L09 Tehtävät

- Poikkeustenkäsittely tiedostonkäsittelyn ja käyttäjäsyötteiden kanssa
- Algoritmi datan luokitteluun ryhmiin

Lue oppaan tämän viikon asioita käsittelevä luku 9. Lisäksi tehtävien suorittamiseen tarvitset aiempien lukujen tietoja. Ohjelmointitehtävissä on oltava otsikkotiedot.

Sisällysluettelo

L09T1: Poikkeustenkäsittely tiedostonkäsittelyn yhteydessä (M)

Tehtävän taso: Minimitaso

Tee ohjelma, joka lukee tekstitiedoston listaan yhdessä aliohjelmassa ja kirjoittaa listan tiedostoon toisessa aliohjelmassa. Pääohjelmassa tulee olla listan määrittely, tiedostonimien kysyminen ja luku- sekä kirjoitusaliohjelmakutsut. Molemmissa aliohjelmissa tulee olla tiedostonkäsittelyyn liittyvät poikkeustenkäsittelyt eli tiedoston avaaminen, lukeminen/kirjoittaminen ja sulkeminen tulee tehdä poikkeustenkäsittelijän sisällä, jotta mahdolliset virhetilanteet saadaan käsiteltyä hallitusti. Luettavan ja kirjoitettavan tiedoston rakenne on sama: yksittäisiä kokonaislukuja aina yksi luku rivillä.

Ohjelman ydin on tuttu ja varsin lyhyt, mutta poikkeustenkäsittely laajentaa ohjelmaa. Mikäli tiedoston käsittely ei onnistu, ei siitä yritetä toipua vaan käyttäjälle kerrotaan alla olevalla virheilmoituksella tilanteesta ja lopetetaan ohjelman suoritus sys.exit(0) –käskyllä. Heittomerkkien sisällä tulee olla ongelman aiheuttaneen tiedoston nimi: **`Tiedoston 'x.txt' käsittelyssä virhe, lopetetaan.`**

Huomaa, että hakemistorakenteet näyttävät erilaisilta Linuxissa ja Windowsissa, mutta Macin hakemistorakenteet vastaavat Linuxia. Tee ja testaa ohjelmasi IDLEssä esimerkkiajon mukaisesti ja kiinnitä huomiota tiedostonimien esitysasuun tarpeen mukaan CodeGradessa. Windows-koneessa tiedoston kirjoittaminen epäonnistuu tyypillisesti viitattaessa levyyn, jota ei ole käytössä. Esim. tiedostonimi '`P:/eivoikirjoittaa.txt`' johtaa tyypillisesti Windows-koneessa virhe-ilmoitukseen ja jos ei, niin P:n tilalla voi kokeilla jotain muita kirjaimia (tai levyn nimiä), joita ei ole määritelty käytettävässä tietokoneessa. Linuxissa, ja CodeGradessä, tiedoston kirjoitusvirhe saadaan tyypillisesti aikaan `/var/eivoikirjoittaa.txt` -tyylisellä tiedostonimellä.

Esimerkki syötetiedostosta 'L09T1D1.txt':

```
22
39
26
2
12
```

```
4
3
39
2
24
```

Esimerkkiajo 1

Syötteet:

```
P:/eiole.txt
```

Tuloste:

```
Anna luettavan tiedoston nimi: P:/eiole.txt
Tiedoston 'P:/eiole.txt' käsittelyssä virhe, lopetetaan.
```

Esimerkkiajo 2

Syötteet:

```
L09T1D1.txt
/var/eionnistu.txt
```

Tuloste:

```
Anna luettavan tiedoston nimi: L09T1D1.txt
Tiedosto 'L09T1D1.txt' luettu.
Anna kirjoitettavan tiedoston nimi: /var/eionnistu.txt
Tiedoston '/var/eionnistu.txt' käsittelyssä virhe, lopetetaan.
```

L09T2: Poikkeustenkäsittely käyttäjäsyötteiden yhteydessä (P)

Tehtävän taso: Perustaso

Tee ohjelma, jolla voit testata erilaisia poikkeuksia käyttäjän antamien syötteiden yhteydessä, erityisesti ValueError, IndexError, ZeroDivisionError ja TypeError. Toteuta ohjelmasi normaalin pääohjelman ja valikko-ohjelman avulla, joiden lisäksi kannattaa tehdä oma aliohjelma jokaisen virheen testaamiseen ValueErroria lukuun ottamatta. ValueError testaus kannattaa tehdä valikko-ohjelmassa kun käyttäjän antama syöte muutetaan kokonaisluvuksi – jos se ei onnistu, informoi käyttäjää tilanteesta ja pyydä uutta valintaa.

