CSE-A1111 Ohjelmoinnin peruskurssi Y1. Tentti 30.11.2013

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).**

Vastaa myös kurssin palautekyselyyn. Kyselyyn vastaamisesta saa 200 harjoitustehtävä-pistettä. Linkki kyselyyn on kurssin Noppa-sivulla.

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 ja palauttaa parametrina annetussa listassa olevien lukujen summan"). Funktioille annettavien parametrien luonne on selitetty kunkin kohdan yhteydessä. Huomaa, että annetuissa ohjelmissa tai funktioissa voi olla myös virheitä. Kerro siinä tapauksessa, mitä annettu virheellinen ohjelma tulostaa tai miten virheellinen funktio toimii - ei siis sitä, miten ohjelman tai funktion pitäisi toimia, jos siinä ei olisi virheitä.

a) (2 p)

```
def main():
    bakteerit = 90
    if bakteerit > 60:
        print("Veden laatu on huono.")
    if bakteerit < 20:
        print("Veden laatu on erinomainen.")
    else:
        print("Veden laatu on hyva.")

main()
```

b) (2 p)

```
def main():
    paiva = "lauantai"
    ika = 15
    if ika < 18 and ika >= 65:
        print("Lippu maksaa 10 euroa.")
    else:
        if paiva == "lauantai" or paiva == "sunnuntai":
            print("Lippu maksaa 20 euroa.")
        else:
            print("Lippu maksaa 15 euroa.")

main()
```

c) (3 p)

```
def main():
    lista = [10, 20, 30, 10, 40]
    tulos = 100
    for luku in lista:
        if luku < 30:
            tulos = tulos + luku
            print(tulos)

main()
```

d) (5 p)

```
def muuta1(luku):
    luku = 2 * luku

def muuta2(lista):
    lista[2] = lista[0] + lista[1]

def main():
    muuttuja = 17
    muuta1(muuttuja)
    luvut = [20, 30, 10]
    muuta2(luvut)
    i = 0
    print(muuttuja)
    while i < 3:
        print(luvut[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
    return tulos
```

f) Funktiolle annetaan ensimmäisenä parametrina merkkijono ja toisena parametrina positiivinen kokonaisluku. (4 p)

```
def mysteeri2(merkkijono, luku):
    if luku > len(merkkijono):
        return merkkijono + "!" * (luku - len(merkkijono))
    else:
        return merkkijono[0:luku]
```

g) Funktiolle annetaan ensimmäisenä parametrina kokonaislukuja sisältävä lista, jossa on vähintään kaksi alkiota, ja toisena parametrina positiivinen kokonaisluku (5 p)

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

2. a) Eräällä kuntosalilla asiakkaat voivat maksaa käyntinsä joko kertamaksulla tai kuukausimaksulla. Jos asiakas on valinnut kertamaksun, hänen pitää maksaa se jokaisesta käynnistä. Jos taas asiakas on maksanut kuukausimaksun, saa hän käydä maksun voimassaoloaikana salilla niin usein kuin hän haluaa ilman ylimääräisiä maksuja. Kuukausimaksun valinneen asiakkaan pitää kuitenkin sitoutua kuukausimaksuun seuraavaksi 12 kuukaudeksi. Kertamaksulla käyvät asiakkaat voivat halutessaan maksaa kuntosalin jäsenmaksun, joka on 60 eur vuodessa (sitä ei siis peritä kuukausimaksun maksaneilta asiakkailta). Jäsenmaksun maksaneet asiakkaat saavat kertamaksusta 20 %:n alennuksen. Kirjoita Python-ohjelma, joka kysyy käyttäjältä, montako kertaa hän aikoo käydä kuntosalilla seuraavan 12 kuukauden aikana. Ohjelman pitää myös kysyä kertamaksun ja kuukausimaksun hinnat. Tämän jälkeen ohjelma tutkii ja tulostaa, onko käyttäjän edullisinta käydä kuntosalilla maksamalla kertamaksu ilman jäsenkorttia, maksamalla kertamaksu jäsenkortin kanssa vai maksamalla kuukausimaksu. (10 p.)

