T-106.1208 Ohjelmoinnin perusteet Y (Python). Tentti 13.8.2012

Kirjoita jokaisen vastauspaperisi alkuun kurssin nimi, kokeen päivämäärä, nimesi, opiskelijanumerosi, vastauspaperiesi kokonaismäärä sekä allekirjoituksesi.

Tärkeitä ohjeita vastausten kirjoittamiseen: Kun kirjoitat ohjelmakoodia, käytä kahden ruudun levyisiä sisennyksiä ja merkitse sisennykset selvästi. Jos sisennyksiä ei ole käytetty tai ne on merkitty epäselvästi, vähennetään siitä pisteitä. Kirjoitettavaan ohjelmakoodiin ei tarvitse lisätä kommentteja. Missään tehtävässä tulostusta ei tarvitse muotoilla. Voit myös olettaa, että käyttäjän antama syöte on virheetöntä, ellei tehtävässä erikseen käsketä käsittelemään virhetilanteita. **Tentissä ei saa käyttää laskimia eikä lisämateriaalia. Opiskelijat, joiden äidinkieli ei ole suomi, saavat kuitenkin käyttää sanakirjaa, jos siinä ei ole merkintöjä (tentin valvoja tarkistaa sanakirjan).**

1. Kohdissa a, b, c ja d kerro, mitä annettu ohjelma tulostaa. Vastausta ei tarvitse perustella. Kohdissa e, f ja g kerro, mitä tehtävässä esitetty funktio tekee. Älä selitä funktion toimintaa käsky käskyltä, vaan selitä parilla lauseella, mikä on funktion tarkoitus (esimerkiksi: funktio laskee yhteen kaikki parametrina annetussa listassa olevat luvut). Funktioille annettavien parametrien luonne on selitetty kunkin kohdan yhteydessä. Huomaa, että annetuissa ohjelmissa tai funktioissa voi olla myös virheitä. Selitä siinä tapauksessa, miten annettu virheellinen ohjelma tai funktio toimii - ei siis sitä, miten ohjelman tai funktion pitäisi toimia, jos siinä ei olisi virheitä.

a) (2 p)

```
def main():
    pisteet = 1500
    if pisteet  > 1000:
        print "OK pisteet"
    elif pisteet > 1400:
        print "Huippupisteet"
    else:
        print "Huonot pisteet"

main()
```

b) (2 p)

```
def main():
    paiva = "tiistai"
    kello = 6
    if paiva == "lauantai" or paiva == "sunnuntai":
        print "Lippu maksaa 2 euroa."
    else:
        if kello < 7 and kello >= 17:
            print "Lippu maksaa 3 euroa."
        else:
            print "Lippu maksaa 4 euroa."

main()
```

c) (3 p)

```
def main():
    lista = [20, 70, 40, 40, 80, 10]
    tulos = 100
    for luku in lista:
        if luku < 50:
            tulos = tulos - luku
    print tulos

main()
```

d) (5 p)

```
def muuta1(luvut):
    luvut[2] = 28

def muuta2(alkio):
    alkio = 12

def main():
    pisteet = [14, 16, 15]
    muuta1(pisteet)
    raja = 6
    muuta2(raja)
    i = 0
    print raja
    while i < 3:
        print pisteet[i]
        i += 1

main()
```

e) Funktiolle annetaan parametreina kaksi kokonaislukuja sisältävää listaa, joilla on sama pituus (4 p)

```
def mysteeri1(lista1, lista2):
    tulos = 0
    i = 0
    while i < len(lista1):
        if lista1[i] == lista2[i]:
            tulos = tulos + 1
        i += 1
    return tulos
```

f) Funktiolle annetaan ensimmäisenä parametrina kokonaislukuja sisältävä lista, jossa on vähintään kaksi alkiota, ja toisena parametrina kokonaisluku (4 p)

```
def mysteeri3(lista, luku):
    i = 1
    while i < len(lista):
        if lista[i] < lista[i-1] + luku:
            return False
        i = i + 1
    return True
```

g) Funktiolle annetaan parametrina merkkijono. (5 p)

```
def mysteeri3(mjono):
    arvo1 = len(mjono)
    arvo2 = arvo1 / 2
    tulos = mjono[0:arvo2] + "-" + mjono[arvo2:arvo1]
    return tulos
```