Kolmessa muussa virheentestausaliohjelmassa kannattaa käyttää rakennetta, jossa ensin pyritään tekemään haluttu operaatio tyypillisesti 2 koodirivillä ja jos se ei onnistu, mennään poikkeuksenkäsittelijään. Poikkeustenkäsittelijän tulee huomioida vain aliohjelman testaama poikkeus. IndexError tulee tyypillisesti, kun viitataan listaan indeksillä, jota ei ole. Näin ollen tee ko. aliohjelmaan lista, jossa on alkioina 11, 22, 33, 44 ja 55. Jakolaskun yhteydessä pyöristä tulos

kahteen desimaaliin. Ja tyyppivirhe tulee esimerkiksi silloin, kun yrittää kertoa kahta merkkijonoa keskenään eli unohtaa muuttaa käyttäjän syötteen merkkijonosta numeroksi.

Mikäli toiminto onnistuu, tulostetaan alla olevat tulosteet:
`"Listan arvo on 11 indeksillä 0."`
`"12/6 on 2."`

Esimerkkiajo 1

Syötteet:

```
1
auto
2
5
3
0
4
6
0
```

Tuloste:

```
Mitä haluat tehdä:
1) Testaa ValueError
2) Testaa IndexError
3) Testaa ZeroDivisionError
4) Testaa TypeError
0) Lopeta
Valintasi: 1
Valikko-ohjelma testaa ValueError'n.

Mitä haluat tehdä:
1) Testaa ValueError
2) Testaa IndexError
3) Testaa ZeroDivisionError
4) Testaa TypeError
0) Lopeta
Valintasi: auto
Virhe, valinta ei ollut kokonaisluku.
Valintasi: 2
Anna indeksi 0-4: 5
Tuli IndexError, indeksi oli 5.

Mitä haluat tehdä:
1) Testaa ValueError
2) Testaa IndexError
3) Testaa ZeroDivisionError
4) Testaa TypeError
0) Lopeta
Valintasi: 3
Anna jakaja: 0
Tuli ZeroDivisionError, jakaja oli 0.

Mitä haluat tehdä:
1) Testaa ValueError
2) Testaa IndexError
3) Testaa ZeroDivisionError
```

```
4) Testaa TypeError
0) Lopeta
Valintasi: 4
Anna numero: 6
Tuli TypeError, 6*6 merkkijonoilla ei onnistunut.

Mitä haluat tehdä:
1) Testaa ValueError
2) Testaa IndexError
3) Testaa ZeroDivisionError
4) Testaa TypeError
0) Lopeta
Valintasi: 0
Lopetetaan

Kiitos ohjelman käytöstä.
```

L09T3: Valikkopohjainen ohjelma virheen käsittelyllä (P)

Tehtävän taso: Perustaso

Tee Python-ohjelma, joka laskee käyttäjän haluaman päivän tuulen nopeuden keskiarvon annetusta datasta. Ohjelmassasi tulee olla normaali kurssilla käytetty valikko sekä kirjastotiedostossa aliohjelmat tiedoston lukemiselle, tiedoston nimen kysymiselle, päivän kysymiselle sekä analysoinnille.

Lue tiedostossa olevat tiedot oliolistaan ja välitä se pääohjelman kautta analysoivalle aliohjelmalle. Kysy käyttäjältä päivämäärä muodossa 'DD.MM.YYYY" ja muuta se `strptime()` funktion avulla aikarakenteeksi. Välitä aikarakenne ja luetut tiedot analysoivalle aliohjelmalle. Laske analysoivassa ohjelmassa vain käyttäjän antaman päivän datasta tuulen nopeuden keskiarvo ja tulosta tulos. Analysoitujen alkioden määrä on vain käyttäjän antaman päivän data-alkiot. Analysoinnissa kannattaa hyödyntää aikarakenteen jäsenmuuttujia, kuten `tm_year`.