   b) Eräs yritys myy nettikaupassa musiikkitiedostoja. Yhden musiikkitiedoston hinta on 1 eur. Jos kuitenkin asiakas ostaa kerralla vähintään yrityksen määräämän raja1:n (mutta alle raja2:n) verran tiedostoja, saa hän hinnasta 10 %:n alennuksen. Jos asiakkaan kerralla ostamien tiedostojen määrä on yli yrityksen määräämän raja2:n, saa asiakas hinnasta 30 %:n alennuksen. Kirjoita Python-funktio `laske_paivamyynti(kappalemaarat, raja1, raja2)`. Funktion ensimmäinen parametri on lista, joka sisältää eri asiakkaiden eräänä päivänä kerralla ostamien musiikkitiedostojen määrät. Jos esimerkiksi ensimmäinen asiakas on ostanut tarkasteltavana päivänä kerralla 35 tiedostoa, on listan ensimmäinen alkio 35. Funktion toisena ja kolmantena parametrina on edellä kuvatut rajat alennuksille. Funktion pitää laskea, paljonko yritys on yhteensä saanut tarkasteltavana päivänä rahaa musiikkitiedostojen myynnistä. Funktion pitää palauttaa tämä laskettu arvo. Voit olettaa, että funktiolle on annettu järkevät parametrit. Tehtävässä ei tarvitse kirjoittaa muuta kuin pyydetty funktio. (20 p)

3. Kauppa on tallentanut tiedot tilaamistaan ja myymistään tuotteista tekstitiedostoon seuraavasti: Tiedoston rivillä on ensin tuotteen tilausnumero, sitten tuotteen nimi ja tämän jälkeen kolme kokonaislukua. Ensimmäinen kertoo, kuinka monta kappaletta tuotetta on tilattu kauppaan myytäväksi. Toinen luku kertoo, kuinka monta kappaletta tuotetta on myyty alennuksella (koska se ei mennyt kaupaksi normaalihinnalla). Kolmas luku kertoo, kuinka monta kappaletta tuotetta jäi kokonaan myymättä. Eri tiedot on erotettu toisistaan puolipisteellä. Tiedoston rivit voisivat näyttää esim. seuraavilta:

```
T-12345;Bosch vatkain;100;22;5
T-22556;Pilkkihaalari koko 48;50;30;18
```

   Voit olettaa, että yhden tuotteen tietoja on tiedostossa vain yhteen kertaan. Kirjoita Python-ohjelma, joka pyytää käyttäjältä tiedot sisältävän tiedoston nimen. Ohjelma lukee tiedot tästä tiedostosta ja tulostaa kaikkien niiden tuotteiden nimet, joilla alennuksella myytyjen ja kokonaan myymättä jääneiden kappaleiden summa on yli 50 % tilatusta kappalemäärästä.

   Ohjelman on käsiteltävä seuraavat virhetilanteet:
   - Annetun nimistä tiedostoa ei ole olemassa tai tiedoston lukeminen ei onnistu jostain muusta syystä
   - Tiedoston jollain rivillä tilattujen, alennuksella myytyjen tai myymättä jääneiden kappalemäärän 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 viisi toisistaan puolipisteellä erotettua osaa. Ohjelman ei tarvitse osata käsitellä esimerkiksi sellaisia virhetilanteita, joissa rivi on tyhjä tai ei sisällä tuotenumeron ja tuotteen nimen lisäksi muuta tekstiä. Voit myös olettaa, että tiedostossa annetut kappalemäärät ovat järkeviä, kunhan ne ovat lukuja. (20 p)

VIIMEINEN TEHTÄVÄ SEURAAVALLA SIVULLA

4. Kirjoita luokka `Vakuutus` vakuutusyhtiön asiakkaan yhden vakuutuksen kuvaamista varten. Yhtiössä asiakas saa vakuutukseen bonuksia, jos vakuutukselle ei tule tarpeeksi pitkään aikaan korvattavia vahinkoja. Uuden vakuutuksen bonus on 20 % eli asiakkaan maksama vakuutusmaksu on 20 % pienempi kuin se olisi ilman bonuksia. Joka kerta, kun asiakas maksaa vakuutusmaksun, hänen vakuutuksensa bonus kasvaa 10 prosenttiyksikköä. (Jos esim. uudelle asiakkaalle ei ole sattunut yhtään vahinkoa, nousee vakuutuksen bonus ensimmäisen vakuutusmaksun maksamisen jälkeen 30 %:iin.) Jokainen korvattava vahinko taas pienentää vakuutuksen bonusta 20 prosenttiyksiköllä. Jos kuitenkin maksettu korvaus (omavastuun vähentämisen jälkeen) on yli 5000 euroa, laskee bonus nollaan. Jos bonus kerrottujen sääntöjen mukaan menisi negatiiviseksi, tulee bonukseksi 0. Bonus ei voi myöskään kasvaa yli 60 %:n.

