Ohjelmoinnin perusteet Y Python

T-106.1208

24.3.2010

Oliomuuttuja toisen olion kenttänä

- Olion kenttänä voi olla viite saman tai toisen luokan olioon.
- Halutaan kirjoittaa luokka `Kellonaytto`, jonka avulla voidaan esittää kellonaikoja muodossa `hh:mm`.
- Luokassa on metodi ajan asettamista varten sekä metodi, joka kasvattaa aikaa minuutilla.
- Luokka pitää huolen siitä, että tunnit ovat aina välillä 0–24 ja minuutit välillä 0–60.
- Määritellään luokka `Numeronaytto`, jonka avulla voidaan esittää lukuja kahdella numerolla. Luokassa on metodi luvun kasvatukseen. Numeronäytön arvo voi olla välillä 0 – (raja - 1), missä raja on määritelty `Numeronaytto`-oliota luodessa.
- `Kellonaytto`-olion kenttinä on kaksi `Numeronaytto`-oliota.
- Esimerkin idea on kirjasta Barnes and Kölling: Objects first with Java.

Numeronaytto, koodi

```
class Numeronaytto:

    def __init__(self, nollausraja):
        self.__arvo = 0
        if 1 <= nollausraja <= 100:
            self.__raja = nollausraja
        else:
            self.__raja = 1

    def kerro_arvo(self):
        return self.__arvo

    def kerro_raja(self):
        return self.__raja
```

Numeronaytto, koodi jatkuu

```
def aseta_arvo(self, uusi_arvo):
    if 0 <= uusi_arvo < self.__raja:
        self.__arvo = uusi_arvo


def kasvata_arvoa(self):
    self.__arvo = (self.__arvo + 1) % self.__raja


def __str__(self):
    if self.__arvo < 10:
        return "0" + str(self.__arvo)
    else:
        return str(self.__arvo)
```

Kellonaytto, koodi

```
import numeronaytto

class Kellonaytto:

    def __init__(self):
        self.__tunnit = numeronaytto.Numeronaytto(24)
        self.__minuutit = numeronaytto.Numeronaytto(60)
        self.aseta_aika(0, 0)


    def aseta_aika(self, uudet_tunnit, uudet_min):
        self.__tunnit.aseta_arvo(uudet_tunnit)
        self.__minuutit.aseta_arvo(uudet_min)
```

Kellonaytto, koodi jatkuu

```
def lisaa_minuutilla(self):
    self.__minuutit.kasvata_arvoa()
    if self.__minuutit.kerro_arvo() == 0:
        self.__tunnit.kasvata_arvoa()


def __str__(self):
    return str(self.__tunnit) + ":" + \
           str(self.__minuutit)
```

Pääohjelma, koodi

```
import kellonaytto

def main():
    kello1 = kellonaytto.Kellonaytto()
    print "Kello aluksi:", kello1
    kello1.aseta_aika(12, 45)
    print "Muutoksen jalkeen:", kello1
    for i in range(128):
        kello1.lisaa_minuutilla()
    print "Lisattiin 128 min:", kello1
```

Pääohjelma, koodi jatkuu

```
    kello2 = kellonaytto.Kellonaytto()
    kello2.aseta_aika(23, 55)
    print "Toinen kello aluksi:", kello2
    for i in range(10):
        kello2.lisaa_minuutilla()
    print "keskiyon jalkeen:", kello2


main()
```

Esimerkki: olion kenttänä olioviitteitä sisältävä lista

- Kirjoitetaan ohjelma oppilasrekisteriä varten.
- Jokaisesta oppilaasta on tallenettu nimi, opiskelijanumero ja tiedot kurssisuorituksista.
- Yhtä kurssisuoritusta kuvataan `Kurssisuoritus`-oliolla. Oliolla on kenttinä suoritetun kurssin koodi, nimi, suorituspäivä, opintopistemäärä ja arvosana.
- `Oppilas`-olion kenttänä on nimen ja opiskelijanumeron lisäksi lista, joka sisältää suoritettuja kursseja vastaavat `Kurssisuoritus`-oliot.
- Lisäksi on kirjoitettu omaan moduuliinsa valikkopohjainen ohjelma, jolla käyttäjä voi luoda uusia opiskelijoita ja lisätä heille kurssisuorituksia.

