L12 Tehtävät

- Tiedostoformaatit CSV, TXT, JSON, YAML
- Omatoiminen ongelmanratkaisu

Lue oppaan tämän viikon asioita käsittelevä luku 12 ja lisäksi tehtävien suorittamiseen tarvitset aiempien lukujen tietoja. Ohjelmointitehtävissä on oltava otsikkotiedot ja ne palautetaan CodeGradeen.

`Huomaa`, että tähdellä merkityt (*) tehtävät ovat lisätehtäviä, jotka tekemällä saa lisäpisteitä.

L12 tiedostot

Tällä viikolla testattava data tulee kopioida tehtäväpaperista, lukuunottamatta tehtävän L12T5 YAML-formaatissa olevaa tiedostoa. Tämä valmistaa tenttiä varten, sillä tentissä tiedot ovat vain tehtäväpaperissa. Kopioi siis syötetiedosto -laatikossa olevat tiedot IDLEen ja tallenna tiedosto otsikon mukaisella nimellä.

Sisällysluettelo

L12T1: Tiedostoformaatit: CSV (P)

Tehtävän taso: Perustaso

Kurssilla on käytetty usein CSV-tiedostoja eli *Comma Separated Values* -tiedostoja, joissa rivillä on pilkulla erotettuja arvoja. Taulukkolaskentaohjelmat tukevat tällaisten tiedostojen lukemista ja kirjoittamista ja siksi näiden tiedostojen pääte on tyypillisesti `.CSV` (kurssilla on käytetty yleisesti tekstitiedoston päätettä `.TXT`).

Tee Python-ohjelma, joka lukee annetun CSV tiedoston, poimii sieltä tarvittavat sarakkeet ja kirjoittaa ne toiseen tiedostoon. Tiedostot L12T1D1.csv ja L12T1D2.csv sisältävät junien pysähtymispaikkojen ja asemien tietoja Suomen junaverkossa. Lue tiedot ohjelmassasi oliolistaan. Välitä lista pääohjelman kautta kirjoittavalle aliohjelmalle. Kirjoittavassa aliohjelmassa kirjoita vain sarakkeet "tyyppi", "asemanLyhytKoodi" ja "asemanNimi" uuteen tiedostoon (huomaa otsikkorivin kirjoitusasu esimerkkiajossa).

Datan lähde: Digitraffic

Esimerkki syötetiedostosta 'L12T1D1.csv':

```
matkustajaLiikenne;tyyppi;asemanNimi;asemanLyhytKoodi;asemanUICKoodi;maakoodi;pituusaste;leveysaste
False;STATION;Ahonpää;AHO;1343;FI;25.006783;64.537118
False;STATION;Ahvenus;AHV;1000;FI;22.498185;61.291923
True;STOPPING_POINT;Ainola;AIN;628;FI;25.101494;60.456863
False;STATION;Airaksela;ARL;869;FI;27.4295;62.724396
False;STATION;Aittaluoto;ATL;676;FI;21.84537;61.476933
False;STATION;Ajos;AJO;767;FI;24.541716;65.673679
False;STATION;Alapitkä;APT;415;FI;27.535426;63.200823
True;STATION;Alavus;ALV;284;FI;23.600437;62.617769
False;STATION;Alholma;ALH;308;FI;22.695265;63.706765
False;STATION;Arola;ARO;939;FI;29.022513;64.450787
False;STATION;Asola;ASO;1340;FI;25.047287;60.320974
True;STOPPING_POINT;Aviapolis;AVP;1331;FI;24.956191;60.30435
False;STATION;Buslovskaja;BSL;2140;RU;28.3777;60.840338
True;STATION;Dragsvik;DRA;167;FI;23.488607;59.990199
False;TURNOUT_IN_THE_OPEN_LINE;Dynamiittivaihde;DMV;581;FI;23.084138;59.866408
True;STOPPING_POINT;Eläinpuisto-Zoo;EPZ;623;FI;24.191479;62.540677
```

Esimerkkiajo 1

Syötteet:

```
L12T1D1.csv
L12T1T1.csv
```

Tuloste:

```
Anna luettavan tiedoston nimi: L12T1D1.csv
Tiedosto 'L12T1D1.csv' luettu.
Tiedostossa oli 16 asemaa.
Anna kirjoitettavan tiedoston nimi: L12T1T1.csv
Tiedosto 'L12T1T1.csv' kirjoitettu.
Kiitos ohjelman käytöstä.
```

