Ohjelmoinnin perusteet Y Python

T-106.1208

2.3.2011

Kertausta: tiedoston avaaminen

- Kun ohjelma haluaa lukea tai kirjoittaa tekstitiedostoon, on ohjelmalle kerrottava, mikä fyysinen tiedosto vastaa ohjelmassa käytettyä tiedostomuuttujaa.
- Samalla käyttöjärjestelmäpuolella varaudutaan käsittelemään ko. tiedostoa.
- Tätä kutsutaan *tiedoston avaamiseksi*.
- Esimerkki tiedoston avaamisesta

```
tiedostomuuttuja = open("teksti.txt","r")
```

- Ensimmäinen parametri on käsiteltävän tiedoston nimi käyttöjärjestelmässä. Jos tiedosto ei ole samassa hakemistossa kuin missä ohjelmaa ajetaan, on nimeen sisällytettävä polku tiedoston hakemistoon.
- Toinen parametri kertoo tiedoston käsittelytavan. Arvo `"r"` kertoo, että tiedosto avataan lukemista varten.

Kertausta: rivin lukeminen ja tiedoston sulkeminen

- Jos muuttuja `tiedostomuuttuja` viittaa lukemista varten avattuun tiedostoon, niin siitä voi lukea rivin kerrallaan metodin `readline` avulla seuraavasti:

  ```
  luettu_rivi = tiedostomuuttuja.readline()
  ```

  Luettu rivi sisältää myös sen lopussa olevan rivinvaihtomerkin. Sen voi poistaa esimerkiksi metodilla `rstrip`.
- Seuraava `readline`-käsky lukee tiedoston seuraavan rivin jne.
- Jos tiedosto on jo luettu loppuun ja kutsutaan `readline`-metodia, se palauttaa arvona tyhjän merkkijonon `""`
- Kun tiedoston lukeminen päättyy, tiedosto pitää sulkea `close`-käskyllä:

  ```
  tiedostomuuttuja.close()
  ```

Kertausta: parannettu versio tiedostonlukuohjelmasta

```
def main():
    nimi = raw_input("Anna luettavan tiedoston nimi: ")
    try:
        lahtotiedosto = open(nimi, "r")
        rivi = lahtotiedosto.readline()
        while rivi != "":
            rivi = rivi.rstrip()
            print rivi
            rivi = lahtotiedosto.readline()
        lahtotiedosto.close()
    except IOError:
        print "Virhe tiedoston", nimi, \
            "lukemisessa. Ohjelma paattyy."

main()
```

Kertausta: Rivin lukeminen for-käskyllä ja rivin etsiminen

```
def main():
    nimi = raw_input("Anna tiedoston nimi: ")
    etsittava_nimi = raw_input("Anna etsittava nimi: ")
    loytyi = False
    try:
        lahtotiedosto = open(nimi, "r")
        for rivi in lahtotiedosto:
            rivi = rivi.rstrip()
            if rivi == etsittava_nimi:
                loytyi = True
        lahtotiedosto.close()
        if loytyi:
            print "Nimi", etsittava_nimi, "loytyi."
        else:
            print "Nimea", etsittava_nimi, "ei loytynyt."
```

Rivin etsiminen, koodi jatkuu

```
    except IOError:
        print "Virhe tiedoston", nimi, \
            "lukemisessa. Ohjelma paattyy."


main()
```

Kaikkien rivien lukeminen yhdellä käskyllä

- Tiedostosta voidaan myös lukea kaikki (jäljellä olevat) rivit metodilla `readlines`.
- Metodi palauttaa listan, joka sisältää tiedoston eri rivit merkkijonoina.
- Rivit sisältävät myös rivinvaihtomerkin.
- Seuraavan kalvon esimerkkiohjelma lukee tiedoston rivit listaan ja tulostaa ne.

