L10 Tehtävät

- Tietorakenne sanakirja, dictionary, ja sen käyttö
- Tietorakenne numpy-matriisi ja sen käyttö
- Lajittelu: tuple, sanakirja, oliolista

Tällä viikolla keskitytään uusin tietorakenteisiin. Ohjelmointitehtävissä on oltava otsikkotiedot ja poikkeustenkäsittely.

Poikkeustenkäsittely
Laita ohjelmien tiedostonkäsittelyyn tyyliohjeen mukainen poikkeustenkäsittely eli
`"Tiedoston 'x.txt' käsittelyssä virhe, lopetetaan."`

Sisällysluettelo

L10T1: Sanakirjan peruskäyttö (P)

Tehtävän taso: Perustaso

Tiedostot `L10T1D1.txt` sekä `L10T1D2.txt` sisältävät nopanheittodataa *Dungeons & Dragons®* roolipelistä. Tee ohjelma, joka laskee montako kertaa eri syyn vuoksi on heitetty noppaa.

Tässä tehtävässä lajittelu onnistuu sanakirjan eli dictionary-rakenteen ja `sorted()`-funktion avulla. Lue tiedostosta rivillä oleva heiton syy ja tallenna se listaan. Koko riviä ei tarvitse tallentaa, pelkkä syy riittää tässä tehtävässä. Vie lista analysoivalle aliohjelmalle, joka lajittelee syyt sanakirjaan. Lajittelun voi tehdä esimerkiksi niin, että käyttää tiedostosta saatua syytä sanakirjan avaimena ja syyn esiintymiskertoja avaimen arvona. `sorted()` -funktion avulla saat sanakirjan avaimet aakkosjärjestykseen järjestettynä listana.

Kun lajittelu on suoritettu, tulosta tiedot sekä kirjoita ne tiedostoon alla olevan esimerkin mukaisesti. Tiedostoon kirjoitetaan täsmälleen samat asiat mitä aliohjelma tulostaa näytölle, ts. otsikkorivi ja data-rivit. Kysy tiedostonimet pääohjelmassa ja välitä ne aliohjelmiin parametreina.

Heiton syiden kääntämisessä on käytetty apuna suomenkielistä teosta 'Legendoja ja Lohikäärmeitä' (Mustonen et al., 2022)

Esimerkki syötetiedostosta 'L10T1D1.txt':

```
Sessio;Hahmo;Heiton syy;Noppien määrä;Noppa;Heitot
2.1;Fenian;Hyökkäys;2;d20;17, 15
2.1;Fenian;Vahinko;2;d8;1, 5
2.1;Fenian;Hyökkäys;2;d20;11, 16
2.1;Fenian;Vahinko;2;d8;5, 4
2.1;Fenian;Hiipiminen;1;d20;9
2.1;Fenian;Ketteryyden pelastusheitto;1;d20;2
2.1;Fenian;Ketteryyden pelastusheitto;1;d20;2
2.1;Fenian;Ketteryyden pelastusheitto;1;d20;9
```

```
2.2;Fenian;Hiipiminen;1;d20;5
2.2;Fenian;Tarkkaavaisuus;1;d20;19
```

Esimerkkiajo 1

Syötteet:

```
L10T1D1.txt
L10T1T1.txt
```

Tuloste:

```
Anna luettavan tiedoston nimi: L10T1D1.txt
Anna kirjoitettavan tiedoston nimi: L10T1T1.txt
Tunnistettiin 5 erilaista syytä ja 10 heittoa:
Hiipiminen: 2 heittoa
Hyökkäys: 2 heittoa
Ketteryyden pelastusheitto: 3 heittoa
Tarkkaavaisuus: 1 heitto
Vahinko: 2 heittoa
Kiitos ohjelman käytöstä.
```

Esimerkki tulostiedostosta 'L10T1T1.txt':

```
Tunnistettiin 5 erilaista syytä ja 10 heittoa:
Hiipiminen: 2 heittoa
Hyökkäys: 2 heittoa
Ketteryyden pelastusheitto: 3 heittoa
Tarkkaavaisuus: 1 heitto
Vahinko: 2 heittoa
```

L10T2: Ohjelma osoitekirjan ylläpitoon oliolistalla (P)

Tehtävän taso: Perustaso

Tee Python ohjelma, jolla voit ylläpitää osoitekirjaa. Toteuta osoitekirja niin, että se on lista, jossa on henkilön tiedot sisältäviä olioita. Tässä osoite kirjassa yhdellä henkilöllä on nimi, puhelinnumero ja osoite.

Tee ohjelmaasi aliohjelmat, jolla voit lisätä, poistaa ja tulostaa osoitekirjan tietoja. Lisäävä aliohjelma kysyy tiedot ja lisää puhelinnumeron ja osoitteen olioon ja olion listaan. Henkilön poistavassa aliohjelmassa yksi vaihtoehto on käydä lista läpi, ja jos oikean henkilön tiedot löytyvät, poistaa olio listasta listan `remove()`-jäsenfunktion avulla. Jos poistettavaa ei löydy sanakirjasta, ilmoita siitä käyttäjälle.

Ennen kuin välität tulostavalle tai poistavalle aliohjelmalle osoitekirjan, tarkista ensin, ettei osoitekirja ole tyhjä. Jos osoitekirjassa on dataa, kysy tulostettavan tai poistettavan henkilön nimeä. Mikäli tulostettavaa tai poistettavaa henkilöä ei löydy osoitekirjasta, ilmoita siitä käyttäjälle. Jos taas käyttäjä antaa tyhjän, tulostetaan koko osoitekirja. Henkilöä tulostettaessa tulostetaan kaikki tiedot omille riveilleen esimerkkiajon mukaisesti.