Esimerkki tulostiedostosta 'L12T1T1.csv':

```
Tyyppi;LyhytKoodi;AsemanNimi
STATION;AHO;Ahonpää
STATION;AHV;Ahvenus
STOPPING_POINT;AIN;Ainola
STATION;ARL;Airaksela
STATION;ATL;Aittaluoto
STATION;AJO;Ajos
STATION;APT;Alapitkä
STATION;ALV;Alavus
STATION;ALH;Alholma
STATION;ARO;Arola
STATION;ASO;Asola
STOPPING_POINT;AVP;Aviapolis
STATION;BSL;Buslovskaja
STATION;DRA;Dragsvik
TURNOUT_IN_THE_OPEN_LINE;DMV;Dynamiittivaihde
STOPPING_POINT;EPZ;Eläinpuisto-Zoo
```

L12T2: CSV-tiedoston analysointi (P)

Tehtävän taso: Perustaso

Jatka ja muokkaa edellistä tehtävää (L12T1) lisäämällä siihen tiedoston analyysi. Laske datasta seuraavat tiedot:

1. Asemien yhteismäärä
2. Matkustaja-asemien määrä (eli asemat, jossa `"matkustajaLiikenne"` -kenttä on tosi ja niiden tyyppi on `"STATION"`)
3. Matkustajakäytössä olevat pysähdyspaikat (eli jossa eli asemat, jossa `"matkustajaLiikenne"` -kenttä on tosi ja niiden tyyppi on `"STOPPING_POINT"`)
4. Kaikki asemat ja pysähdyspaikat, jotka sijaitsevat idässä, eli niiden pituusaste on suurempi kuin 26.
5. Kaikki asemat ja pysähdyspaikat, jotka sijaitsevat etelässä, eli niiden leveysaste on pienempi kuin 61.30.

Kirjoita analyysin tulokset tulostiedostoon. Katso tulosten muotoilu esimerkkiajosta.

Datan lähde: Digitraffic

Esimerkkiajo 1

Syötteet:

```
L12T1D1.csv
L12T2T1.txt
```

Tuloste:

```
Anna luettavan tiedoston nimi: L12T1D1.csv
Tiedosto 'L12T1D1.csv' luettu.
Tiedostossa oli 16 asemaa.
Tiedot analysoitu.
Anna kirjoitettavan tiedoston nimi: L12T2T1.txt
Tiedosto 'L12T2T1.txt' kirjoitettu.
Kiitos ohjelman käytöstä.
```

Esimerkki tulostiedostosta 'L12T2T1.txt':

```
Datassa oli 16 aseman tiedot.
Matkustaja-asemia oli 2 kpl.
Matkustajakäytössä olevia pysähdyspaikkoja oli 3 kpl.
Itäisessä Suomessa sijaitsevia asemia oli 4 kpl.
Eteläisessä Suomessa sijaitsevia asemia oli 7 kpl.
```

L12T3: Tiedostoformaatit: binääritiedostot (P)

Tehtävän taso: Perustaso

Binääritiedostot on yleinen datan tallennukseen käytetty formaatti, jossa data on tallennettu binäärimuodossa tiedostoon. Tällaisia tiedostoja pystyy yleensä lukemaan vain ne tehneellä ohjelmalla. Lisätietoa binääritiedostoista on oppaassa.

Muokkaa tehtävää L12T1. Muokkaa kirjoitusaliohjelmaa niin, että normaalin tekstitiedoston sijaan kirjoitat koko oliolistan kerralla binäärimuodossa tiedostoon. Käytä tähän `pickle` -kirjastoa ja oppaan esimerkkejä.

Esimerkkiajo 1

Syötteet:

```
L12T1D1.csv
L12T3T1
```

Tuloste:

```
Anna luettavan tiedoston nimi: L12T1D1.csv
Tiedosto 'L12T1D1.csv' luettu.
Tiedostossa oli 16 asemaa.
Anna kirjoitettavan tiedoston nimi: L12T3T1
Tiedosto 'L12T3T1' kirjoitettu.
Kiitos ohjelman käytöstä.
```

L12T4: Tiedostoformaatit: JSON (*) (P)

Tehtävän taso: Perustaso

Erityisesti rajapinnoissa ja tietoliikenteessä sekä tiedon tallennuksessa yleiseti käytetty, helposti koneluettava tiedostoformaatti on JSON eli *JavaScript Object Notation*. JSON rakentuu tarkasti määritellystä rakenteesta, jossa jokaisen tietoalkion yhteydessä on sitä vastaava muuttujan nimi.

