T-106.1208 Ohjelmoinnin perusteet Y (Python). Tentti 12.5.2010

Kirjoita jokaisen vastauspaperisi alkuun kurssin nimi, kokeen päivämäärä, nimesi, opiskelijanumerosi (myös tarkistuskirjain), 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.

Vastaa myös kurssin palautekyselyyn, jos et ole vielä vastannut siihen. Kyselyyn vastaamisesta saa 200 harjoitustehtäväpistettä. Linkki kyselyyn on kurssin Noppa-sivulla.

1. Kohdissa a, b ja c kerro, mitä annettu ohjelma tulostaa. Vastausta ei tarvitse perustella. Kohdissa d, e, f ja g kerro, mitä tehtävässä esitetty funktio tekee. Älä selitä funktion toimintaa käsky käskyltä, vaan selitä parilla lauseella, mikä on funktion tarkoitus. 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 = 80
    if pisteet > 60:
        arvosana = 5
    if pisteet > 30:
        arvosana = 2
    else:
        arvosana = 0
    print "Arvosana on", arvosana

main()
```

b) (2 p)

```
def main():
    lampotila = 22.0
    sademaara = 5.0
    if sademaara < 1.0:
        if lampotila > 15.0:
            print "Kaunista ja lamminta."
        else:
            print "Kaunista, mutta viileaa."
    else:
        print "Sateista."


main()
```

c) (3 p)

```
def main():
    lista = [20, 50, 10, 70, 25, 90]
    i = 0
    while i < len(lista):
        if lista[i] < 30:
            print lista[i]

main()
```

d) Funktiolle annetaan ensimmäisenä parametrina positiivinen kokonaisluku ja toisena parametrina positiivisia kokonaislukuja sisältävä lista. (4 p)

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

e) Funktiolle annetaan parametrina kaksi kokonaislukuja sisältävää listaa, joiden pituus on sama. (4 p)

```
def mysteeri2(luvut1, luvut2):
    tulos = 0
    i = 0
    while i < len(luvut1):
        if luvut1[i] < luvut2[i]:
            tulos += luvut1[i]
        i += 1
    return tulos
```

f) Funktiolle annetaan parametrina kokonaislukuja sisältävä lista ja kokonaisluku. (5 p)

```
def mysteeri3(luvut, numero):
    if luvut[0] != 1:
        return False
    i = 1
    while i < len(luvut):
        if luvut[i] != luvut[i - 1] * numero:
            return False
        i += 1
    return True
```

g) Funktiolle annetaan parametrina merkkijono. (5 p)