2. a) Eräässä yrityksessä on tuntipalkkaisten työntekijöiden kohdalla seuraava käytäntö: Jos viikon työtuntien määrä on korkeintaan 37, saa kaikista tunneista perustuntipalkan. Jos työtuntien määrä on yli 37, mutta korkeintaan 47, maksetaan ensimmäisiltä 37 tunnilta perustuntipalkka ja tämän jälkeen seuraavilta tunneilta 1,5-kertainen tuntipalkka. Jos työtuntien määrä on yli 47, maksetaan ensimmäisiltä 37 tunnilta perustuntipalkka, kymmeneltä seuraavalta 1,5-kertainen tuntipalkka ja lopuilta tunneilta kaksinkertainen tuntipalkka. Kirjoita Python-ohjelma, joka pyytää käyttäjältä yhden työntekijän tuntipalkan sekä viikon aikana tekemien työtuntien määrän ja laskee ja tulostaa työntekijälle kuuluvan kokonaispalkan. (10 p.)

   b) Eräs postimyyntiyritys antaa asiakkailleen bonuspisteitä vuoden lopussa seuraavasti: Jokaisesta yksittäisestä ostoksesta saa yhden bonuspisteen. Jos yksittäisen ostoksen arvo ylittää yrityksen määräämän rajan, saa siitä lisäksi kolme ylimääräistä bonuspistettä. Eri ostosten bonuspisteet lasketaan yhteen. Kirjoita yrityksen tietojärjestelmien käyttöön Python-funktio `laske_bonuspisteet(ostokset, raja)`. Funktio saa ensimmäisenä parametrina listan, joka sisältää asiakkaan vuoden aikana tekemien ostosten arvot desimaalilukuina, ja toisena parametrina yrityksen määräämän rajan lisäbonuksen saamiselle. (Listan pituus vaihtelee sen mukaan, montako ostosta asiakas on tehnyt.) Funktio laskee ja palauttaa asiakkaalle kuuluvien bonuspisteiden yhteismäärän. (20 p)

3. Erään yrityksen varastokirjanpito on tekstitiedostossa, jossa kullakin rivillä on ensin tuotteen tilausnumero, sitten tuotteen nimi, sen jälkeen tuotteen määrä varastossa ja lopuksi tuotteen yksikköhinta. Eri tiedot on erotettu toisistaan kaksoispisteellä. Tiedoston kaksi peräkkäistä riviä voisivat näyttää esim. seuraavalta:

```
51245-0:Hyttysverkko:60:5.5
9439-2:Talvitakki (sininen):67:78.0
```

   Kirjoita Python-ohjelma, joka pyytää käyttäjältä varastokirjanpidon sisältävän tiedoston nimen. Ohjelma lukee tiedoston rivit ja tulostaa kaikkien niiden tuotteiden tilausnumerot ja nimet, joilla sekä yksikköhinta on yli 70.0 euroa että tuotteen määrä varastossa yli 50 kappaletta. Molempien ehtojen pitää siis olla samalle tuotteelle tosia, jotta tuotteen tilausnumero ja nimi tulostettaisiin. Tuotteet tulostetaan samassa järjestyksessä kuin ne ovat tiedostossa. Tuotteiden hinta- ja määrätietoja ei tulosteta.

   Ohjelman on käsiteltävä seuraavat virhetilanteet:
   - Annetun nimistä tiedostoa ei ole olemassa tai tiedoston lukeminen ei onnistu jostain muusta syystä
   - Tiedoston jollain rivillä määrän paikalla olevaa tekstiä ei voi tulkita kokonaisluvuksi tai hinnan paikalla olevaa tekstiä desimaaliluvuksi.