Pythonissa JSON rakentuu yleensä sanakirjoista ja listoista (perustietotyyppien kuten str, int ja float lisäksi). Pythonin standardikirjastossa on valmiina `json` -kirjasto, jolla JSON-muotoista dataa on helppo lukea ja kirjoittaa.

Tee Python-ohjelma, joka lukee JSON-muotoista opinnäytetöiden metadataa. Lajittele jokaisesta opinnäytetyöstä löytyvät avainsanat (keywords) yhteen sanakirjaan. Vie sanakirja kirjoittavalle aliohjelmalle, joka kirjoittaa sen JSON-muotoisena tiedostoon.

HUOM. Älä käytä tässä tehtävässä ohjelmointivideolla näytettyä `jsonpickle`-kirjastoa, sillä se tekee asiat eri tavalla kuin tässä tehtävässä halutaan.

Käytä tiedoston lukemiseen joko `load()` tai `loads()` -funktioita. Näiden ero on se, että `load()` ottaa parametrina tiedostokahvan ja `loads()` ottaa merkkijonon. Jos käytät `loads()` -funktiota, lue tiedosto käyttäen `read()` -funktiota. JSON-muotoisen tiedoston lukeminen rivi kerrallaan johtaa lähes varmasti virheeseen datan formaatista johtuen.

Tiedoston kirjoittamiseen vaihtoehdot ovat `dump()` ja `dumps()`. Näiden ero on sama kuin yllä, eli `dump()` ottaa parametrina tallennettavan datan ja tiedostokahvan, kun taas `dumps` ottaa pelkästään tallennettan datan ja palauttaa JSON-muotoisen merkkijonon, jonka voi normaalisti kirjoittaa käyttämällä `write()` -funktiota. **HUOM.** Käytä molempien kirjoittavien funktioiden yhteydessä

parametreja `ensure_ascii=False`, `indent=4` ja `sort_keys=True`, jotta tiedosto saadaan ihmisluettavaan muotoon ja merkit tulostuvat oikein ja oikeassa järjestyksessä tarkastusta varten.

Datan lähde: Turun ammattikorkeakoulu

Esimerkki syötetiedostosta 'L12T4D1.json':

```
[{"studyEntitlementDays": 1127, "studyCredits": 250, "grade": 5, "supervisorId": 5,
"pages": 52, "words": 12018, "onlineReferences": 30, "printedReferences": 34,
"weakReferences": 2, "keywordAppearances": 200, "keywords": ["zephyr",
"power management", "rtos", "embedded systems", "iot"]}]
```

Esimerkkiajo 1

Syötteet:

```
L12T4D1.json
L12T4T1.json
```

Tuloste:

```
Anna luettavan tiedoston nimi: L12T4D1.json
Tiedosto 'L12T4D1.json' luettu.
Anna kirjoitettavan tiedoston nimi: L12T4T1.json
Tiedosto 'L12T4T1.json' kirjoitettu.
Kiitos ohjelman käytöstä.
```

Esimerkki tulostiedostosta 'L12T4T1.json':

```
{
    "embedded systems": 1,
    "iot": 1,
    "power management": 1,
    "rtos": 1,
    "zephyr": 1
}
```

L12T5: Tiedostoformaatit: YAML (*) (T)

Tehtävän taso: Tavoitetaso

JSON-formaatin rinnalla toinen yleinen formaatti on YAML (alunperin `Yet Another Markup Language`, nykyään `YAML Ain't Markup Language`). YAML'ssa on JSON'in tapaan tarkasti määritelty rakenne, joka ei kuitenkaan koostu aaltosulkeista, vaan data ja muuttujat ryhmitellään yleisesti viivojen ja välilyöntien avulla.

Pythonin standardikirjastossa ei valmista kirjastoa YAML'n lukemiseen löydy, mutta sellainen löytyy asennettavaksi pip'in kautta:

```
python -m pip install pyyaml
```

Tämän jälkeen kirjaston voi tuoda ohjelmaan käyttämällä `import yaml` -käskyä.