Luokka Kurssisuoritus

```
class Kurssisuoritus:

    def __init__(self, kurssikoodi, kurssin_nimi, pvm,\
                 pisteet, arvostelu):
        self.__koodi = kurssikoodi
        self.__nimi = kurssin_nimi
        self.__suorituspvm = pvm
        self.__laajuus = pisteet
        self.__arvosana = arvostelu

    def kerro_koodi(self):
        return self.__koodi

    def kerro_nimi(self):
        return self.__nimi
```

Luokka Kurssisuoritus, koodi jatkuu

```
def kerro_suorituspvm(self):
    return self.__suorituspvm

def kerro_laajuus(self):
    return self.__laajuus

def kerro_arvosana(self):
    return self.__arvosana

def __str__(self):
    mjono = "%-10s %-30s %5.1f %2d %10s" %(self.__koodi,\
            self.__nimi, self.__laajuus,\
            self.__arvosana, self.__suorituspvm)
    return mjono
```

Luokka Oppilas

```
class Oppilas:

    def __init__(self, annettu_nimi, nro):
        self.__nimi = annettu_nimi
        self.__opnro = nro
        self.__suoritukset = []


    def kerro_nimi(self):
        return self.__nimi


    def kerro_opnro(self):
        return self.__opnro
```

Luokka Oppilas jatkuu

```
def lisaa_suoritus(self, uusi):
    if uusi.kerro_arvosana() < 1 or\
       uusi.kerro_arvosana() > 5 or\
       uusi.kerro_laajuus() < 0.0:
        return False
    for suoritus in self.__suoritukset:
        if suoritus.kerro_koodi() == uusi.kerro_koodi():
            if uusi.kerro_arvosana() >\
               suoritus.kerro_arvosana():
                self.__suoritukset.remove(suoritus)
                self.__suoritukset.append(uusi)
                return True
            else:
                return False
    self.__suoritukset.append(uusi)
    return True
```

Luokka Oppilas jatkuu

```
def onko_suoritettu(self, kurssikoodi):
    for suoritus in self.__suoritukset:
        if suoritus.kerro_koodi() == kurssikoodi:
            return True
    return False


def laske_opintopistesumma(self):
    summa = 0.0
    for suoritus in self.__suoritukset:
        summa += suoritus.kerro_laajuus()
    return summa
```

Luokka Oppilas jatkuu

```
def laske_keskiarvo(self):
    arvosanasumma = 0.0
    opintopistesumma = 0.0
    for suoritus in self.__suoritukset:
        arvosanasumma += suoritus.kerro_laajuus() *\
                         suoritus.kerro_arvosana()
        opintopistesumma += suoritus.kerro_laajuus()
    if opintopistesumma == 0.0:
        return 0.0
    else:
        return arvosanasumma / opintopistesumma
```

Luokka Oppilas jatkuu

```
def tee_raportti(self):
    raportti = self.__opnro + "  " + self.__nimi + "\n"
    raportti += "Suoritetut kurssit:\n"
    for suoritus in self.__suoritukset:
        raportti += str(suoritus) + "\n"
    opsumma = self.laske_opintopistesumma()
    keskiarvo = self.laske_keskiarvo()
    raportti += str(opsumma) +\
                " op, keskiarvo %.2f." % (keskiarvo)
    return raportti


def __str__(self):
    return self.__opnro + "  " + self.__nimi
```

Käyttöliittymämoduuli

```
import oppilas
import kurssisuoritus

def lue_kokonaisluku():
    luku_onnistui = False
    while not luku_onnistui:
        try:
            syote = raw_input()
            luku = int(syote)
            luku_onnistui = True
        except ValueError:
            print "Virheellinen kokonaisluku!"
            print "Anna uusi!"
    return luku
```

