcerētos tikai pēdējo ievadīto kravu.

8. Rezultātu izvadīšana

```
for k in kravas:
    print(f"{k['nosaukums']} ({k['veids']}): €{k['cena']:.2f}; "
          f"CO₂: {k['co2']:.1f} kg → {k['efekts']}")
```

- Šis for cikls izvada visu kravu sarakstu.

- k pārstāv vienu kravu no saraksta kravas.
- k['nosaukums'] nozīmē – paņem vērtību no vārdnīcas pēc atslēgas.
- :.2f nozīmē – noapaļo skaitli līdz 2 zīmēm aiz komata.
- :.1f – noapaļo līdz 1 zīmei aiz komata.

9. Atrod lētāko piegādi un izvada secinājumu

```
letaka = min(kravas, key=lambda x: x["cena"])

print(f"\nLētākā piegāde ir '{letaka['nosaukums']}' ar {letaka['veids']}.")
print("Secinājums: Šis maršruts ir", letaka["efekts"].lower())
```

- Šī koda daļa izmanto funkciju min(), lai atrastu kravu ar viszemāko cenu sarakstā kravas. Parametrs key=lambda x: x["cena"] nozīmē, ka programma skatās uz katras kravas cenas vērtību un salīdzina tās savā starpā. Rezultātā tiek atrasta lētākā piegāde, un programma izvada tās nosaukumu, transporta veidu un arī secinājumu par to, cik videi draudzīgs ir šis maršruts.

```
=== REZULTĀTI ===
Koks (vilciens): €600.00; CO₂: 800.0 kg → Videi draudzīgs
Metāls (kuģis): €84000.00; CO₂: 84000.0 kg → Liels CO₂ piesārņojums
Smiltis (kravas_auto): €3000.00; CO₂: 4800.0 kg → Liels CO₂ piesārņojums

Lētākā piegāde ir 'Koks' ar vilciens.
Secinājums: Šis maršruts ir videi draudzīgs
```