Rivien lukeminen yhdellä käskyllä: koodi

```
def main():
    nimi = raw_input("Anna luettavan tiedoston nimi: ")
    try:
        lahtotiedosto = open(nimi, "r")
        rivilista = lahtotiedosto.readlines()
        lahtotiedosto.close()
        for rivi in rivilista:
            rivi = rivi.rstrip()
            print rivi
    except IOError:
        print "Virhe tiedoston", nimi, \
            "lukemisessa. Ohjelma paattyy."


main()
```

Rivien jakaminen

- Jos tiedostosta luettu rivi pitää jakaa osiin, voidaan käyttää `split`-metodia.
- Seuraava esimerkkiohjelma lukee tiedoston, joka sisältää nimiä niin, että samalla rivillä on etu- ja sukunimi toisistaan välilyönnillä erotettuna.
- Ohjelma tulostaa luetut nimet toisinpäin (sukunimi ennen etunimeä).
- Jos joku riveistä ei muodostu kahdesta osasta, ohjelma ilmoittaa virheestä, mutta jatkaa tiedoston lukemista seuraavalta riviltä.

Rivien jakaminen, koodi

```
def main():
    nimi = raw_input("Anna luettavan tiedoston nimi: ")
    try:
        lahtotiedosto = open(nimi, "r")
        rivinro = 0
        for rivi in lahtotiedosto:
            rivinro +=1
            rivi = rivi.rstrip()
            osat = rivi.split()
            if len(osat) != 2:
                print "Virhe rivilla", rivinro
            else:
                print osat[1], osat[0]
        lahtotiedosto.close()
```

Rivien jakaminen, koodi jatkuu

```
    except IOError:
        print "Virhe tiedoston", nimi, \
            "lukemisessa. Ohjelma paattyy."


main()
```

Lukujen lukeminen tiedostosta

- Kun tiedostosta luetaan lukuja, pitää luetut rivit muuttaa tyypinmuunnoksella oikeantyyppiksiksi.
- Jos riviä ei voida muuttaa luvuksi, aiheutuu `ValueError`, joka on syytä käsitellä.
- Jos samalla rivillä on useita lukuja, pitää rivi ensin jakaa. Tyypinmuunnos tehdään vasta jaon tuloksena syntyneille luvuille.
- Seuraava esimerkkiohjelma lukee tiedostosta lämpötiloja ja laskee niiden keskiarvon. Kukin lämpötila on annettu omalla rivillään. Jos tiedostossa on virhe, ohjelma ilmoittaa virheestä ja lopettaa toimintansa.

Desimaalilukuja tiedostosta, koodi

```
def main():
    nimi = raw_input("Mista tiedostosta lampotilat luetaan: ")
    summa = 0.0
    lkm = 0
    try:
        lampotiedosto = open(nimi, "r")
        for rivi in lampotiedosto:
            rivi = rivi.rstrip()
            lampotila = float(rivi)
            summa += lampotila
            lkm += 1
        lampotiedosto.close()
```

Desimaalilukuja tiedostosta, koodi jatkuu

```
        if lkm == 0:
            print "Tiedostossa ei ollut yhtaan lampotilaa."
        else:
            keskiarvo = summa / lkm
            print "Lampotilojen keskiarvo on", keskiarvo
    except IOError:
        print "Virhe tiedoston", nimi, \
            "lukemisessa. Ohjelma paattyy."
    except ValueError:
        print "Virheellinen rivi tiedostossa", nimi, \
            ". Ohjelma paattyy."


main()
```

Tiedostoon kirjoittaminen: tiedoston avaaminen

- Tiedosto on avattava myös silloin, kun tiedostoon halutaan kirjoittaa. Käsitettelytapa on kuitenkin eri kuin tiedostoon kirjoitettaessa.

```
tiedostomuuttuja = open("teksti.txt","w")
```

Tai

```
tiedostomuuttuja = open("teksti.txt","a")
```

- Käsittelytapojen ero: `"w"` kirjoittaa olemassaolevan tiedoston päälle (vanha sisältö häviää kokonaan), `"a"` kirjoittaa olemassaolevan tiedoston loppuun.