Muista lisätä ohjelmaasi myös valikon sekä tiedostonkäsittelyn virheenkäsittelyt. Valikossa tulee tarkistaa ennen analysointia, että dataa on olemassa. Jos sitä ei löydy, tai datasta ei löydy käyttäjän syöttämää päivää, tulostetaan alla olevat virheilmoitukset:
`"Annettua päivää ei löydy syötetystä datasta."`
`"Ei analysoitavia tietoja, lue tiedot ensin."`

Esimerkki syötetiedostosta 'L09T3D1.txt':

```
Havaintoasema;Aika [Paikallinen aika];Keskituulen nopeus keskiarvo [m/s]
Lappeenranta lentoasema;01.11.2023 00:00;10
Lappeenranta lentoasema;01.11.2023 01:00;10.1
Lappeenranta lentoasema;01.11.2023 02:00;8.3
Lappeenranta lentoasema;01.11.2023 03:00;8.6
Lappeenranta lentoasema;01.11.2023 04:00;8.3
Lappeenranta lentoasema;01.11.2023 05:00;8.4
Lappeenranta lentoasema;01.11.2023 06:00;7.5
Lappeenranta lentoasema;01.11.2023 07:00;7.4
Lappeenranta lentoasema;01.11.2023 08:00;6.6
Lappeenranta lentoasema;01.11.2023 09:00;6.9
```

```
Lappeenranta lentoasema;01.11.2023 10:00;6.5
Lappeenranta lentoasema;01.11.2023 11:00;6.4
Lappeenranta lentoasema;01.11.2023 12:00;6.4
Lappeenranta lentoasema;01.11.2023 13:00;5.8
Lappeenranta lentoasema;01.11.2023 14:00;5.2
Lappeenranta lentoasema;01.11.2023 15:00;3.2
Lappeenranta lentoasema;01.11.2023 16:00;2.3
Lappeenranta lentoasema;01.11.2023 17:00;2.5
Lappeenranta lentoasema;01.11.2023 18:00;3.8
Lappeenranta lentoasema;01.11.2023 19:00;5.2
Lappeenranta lentoasema;01.11.2023 20:00;5.8
Lappeenranta lentoasema;01.11.2023 21:00;5.9
Lappeenranta lentoasema;01.11.2023 22:00;5.1
Lappeenranta lentoasema;01.11.2023 23:00;4.5
```

Esimerkkiajo 1

Syötteet:

```
1
L09T3D1.txt
2
1.11.2023
0
```

Tuloste:

```
Valitse haluamasi toiminto:
1) Lue tiedosto
2) Analysoi
0) Lopeta
Anna valintasi: 1
Anna luettavan tiedoston nimi: L09T3D1.txt
Tiedosto 'L09T3D1.txt' luettu.
Tiedostosta lisättiin 24 datariviä listaan.

Valitse haluamasi toiminto:
1) Lue tiedosto
2) Analysoi
0) Lopeta
Anna valintasi: 2
Anna analysoitava päivä muodossa 'DD.MM.YYYY': 1.11.2023
Analyysi suoritettu, 24 alkiota analysoitu.
Tuulen nopeuden keskiarvo oli 6.3 metriä sekunnissa 01.11.2023.

Valitse haluamasi toiminto:
1) Lue tiedosto
2) Analysoi
0) Lopeta
Anna valintasi: 0
Lopetetaan.

Kiitos ohjelman käytöstä.
```

L09T4: Poikkeustenkäsittely kirjastofunktioille (T)

Tehtävän taso: Tavoitetaso

Moodlesta saatava `L09T4Kirjasto.py` -moduuli sisältää valmiiksi tehtyjä funktioita. Funktioilla ei kuitenkaan ole virheenkäsittelyä. Tee Python-ohjelma, joka kutsuu kirjastossa olevia funktioita. Laita funktiokutsut `try...except` -rakenteen sisään ja anna rakenteelle virhetyypiksi funktion mahdollisesti aikaansaama virhe alla olevan listan mukaisesti. Kirjaston funktioissa käytetään joitakin asioita, joita tällä kurssilla ei käydä, joten funktioiden täysi ymmärtäminen ei ole tarpeen tässä tehtävässä, vaan tarkoituksena on opetella virheenkäsittelyä kun käytetään itselle tuntemattomia kirjastoja. Usein esimerkiksi standardikirjastojen koodin tarkasteleminen voi olla vaikeaa, joten kirjastojen dokumentaatioon kannattaa sen sijaan paneutua.