Käyttöliittymämoduuli jatkuu

```
def lue_desimaaliluku():
    luku_onnistui = False
    while not luku_onnistui:
        try:
            syote = raw_input()
            luku = float(syote)
            luku_onnistui = True
        except ValueError:
            print "Virheellinen desimaaliluku!"
            print "Anna uusi!"
    return luku
```

Käyttöliittymämoduuli jatkuu

```
def lisaa_oppilas(oppilaslista):
    print "Anna uuden oppilaan nimi: "
    uusi_nimi = raw_input()
    print "Anna uuden oppilaan opiskelijanumero: "
    uusi_nro = raw_input()
    for jasen in oppilaslista:
        if jasen.kerro_opnro() == uusi_nro:
            print "Opiskelija on jo listassa, ei lisatty."
            return
    oppilaslista.append(oppilas.Oppilas(uusi_nimi, uusi_nro))
```

Käyttöliittymämoduuli jatkuu

```
def lisaa_uusi_suoritus(oppilaslista):
    print "Kenelle suoritus lisataan:"
    nro = kysy_oppilas(oppilaslista)
    if nro < 0:
        print "Kelvoton oppilaan numero"
    else:
        print "Anna kurssikoodi."
        uusi_koodi = raw_input()
        print "Anna kurssin nimi."
        uusi_nimi = raw_input()
        print "Anna suorituspaiva."
        paiva = raw_input()
        print "Anna kurssin opintopistemaara."
        pistemaara = lue_desimaaliluku()
        print "Anna kurssin arvosana."
        numero = lue_kokonaisluku()
```

Käyttöliittymämoduuli jatkuu

```
tehty_suoritus =
      kurssisuoritus.Kurssisuoritus(uusi_koodi,\
      uusi_nimi, paiva, pistemaara, numero)
if oppilaslista[nro].lisaa_suoritus(tehty_suoritus):
    print "Suoritus lisattiin."
else:
    print "Suorituksen lisays ei onnistunut."
```

Käyttöliittymämoduuli jatkuu

```
def tarkista_suoritus(oppilaslista):
    print "Kenen suoritus tarkistetaan:"
    nro = kysy_oppilas(oppilaslista)
    if nro < 0:
        print "Kelvoton oppilaan numero"
    else:
        print "Anna tarkistettavan kurssin koodi."
        annettu_koodi = raw_input()
        if oppilaslista[nro].onko_suoritettu(annettu_koodi):
            print "Oppilas on suorittanut kurssin."
        else:
            print "Oppilas ei ole suorittanut kurssia."
```

Käyttöliittymämoduuli jatkuu

```
def tulosta_oppilaan_raportti(oppilaslista):
    print "Kenen raportti tulostetaan:"
    nro = kysy_oppilas(oppilaslista)
    if nro < 0:
        print "Kelvoton oppilaan numero"
    else:
        print oppilaslista[nro].tee_raportti()
```

Käyttöliittymämoduuli jatkuu

```
def kysy_oppilas(oppilaslista):
    i = 0
    while i < len(oppilaslista):
        print "%d. %s" % (i + 1, oppilaslista[i])
        i += 1
    oppilaan_nro = lue_kokonaisluku()
    if oppilaan_nro < 1 or oppilaan_nro > len(oppilaslista):
        return -1
    else:
        return oppilaan_nro - 1
```

Käyttöliittymämoduuli jatkuu

```
def valikko():
    print "Valitse toiminto:"
    print "1. lisaa uusi oppilas"
    print "2. lisaa kurssisuoritus"
    print "3. tarkista kurssin suoritus"
    print "4. tulosta oppilaan raportti"
    print "5. lopeta"
    valinta = lue_kokonaisluku()
    return valinta
```

Käyttöliittymämoduuli jatkuu

```
def main():
    oppilaat = []
    toiminto = valikko()
    while toiminto != 5:
        if toiminto == 1:
            lisaa_oppilas(oppilaat)
        elif toiminto == 2:
            lisaa_uusi_suoritus(oppilaat)
        elif toiminto == 3:
            tarkista_suoritus(oppilaat)
        elif toiminto == 4:
            tulosta_oppilaan_raportti(oppilaat)
        toiminto = valikko()
    print "Ohjelman suoritus paattyi."
```