```
def mysteeri4(nimi):
    pit = len(nimi)
    if pit >= 4 and nimi[pit-4:pit] == ".doc":
        return nimi[0:pit-4] + ".txt"
    else:
        return nimi
```

2. a) Matkapuhelinoperaattori laskuttaa tekstiviesteistä asiakkaan valinnan mukaan joko kappaleittain tai pakettina. Jos asiakas valitsee kappalelaskutuksen, hän maksaa jokaisesta viestistä x senttiä. Jos asiakas valitsee pakettilaskutuksen, hän maksaa y senttiä kuukaudessa ja saa lähettää kuukaudessa 50 tekstiviestiä ilman lisämaksuja. Jos pakettilaskutuksessa viestejä on kuukauden aikana yli 50, maksaa asiakas jokaisesta ylimenevästä z senttiä. Kirjoita Python-ohjelma, joka kysyy käyttäjältä, kuinka monta teksiviestiä hän aikoo lähettää kuukauden aikana sekä pyytää edellä mainitut hinnat x, y ja z. Ohjelman pitää kertoa käyttäjälle, tuleeko hänelle halvemmaksi valita tekstiviestien laskutus kappaleittain vai pakettina. (10 p.)

   b) Erään tehtaan prosessia valvotaan mm. anturilla, joka mittaa laitteistoon kuuluvassa kattilassa olevan nesteen lämpötilan 5 sekunnin välein ja tallentaa mittaustulokset. Muu laitteisto aiheuttaa kuitenkin välissä häiriöitä, joiden takia osa lämpötilamittauksista epäonnistuu ja tämän seurauksena tuloksiin kirjautuu ajoittain täysin vääriä lämpötiloja. Nämä virheelliset lämpötilat on kuitenkin helppo poimia pois, koska ne ovat useita kymmeniä asteita oikeita lämpötiloja alhaisempia tai korkeampia. Kirjoita prosessia säätelevää tietokoneohjelmaa varten Python-funktio `laske_keskiarvo(lampotilat, alaraja, ylaraja)`, joka saa ensimmaisenä parametrina mitattuja lämpötiloja (desimaalilukuja) sisältävän listan, toisena parametrina kelvollisten lämpötilojen alarajan ja kolmantena parametrina kelvollisten lämpötilojen ylärajan. Funktio laskee ja palauttaa keskiarvon listan `lampotilat` niistä alkioista, jotka ovat vähintään `alaraja`, mutta korkeintaan `ylaraja`. Keskiarvoa laskettaessa ei siis oteta lainkaan huomioon niitä lämpötiloja, jotka eivät ole sallitulla välillä. Jos listassa ei ole yhtään oikealla välillä olevaa lämpötilaa, funktio palauttaa arvon 0.0. (20 p)

3. Yhden CD-levyn kappaleiden nimet ja kestot on tallennettu tiedostoon siten, että kullakin rivillä on aina ensin kappaleen nimi ja sitten kappaleen kesto niin, että ensin on annettu minuutit ja sen jälkeen sekunnit. Eri tiedot on erotettu toisistaan kaksoispisteellä. Yksi tiedoston rivi voisi näyttää esim. seuraavalta:
   `Lumi teki enkelin eteiseen:4:45`
   Tässä siis kappaleen kesto on 4 minuuttia 45 sekuntia.

   Kirjoita Python-ohjelma, joka pyytää käyttäjältä CD-levyn kappaleiden tiedot sisältävän tiedoston nimen. Ohjelma lukee tästä tiedostosta kappaleiden tiedot ja laskee yhteen eri kappaleiden kestot eli laskee koko CD:n soittoajan. Se tulostaa lasketun koko CD:n soittoajan niin, että mahdollisimman suuri osa sekunneista on muutettu minuuteiksi. (Ohjelma siis ilmoittaa CD:n soittoajaksi esimerkiksi 59 min 45 s, eikä esimerkiksi 57 min 165 s.)

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

   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 kolme toisistaan kaksoispisteellä erotettua osaa. Ohjelman ei tarvitse osata käsitellä esimerkiksi sellaisia virhetilanteita, joissa rivi on tyhjä tai ei sisällä nimen lisäksi muuta tekstiä. (20 p)

4. Kirjoita Python-kielellä luokka `Matkakortti` yhden pääkaupunkiseudun matkakortin kaltaisen, mutta yksinkertaisemman matkakortin tietojen kuvaamiseen. Tehtävän yksinkertaistamiseksi oletetaan, että matkakortille voi ladata vain rahaa (ei aikaa) ja että matkustaja ostaa lipun jokaisen matkan alkaessa (vaihto-oikeuksista ei tarvitse välittää). Voit myös olettaa, että kaikilla matkakorteilla liput ovat samanhintaisia ja vyöhykkeitä on vain kaksi. (Lasten matkakortteja ja erilaisia alennuskortteja ei siis tunneta.)

   `Matkakortti`-oliolla on oltava seuraavat kentät:
   - `__omistaja` kortin omistajan nimi
   - `__saldo` kortille ladatun rahan määrä (määrä vähenee, kun kortilla tehdään matkoja)
   - `__onko_validi` kentän arvo on `True`, jos korttia voidaan käyttää (sillä voidaan ostaa matkoja) ja `False`, jos korttia ei voida käyttää esimerkiksi siksi, että omistaja on ilmoittanut sen varastetuksi.

   Määrittele luokkaan seuraavat metodit. (Jos metodin kuvauksessa ei ole kerrottu mitään metodin palauttamasta arvosta, metodin ei tarvitse palauttaa mitään.)
   - `__init__(self, nimi, alkusaldo)` luo uuden `Matkakortti`-olion. Luotavan kortin omistajan nimi ja kortille aluksi ladattava saldo annetaan parametreina. Jos viimeinen parametri on negatiivinen, alkusaldoksi asetetaan 0.0. Uusi kortti on validi.
   - `kerro_omistaja(self)` palauttaa matkakortin omistajan nimen.
   - `kerro_saldo(self)` palauttaa matkakortilla olevan rahan määrän.
   - `lataa_rahaa(self, summa)` lataa kortille rahaa parametrina annetun määrän. Ladattava määrä siis lisätään kortin aikaisempaan saldoon. Lataus onnistuu vain siinä tapauksessa, että kortti on validi ja ladattava summa on positiivinen. Muussa tapauksessa metodi ei tee mitään.
   - `nollaa_saldo(self)` asettaa kortin saldon nollaksi. Tätä voidaan käyttää esimerkiksi silloin, kun kortti on kadonnut.
   - `muuta_ei_validiksi(self)` muuttaa sopivan kentän arvoa niin, että kortti ei ole enää validi.
   - `osta_lippu(self, onko_seutu)` ostaa kortilla yhden lipun. Parametri kertoo, ostetaanko seutulippu vai sisäinen lippu. Jos parametri on `False`, ostetaan sisäinen lippu ja kortin saldoa pienennetään 2 eurolla. Jos parametrin arvo on `True`, ostetaan seutulippu ja kortin saldoa pienennetään 4 eurolla. Lipun osto ei onnistu, jos kortti ei ole validi tai kortin saldo ei riitä halutun lipun ostamiseen. Siinä tapauksessa kortin saldo ei muutu. Metodi palauttaa arvon `True`, jos lipun osto onnistui ja muuten arvon `False`.
   - `__str__(self)` palauttaa merkkijonon, joka sisältää kortin omistajan nimen, kortin saldon, ja joko tekstin "kortti on validi" tai "kortti ei ole validi" sen mukaan, onko kortti validi vai ei.

   Kirjoita lisäksi pääohjelma, joka luo kaksi `Matkakortti`-oliota ja sen jälkeen lataa toiselle niistä lisää rahaa. Tämän jälkeen ohjelman pitää ostaa jommalla kummalla matkakortilla seutulippu ja tulostaa, onnistuiko lipun ostaminen. Sitten ohjelman pitää kutsua samalle kortille `nollaa_saldo`-metodia. Lopuksi ohjelman on tulostettava molemmista korteista omistajan nimi, saldo ja tieto siitä, onko kortti validi. Voit päättää korttien alkutiedot ja ladattavan rahasumman 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)