Huomaa, että osan kirjaston funktioiden antamista virheistä voi ennaltaehkäistä valintarakenteella (esim. nollalla jako) ennen kuin kirjastofunktiota kutsutaan, joten `try...except` rakenteen käyttö ei välttämättä ole pakollista. Osaan virheistä ei kuitenkaan voi valintarakenteella varautua, jolloin tulee käyttää virheenkäsittelyä.

Tuo kirjasto normaaliin tapaan `import` -käskyllä, mutta varaudu mahdollisuuteen, että kirjasto ei olekaan saatavilla, jolloin Python antaa virheeksi `ImportError`. Jos näin tapahtuu, tulee tästä ilmoittaa käyttäjälle ja poistua ohjelmasta hallitusti `sys.exit(0)` -käskyllä.

Kirjastossa on seuraavat funktiot:

1. **testaaPienempi(x)**: Funktion parametri on luku, ja funktio testaa on luku pienempi kuin 10. Funktio hyväksyy vain positiivisia lukuja ja antaa virheen `AssertationError` jos parametri on negatiivinen luku tai nolla.
2. **jaaLuku(x):** Funktion parametri on luku. Funktio jakaa luvun 10 annetulla luvulla. Funktio ei tarkista onko luku nolla, joten mahdollinen virhe on `ZeroDivisionError`.
3. **arvoListalta(x):** Funktio palauttaa kirjastossa kiintoarvona olevalta listalta arvon funktion parametrina annetun indeksin avulla. Koska funktio ei tarkista onko indeksi alle tai yhtäsuuri kuin listan pituus, mahdollinen virhe on `IndexError`.
4. **listanSatunnainenArvo():** Tämä funktio palauttaa kirjastossa kiintoarvona olevalta listalta satunnaisen arvon. Toinen ohjelmoija on kuitenkin tehnyt kepposen kirjaston tehneelle ohjelmoijalle, ja lisännyt koodinpätkän, joka muuttaa indeksin satunnaisesti merkkijonoksi. Tällöin mahdollinen virhe on `TypeError`.

Ohjelman oikeat tulosteet näkyvät esimerkkiajossa. Mahdollisissa virhetilanteissa annetaan seuraavat virheilmoitukset:

```
"Virhe, kirjastoja ei voitu tuoda, lopetetaan."
"Virhe, valinta ei ollut kokonaisluku."
"Virhe, annoit negatiivisen luvun."
"Virhe, nollalla ei voi jakaa."
"Virhe, listalla ei ollut annettua indeksiä."
"Virhe, listan indeksi oli merkkijono."
```

Esimerkkiajo 1

Syötteet:

```
1
8
2
2
```

```
3
0
4
0
```

Tuloste:

```
Valitse haluamasi toiminto:
1) Testaa testaaPienempi
2) Testaa jaaLuku
3) Testaa arvoListalta
4) Testaa listanSatunnainenArvo
0) Lopeta
Anna valintasi: 1
Anna testattava luku: 8
Antamasi luku on pienempi kuin 10.

Valitse haluamasi toiminto:
1) Testaa testaaPienempi
2) Testaa jaaLuku
3) Testaa arvoListalta
4) Testaa listanSatunnainenArvo
0) Lopeta
Anna valintasi: 2
Anna jakaja: 2
10/2 on 5.0.

Valitse haluamasi toiminto:
1) Testaa testaaPienempi
2) Testaa jaaLuku
3) Testaa arvoListalta
4) Testaa listanSatunnainenArvo
0) Lopeta
Anna valintasi: 3
Anna haettava indeksi: 0
Listan indeksillä 0 löytyi arvo 'Appelsiini'.

Valitse haluamasi toiminto:
1) Testaa testaaPienempi
2) Testaa jaaLuku
3) Testaa arvoListalta
4) Testaa listanSatunnainenArvo
0) Lopeta
Anna valintasi: 4
Listalta löytyi arvo 'Banaani'.

Valitse haluamasi toiminto:
1) Testaa testaaPienempi
2) Testaa jaaLuku
3) Testaa arvoListalta
4) Testaa listanSatunnainenArvo
0) Lopeta
Anna valintasi: 0
Lopetetaan.

Kiitos ohjelman käytöstä.
```