Huomautuksia

- Olio-ohjelmointia puhtaasti käyttävässä ohjelmassa myös opiskelijoita sisältävästä listasta olisi tehty oma luokkansa, jonka metodien avulla voitaisiin lisätä opiskelijoita ja opiskelijoille kurssisuorituksia.
- Kurssisuoritusten hakeminen olisi tehostunut, jos `Opiskelija`-olion kurssisuoritukset olisi kerätty listan sijaan sanakirjaan, jossa avaimena on kurssikoodi ja avaimeen liittyvänä arvona koko `Kurssisuoritus`-olio.

Olioiden tietojen lukeminen tekstitiedostosta

- Luotavien olioiden tiedot voidaan lukea tekstitiedostosta samaan tapaan kuin ne luettaisiin suoraan käyttäjältä.
- On muistettava käsitellä erilaiset virhetilanteet (tiedostoa ei pystytä lukemaan, jokin rivi ei ole oletetussa muodossa jne.)
- Seuraavassa esimerkkiohjelmassa luetaan viime luennolla esitetyn `Opiskelija`-luokan olioiden tietoja tekstitiedostosta. Tiedoston rivillä on annettu opiskelijan nimi, opiskelijanumero, tenttiarvosana ja harjoitusarvosana toisistaan kauttaviivalla erotettuna.
- Ohjelma luo tietojen perusteella `Opiskelija`-oliot, lisää heidät listaan ja tekee lopuksi listan opiskelijoista tuloslistan.
- Virheellisistä riveistä aiheutuneet virheet on käsitelty opiskelijoiden tiedot lukevan funktion sisällä. Ohjelma jatkaa toimintaansa normaalisti virheellisen rivin jälkeen.
- Tiedoston lukemisessa tapahtunut virhe käsitellään pääohjelmassa. Tällainen virhe päättää ohjelman suorituksen. Virheen voisi käsitellä myös tiedot lukevan funktion sisällä.

Opiskelijat tiedostosta, koodi

```
import opiskelija

def lue_opiskelijatiedot():
    opiskelijat = []
    print "Mista tiedostosta opiskelijoiden tiedot luetaan?"
    tiedoston_nimi = raw_input()
    tiedosto = open(tiedoston_nimi, "r")
    for rivi in tiedosto:
        rivi = rivi.rstrip()
        tiedot = rivi.split("/")
        if len(tiedot) != 4:
            print "Virheellinen rivi:", rivi
        else:
            uusi = opiskelija.Opiskelija(tiedot[0], tiedot[1])
```

Opiskelijat tiedostosta, koodi jatkuu

```
            try:
                tenttias = int(tiedot[2])
                harj_as = int(tiedot[3])
                uusi.muuta_tenttiarvosana(tenttias)
                uusi.muuta_harjoitusarvosana(harj_as)
                opiskelijat.append(uusi)
            except ValueError:
                print "Rivilla virheellinen arvosana:", rivi
    return opiskelijat
```

Opiskelijat tiedostosta, koodi jatkuu

```
def tulosta_tulokset(opiskelijat):
    print "numero nimi                tentti harj   kurssi"
    for i in range(len(opiskelijat)):
        print "%-6s %-15s %-6d %-6d %-6d" % \
              (opiskelijat[i].kerro_opiskelijanumero(), \
               opiskelijat[i].kerro_nimi(),\
               opiskelijat[i].kerro_tenttiarvosana(),\
               opiskelijat[i].kerro_harjoitusarvosana(),\
               opiskelijat[i].laske_kokonaisarvosana())
```

Opiskelijat tiedostosta, koodi jatkuu

```
def main():
    try:
        opiskelijatiedot = lue_opiskelijatiedot()
        tulosta_tulokset(opiskelijatiedot)
    except IOError:
        print "Virhe tiedoston lukemisessa."


main()
```