Tiedostoon kirjoittaminen: rivin kirjoittaminen

- Tiedostoon voi tulostaa rivin `write`-metodilla. Se ei lisää rivinvaihtomerkkiä, vaan merkki on lisättävä kirjoitettavaan riviin.

```
tiedostomuuttuja.write("Kirjoitettava rivi\n")
```

- Metodilla `write` voi tulostaa tiedostoon vain merkkijonoja. Esimerkiksi luvut pitää muuttaa ennen tulostamista merkkijonoiksi joko `str`-tyypinmuunnoksella tai käyttämällä tulostuksen muotoilua.

```
kanta = 3.5
ekspo = 5
tulos = kanta ** ekspo
tiedostomuuttuja.write("%.2f potenssiin %d on %.2f\n" %\
    (kanta, ekspo, tulos))
tiedostomuuttuja.write(str(kanta) + " potenssiin " + \
    str(ekspo) + " on " + str(tulos) + "\n")
```

Virheiden käsittely ja tiedoston sulkeminen

- Kun kaikki haluttu on kirjoitettu tiedostoon, on tiedosto syytä sulkea.

  `tiedostomuuttuja.close()`
- Tiedostoa ei ole syytä yrittää lukea ennen close-käskyn suorittamista, sillä silloin kirjoitettu tieto ei ole välttämättä vielä itse tiedostossa vaan puskurissa odottamassa kirjoittamista.
- Tiedostoon kirjoittaessa voi aiheutua erilaisista virhetilanteista `IOError`-tyyppinen poikkeus, joka on syytä käsitellä try–except-rakenteella.

Esimerkkejä tiedostoon kirjoittamisesta

- Seuraavilla kalvoilla on kaksi esimerkkiohjelmaa tiedostoon kirjoittamisesta.
- Ensimmäinen kirjoittaa käyttäjän antamat nimet tiedostoon.
- Toinen pyytää käyttäjiltä ympyröiden säteitä. Se kirjoittaa tiedostoon kullekin riville yhden säteen ja sitä vastaavan pinta-alan.

Esimerkki: nimiä tiedostoon

```
def main():
    print "Ohjelma kirjoittaa vieraslistan tiedostoon."
    nimi = raw_input("Anna kirjoitettavan tiedoston nimi: ")
    try:
        tulostiedosto = open(nimi, "w")
        print "Anna tallennettavat nimet."
        print "Lopeta tyhjalla rivilla."
        rivi = raw_input()
        while rivi != "":
            tulostiedosto.write(rivi + "\n")
            rivi = raw_input()
        tulostiedosto.close()
        print "Nimet on kirjoitettu tiedostoon", nimi
    except IOError:
        print "Virhe tiedoston", nimi, "kirjoittamisessa."

main()
```

Esimerkki: lukuja tiedostoon

```
import math

def main():
    print "Ohjelma laskee ympyroiden pinta-aloja ja"
    print "tallentaa ne tiedostoon."
    nimi = raw_input("Anna kirjoitettavan tiedoston nimi: ")
    try:
        tulostiedosto = open(nimi, "w")
        tulostiedosto.write("sade    pinta-ala\n")
        print "Anna sateet, lopeta negatiivisella."
        rivi = raw_input()
        sade = float(rivi)
        while sade >= 0:
            pinta_ala = math.pi * sade * sade
            tulostiedosto.write("%-7.2f %-10.2f\n" % \
                (sade, pinta_ala))
```

Esimerkki: lukuja tiedostoon (jatkuu)

```
            rivi = raw_input()
            sade = float(rivi)
        tulostiedosto.close()
        print "Tulokset on kirjoitettu tiedostoon", nimi
    except IOError:
        print "Virhe tiedoston", nimi, "kirjoittamisessa."
    except ValueError:
        print "Ei ollut luku. Tiedoston kirjoitus saattoi"
        print "epaonnistua."


main()
```

Tietojen tallentaminen ohjelman suorituskertojen välillä

- Monissa sovelluksissa ohjelman käyttämät tiedot halutaan tallentaa tiedostoon ohjelman eri suorituskertojen välillä.
- Jos käsiteltävät tietomäärät ovat kohtuullisen kokoisia, menetellään seuraavasti:
  - Ohjelman suorituksen alussa tiedot luetaan tiedostosta ja tallennetaan sopivaan tietorakenteeseen (esim. lista tai sanakirja).
  - Ohjelman suorituksen aikana mahdolliset muutokset tehdään käytettävään tietorakenteeseen, ei suoraan itse tiedostoon.
  - Ohjelman suorituksen päättyessä koko tietorakenne tallennetaan tiedostoon.