   `Vakuutus`-oliolla on oltava seuraavat kentät:
   - `__omistaja` vakuutuksen omistajan nimi
   - `__perusmaksu` vakuutusmaksu ilman bonuksia
   - `__bonus` vakuutuksen bonus prosentteina
   - `__onko_voimassa` kentän arvo on `True`, jos vakuutus on voimassa ja muuten `False` (yhtiö voi irtisanoa vakuutuksen esimerkiksi maksamattomien vakuutusmaksujen vuoksi).

   Määrittele luokkaan seuraavat metodit. (Jos metodin kuvauksessa ei ole kerrottu mitään metodin palauttamasta arvosta, metodin ei tarvitse palauttaa mitään. Tehtävän yksinkertaistamiseksi luokasta puuttuu metodeita, joita siinä käytännössä kannattaisi olla.)
   - `__init__(self, nimi, maksu)` luo uuden `Vakuutus`-olion. Vakuutuksen omistajan nimi ja vakuutusmaksu ilman bonuksia annetaan parametreina. Uuden vakuutuksen bonus on 20 % ja vakuutus on voimassa.
   - `kerro_bonus(self)` palauttaa vakuutuksen bonuksen.
   - `kerro_onko_voimassa(self)` palauttaa `True`, jos vakuutus on voimassa ja muuten `False`.
   - `muuta_perusmaksu(self, uusi_maksu)` muuttaa vakuutuksen uudeksi perusmaksuksi parametrina annetun arvon, jos parametri on positiivinen. Jos parametri on negatiivinen tai nolla, metodi ei tee mitään.
   - `saata_voimaan(self)` muuttaa aikaisemmin irtisanotun vakuutuksen olevan jälleen voimassa.
   - `irtisano(self)` irtisanoo vakuutuksen eli muuttaa sen tiedot niin, että se ei ole voimassa.
   - `laske_vakuutusmaksu(self)` laskee ja palauttaa todellisen vakuutusmaksun, kun bonus on otettu huomioon
   - `maksa_vakuutusmaksu(self, summa)` maksaa vakuutusmaksun parametrina annetulla summalla. Käytännössä metodi tarkistaa, että parametrina annettu summa riittää vakuutusmaksun maksamiseen ja kasvattaa vakuutuksen bonuksia, jos summa on riittävä. Jos maksaminen onnistuu (summa on tarpeeksi suuri), metodi palauttaa arvon `True`. Jos maksaminen ei onnistu, metodi ei muuta bonuksia ja palauttaa arvon `False`.
   - `maksa_korvaus(self, vahinkosumma)` maksaa korvauksen vakuutuksen perusteella. Korvattavan vahingon suuruus (euroina) annetaan metodin parametrina. Korvauksen maksaminen edellyttää sitä, että vakuutus on voimassa. Vahinkosummasta vähennetään 80 euron omavastuu ennen korvauksen maksamista. Metodi pienentää vakuutuksen bonuksia aikaisemmin kerrottujen sääntöjen mukaisesti ja palauttaa arvonaan maksettavan korvaussumman. Jos vahingon suuruus on alle omavastuun tai vakuutus ei ole voimassa, metodi palauttaa arvon 0 eikä muuta bonuksia.
   - `__str__(self)` palauttaa merkkijonon, joka sisältää vakuutuksen omistajan nimen, perusmaksun, bonuksen ja joko tekstin "vakuutus on voimassa" tai "vakuutus ei ole voimassa" sen mukaan, onko vakuutus voimassa.

   Kirjoita lisäksi pääohjelma, joka luo kaksi `Vakuutus`-oliota ja sen jälkeen muuttaa niistä toisen perusmaksua. Sitten pääohjelma maksaa ensiksi luodun vakuutuksen maksun ja tulostaa, onnistuiko se. Tämän jälkeen ohjelman pitää laskea ja tulostaa ensiksi luodun vakuutuksen vakuutusmaksu. Sitten ohjelman pitää kutsua metodia `maksa_korvaus` ensiksi luodulle vakuutukselle ja tulostaa maksettu korvaussumma. Lopuksi ohjelman pitää tulostaa molempien vakuutusten tiedot (omistajan nimi, vakuuksen perusmaksu, bonus ja tieto siitä, onko vakuutus voimassa). Voit päättää vakuutusten ja vahinkojen tiedot itse. Pääohjelman ei siis tarvitse kysyä mitään käyttäjältä. (25 p)