   Näissä tapauksissa ohjelma ilmoittaa käyttäjälle, millainen virhe on sattunut, ja lopettaa toimintansa. Ohjelman ei siis tarvitse jatkaa rivien lukemista virheellisen rivin jälkeen. Voit myös olettaa, että tiedoston jokaisella rivillä on täsmälleen neljä toisistaan kaksoispisteellä erotettua osaa. Ohjelman ei tarvitse osata käsitellä esimerkiksi sellaisia virhetilanteita, joissa rivi on tyhjä tai ei sisällä tilausnumeron ja nimen lisäksi muuta tekstiä. (20 p)

4. Kirjoita Python-kielellä luokka `Luottokortti` yhden luottokortin tietojen kuvaamiseen.

   Luokalla `Luottokortti` on oltava seuraavat kentät:
   - `__nimi` kortin omistajan nimi
   - `__saldo` summa, jonka kortin omistaja on velkaa luottokortin myöntäneelle yhtiölle. Jos esimerkiksi kortin omistaja on tehnyt kortilla 500 euron ostokset eikä ole vielä maksanut luottokorttilaskuaan, niin kentän arvo on 500.0.
   - `__luottoraja` kortin luottoraja eli suurin summa, jonka kortin omistaja voi olla velkaa luottokortin myöntäneelle yhtiölle.
   - `__onko_sulkulistalla` Kentän arvo on `True`, jos kortti on sulkulistalla (korttia ei voi käyttää esimerkiksi sen katoamisen takia) ja `False`, jos korttia voi käyttää normaalisti.

   Määrittele luokkaan seuraavat metodit. (Jos metodin kuvauksessa ei ole kerrottu mitään metodin palauttamasta arvosta, metodin ei tarvitse palauttaa mitään.)
   - `__init__(self, omistajan_nimi, raja)` luo uuden `Luottokortti`-olion. Luotavan kortin omistajan nimi ja luottoraja annetaan metodin parametreina. Uusi luottokortti ei ole sulkulistalla ja sen saldo on 0. Jos parametrina annettu raja on negatiivinen, luottorajaksi asetetaan 500.0.
   - `kerro_nimi(self)` palauttaa luottokortin omistajan nimen.
   - `kerro_luottoraja(self)` palauttaa luottokortin luottorajan.
   - `kerro_saldo(self)` palauttaa kortin saldon.
   - `vie_sulkulistalle(self)` asettaa kortin sopivan kentän arvon niin, että kortti on metodin suorituksen jälkeen sulkulistalla.
   - `tee_lyhennys(self, lyhennys)` pienentää kortin saldoa parametrina annetulla summalla. Metodi ei kuitenkaan tee mitään, jos parametri on negatiivinen. Saldo ei voi myöskään mennä negatiiviseksi, vaan jos parametri on suurempi kuin saldo, uudeksi saldoksi tulee 0.0. Ylimääräinen lyhennys menee siis tavallaan hukkaan.
   - `tee_osto(self, ostoksen_arvo)` lisää kortille tiedon uudesta ostoksesta eli kasvattaa kortin saldoa parametrina annetulla arvolla. Osto onnistuu kuitenkin vain siinä tapauksessa, että kortti ei ole sulkulistalla ja että kortin saldo ei ostoksen jälkeen ylitä kortin luottorajaa. Jos osto ei onnistu, metodi ei muuta kortin tietoja ja palauttaa arvon `False`. Muussa tapauksessa metodi muuttaa kortin saldoa kuvatulla tavalla ja palauttaa arvon `True`.
   - `__str__(self)` palauttaa merkkijonon, joka sisältää kortin omistajan nimen, kortin saldon, luottorajan ja joko tekstin "kortti on käytettävissä" tai "kortti on sulkulistalla" sen mukaan, onko kortti sulkulistalla.

   Kirjoita lisäksi pääohjelma, joka luo kaksi `Luottokortti`-oliota ja kutsuu toiselle niistä `kerro_luottoraja`-metodia. Sitten ohjelman on kutsuttava molemmille korteille kerran `tee_osto`-metodia ja tulostettava, onnistuiko ostosten tekeminen. Sen jälkeen ohjelman on vietävä ensimmäinen kortti sulkulistalle. Lopuksi ohjelman on tulostettava molemmista korteista omistajan nimi, saldo, luottoraja ja tieto siitä, onko kortti käytettävissä vai onko se sulkulistalla. Voit päättää korttien alkutiedot ja ostosten summat itse. Pääohjelman ei siis tarvitse kysyä mitään käyttäjältä. Voit kirjoittaa pääohjelman valintasi mukaan joko niin, että se on samassa moduulissa luokan kanssa tai sitten niin, että se on eri moduulissa. (25 p)