Tulosteet, jotka eivät näy esimerkkiajosta:
`"Henkilöä 'Erkki Esimerkki' ei löydy osoitekirjasta."`
`"Osoitekirja on tyhjä."`

Esimerkkiajo 1

Syötteet:

```
1
Olli Olematon
045 67890
Satukatu 1
1
Satu Hahmo
040 55555
Satukatu 2
3

2
Olli Olematon
0
```

Tuloste:

```
Valitse haluamasi toiminto:
1) Lisää osoitekirjaan
2) Poista osoitekirjasta
3) Tulosta osoitekirja
0) Lopeta
Anna valintasi: 1
Anna lisättävän henkilön tiedot.
Nimi: Olli Olematon
Puhelinnumero: 045 67890
Osoite: Satukatu 1
Henkilö 'Olli Olematon' lisätty osoitekirjaan.

Valitse haluamasi toiminto:
1) Lisää osoitekirjaan
2) Poista osoitekirjasta
3) Tulosta osoitekirja
0) Lopeta
Anna valintasi: 1
Anna lisättävän henkilön tiedot.
Nimi: Satu Hahmo
Puhelinnumero: 040 55555
Osoite: Satukatu 2
Henkilö 'Satu Hahmo' lisätty osoitekirjaan.

Valitse haluamasi toiminto:
1) Lisää osoitekirjaan
2) Poista osoitekirjasta
3) Tulosta osoitekirja
0) Lopeta
Anna valintasi: 3
Anna tulostettavan henkilön nimi tai jätä tyhjäksi:
Osoitekirjassa on seuraavat henkilöt:
Nimi: Olli Olematon
Puhelinnumero: 045 67890
```

```
Osoite: Satukatu 1

Nimi: Satu Hahmo
Puhelinnumero: 040 55555
Osoite: Satukatu 2


Valitse haluamasi toiminto:
1) Lisää osoitekirjaan
2) Poista osoitekirjasta
3) Tulosta osoitekirja
0) Lopeta
Anna valintasi: 2
Anna poistettavan henkilön nimi: Olli Olematon
Henkilö 'Olli Olematon' poistettu osoitekirjasta.

Valitse haluamasi toiminto:
1) Lisää osoitekirjaan
2) Poista osoitekirjasta
3) Tulosta osoitekirja
0) Lopeta
Anna valintasi: 0
Lopetetaan.

Kiitos ohjelman käytöstä.
```

L10T3: NumPy-matriisi (T)

Tehtävän taso: Tavoitetaso

Viikolla 8 asensimme NumPy-kirjaston Pythonin laajennoksena ja tässä tehtävässä tutustutaan NumPy-matriisin perustoimintaan. Jos et ole vielä asentanut NumPy kirjastoa, katso ohjelmointioppaan 8. luvusta, miten standardikirjaston ulkopuolisia kirjastoja asennetaan. NumPy on valmiiksi asennettu automaattitarkastajaan.

Luo 4x4-kokonaislukumatriisi ja alusta se nolliksi NumPyn `zeros()`-jäsenfunktiolla. Sen jälkeen käy läpi matriisin kaikki alkiot kahdella silmukalla, ensin rivit ja sitten sarakkeet, ja sijoita matriisin jokaisen alkion arvoksi sen sarake- ja rivi-indeksin tulo siten, että sarake-indeksiin lisätään kaksi, mutta rivi-indekseihin ei lisätä mitään, ts. (RiviIndeksi) * (SarakeIndeksi+2), ks. esimerkkituloste alla.

Tee tästä lähtömatriisista transpoosi `transpose()`-funktion avulla. Funktio saa parametrina matriisin ja palauttaa siitä transpoosin. Laske yhteen lähtömatriisi ja matriisin transpoosi.

Tulosta lähtömatriisit print-käskyllä, jolloin se noudattaa NumPyn tarjoamaa muotoilua. Tulosta lähtömatriisin ja transpoosin yhteenlaskusta saatu tulosmatriisi riveittäin sarakkeet puolipisteillä eroteltuina alla olevan esimerkkiajon mukaisesti, sillä tämä muoto on helppo siirtää esim. taulukkolaskentaohjelmaan visualisointia varten. Lopuksi muista vapauttaa NumPyn varaama muisti NumPyn omalla `delete()`-käskyllä.

Esimerkkiajo 1

Tuloste:

```
Tämä ohjelma testaa NumPy-matriisin käyttöä.
Lähtömatriisit tulostettuna NumPy-muotoilulla:
Matriisi:
[[ 0  0  0  0]
 [ 2  3  4  5]
 [ 4  6  8 10]
 [ 6  9 12 15]]

Matriisin transpoosi:
[[ 0  2  4  6]
 [ 0  3  6  9]
 [ 0  4  8 12]
 [ 0  5 10 15]]

Tulosmatriisi tulostettuna NumPy-muotoilulla:
[[ 0  2  4  6]
 [ 2  6 10 14]
 [ 4 10 16 22]
 [ 6 14 22 30]]

Tulosmatriisi tulostettuna alkiot puolipisteillä eroteltuna:
0;2;4;6;
2;6;10;14;
4;10;16;22;
6;14;22;30;

Kiitos ohjelman käytöstä.
```