Tee Python ohjelma, joka lukee ja kirjoittaa sekä YAML, että JSON tiedostoja. Tässä tehtävässä data luetaan ja kirjoitetaan suoraan formaatista toiseen. Tee yksi aliohjelma lukemiselle ja toinen kirjoittamiselle. Tarkista aliohjelmissa kumpi tiedostoformaatti on kyseessä käyttäen tiedostopäätettä avuksi. Tässä tehtävässä voi olettaa, että tiedoston nimessä on vain yksi piste. Käytä sen jälkeen oikeaa kirjastoa tiedon lukemiseen ja kirjoittamiseen. Huomaa, että YAML tiedostojen pääte voi olla joko `.YML` tai `.YAML`. Tiedostoina käytetään samoja tiedostoja kuin edellisessä tehtävässä, lisänä YAML-muotoinen L12T5D1.yml.

YAML kirjastossa on monia funktioita lukemiseen ja kirjoittamiseen, mutta meille riittää tässä tehtävässä funktiot `safe_load()` sekä `safe_dump()`. Yleisesti on suositeltavaa käyttää näitä `safe` -alkuisia funktiota datan lukemiseen ja kirjoittamiseen, koska näin vältymme tietyiltä tietoturvariskeiltä erityisesti silloin, jos data olisi meille tuntemattomasta lähteestä.

Funktio `safe_load()` ottaa parametrikseen pelkän tiedostokahvan ja palauttaa datan sopiviksi Python-tietotyypeiksi muutettuna. Samoin kuin JSON-tiedostojen kanssa, YAML'ssa tämä tarkoittaa usein sanakirjoja ja listoja. `safe_dump()` -funktio taas tarvitsee parametrikseen tallennettavan tiedon sekä tiedostokahvan. Käytä tallennuksessa lisäksi parametreja `allow_unicode=True` sekä `indent=4`. Näin saamme datan taas ihmisluettavaan muotoon. Huomaa, että YAML kirjasto järjestää datassa olevat muuttujat oletuksena aakkosjärjestykseen. Käytä JSON -tiedostoa kirjoittaessa samoja parametreja kuin edellisessä tehtävässä.

Datan lähde: Turun ammattikorkeakoulu

Esimerkki syötetiedostosta 'L12T4D1.json':

```
[{"studyEntitlementDays": 1127, "studyCredits": 250, "grade": 5, "supervisorId": 5,
"pages": 52, "words": 12018, "onlineReferences": 30, "printedReferences": 34,
"weakReferences": 2, "keywordAppearances": 200, "keywords": ["zephyr",
"power management", "rtos", "embedded systems", "iot"]}]
```

Esimerkki syötetiedostosta 'L12T5D1.yml':

```
-   grade: 5
    keywordAppearances: 200
    keywords:
    - zephyr
    - power management
    - rtos
    - embedded systems
    - iot
    onlineReferences: 30
    pages: 52
    printedReferences: 34
    studyCredits: 250
    studyEntitlementDays: 1127
    supervisorId: 5
    weakReferences: 2
    words: 12018
```

Esimerkkiajo 1

Syötteet:

```
L12T4D1.json
L12T5T1.yaml
```

Tuloste:

```
Anna luettavan tiedoston nimi: L12T4D1.json
Tiedosto 'L12T4D1.json' luettu.
Anna kirjoitettavan tiedoston nimi: L12T5T1.yaml
Tiedosto 'L12T5T1.yaml' kirjoitettu.
```

Esimerkki tulostiedostosta 'L12T5T1.yaml':

```yaml
-   grade: 5
    keywordAppearances: 200
    keywords:
    - zephyr
    - power management
    - rtos
    - embedded systems
    - iot
    onlineReferences: 30
    pages: 52
    printedReferences: 34
    studyCredits: 250
    studyEntitlementDays: 1127
    supervisorId: 5
    weakReferences: 2
    words: 12018
```

Esimerkkiajo 2

Syötteet:

```
L12T5D1.yml
L12T5T2.json
```

Tuloste:

```
Anna luettavan tiedoston nimi: L12T5D1.yml
Tiedosto 'L12T5D1.yml' luettu.
Anna kirjoitettavan tiedoston nimi: L12T5T2.json
Tiedosto 'L12T5T2.json' kirjoitettu.
```

Esimerkki tulostiedostosta 'L12T5T2.json':

```json
[
    {
        "grade": 5,
        "keywordAppearances": 200,
        "keywords": [
            "zephyr",
            "power management",
            "rtos",
            "embedded systems",
```

```
            "iot"
        ],
        "onlineReferences": 30,
        "pages": 52,
        "printedReferences": 34,
        "studyCredits": 250,
        "studyEntitlementDays": 1127,
        "supervisorId": 5,
        "weakReferences": 2,
        "words": 12018
    }
]
```