T-106.1208 Ohjelmoinnin perusteet Y (Python). Tentti 19.5.2011

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. (Palautekyselyjärjestelmä voi olla pois toiminnasta 19.5. illalla, mutta avautuu taas 20.5.)

1. Kohdissa a, b, c ja d kerro, mitä annettu ohjelma tulostaa. Vastausta ei tarvitse perustella. Kohdissa e ja f 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():
    edustajalkm = 39
    if edustajalkm > 25:
        print "Meni jotenkin."
    elif edustajalkm > 35:
        print "Iso jytky!"
    else:
        print "Huonosti meni!"

main()
```

b) (2 p)

```
def main():
    paiva = "maanantai"
    ika = 14
    if paiva == "sunnuntai":
        print "Lippu maksaa 10 euroa."
    else:
        if ika > 15 or ika <= 65:
            print "Lippu maksaa 25 euroa."
        else:
            print "Lippu maksaa 15 euroa."

main()
```

c) (3 p)

```
def main():
    lista = [20, -50, 5, 40, -10, 0]
    tulos = 150
    for luku in lista:
        if luku > 0:
            tulos = tulos - luku
    print tulos

main()
```

d) (5 p)

```
def muuta1(lista):
    lista[1] = lista[1] + 5

def muuta2(luku):
    luku = luku + 2

def main():
    luvut = [4, 8, 12]
    muuta1(luvut)
    muuttuja = 15
    muuta2(muuttuja)
    i = 0
    while i < 3:
        print luvut[i]
        i += 1
    print muuttuja

main()
```

e) Funktiolle annetaan ensimmäisenä parametrina desimaalilukuja sisältävä lista ja toisena parametrina desimaaliluku (4 p)

```
def mysteeri1(lista2, luku2):
    tulos = 0.0
    for alkio in lista2:
        if alkio > luku2:
            tulos = tulos + 1.22 * alkio
        else:
            tulos = tulos + alkio
    return tulos
```

f) Funktiolle annetaan parametrina merkkijono. (5 p)

```
def mysteeri2(mjono):
    uusi_jono = "*"
    if len(mjono) > 3:
        for m in mjono:
            uusi_jono += m
            uusi_jono += "*"
        return uusi_jono
    else:
        return mjono
```

g) Seuraavan funktion tarkoituksena on tarkistaa, onko sille parametrina annettujen listojen sisältö sama eli onko niissä samat luvut vastaavilla paikoilla. Voit olettaa, että funktiolle annetaan parametrina kaksi kokonaislukuja sisältävää listaa, joiden pituus on sama. Mikä virhe funktiossa on ja miten se vaikuttaa funktion toimintaan? (Etsitty virhe ei siis ole se, että listojen pituus voi olla eri, jos tehtävänannon oletukset eivät pidä paikkaansa.) (4 p)

```
def onko_samat(lista1, lista2):
    i = 0
    while i < len(lista1):
        if lista1[i] != lista2[i]:
            return False
        else:
            return True
        i += 1
```

2. a) Olet hankkimassa mobiililaajakaistaa kannettavaan tietokoneeseesi. Tarvitset myös nettitikun. Eräs operaattori antaa normaalikuukausihinnasta 5 %:n alennuksen, jos teet vähintään 12 kuukauden sopimuksen. Tämän alennuksen lisäksi operaattori antaa nettitikun ilmaiseksi, jos teet vähintään 24 kuukauden sopimuksen. Muussa tapauksessa joudut ostamaan nettitikun itse. Mietit, miten pitkä sopimus sinun kannattaisi tehdä. Kirjoita valinnan tueksi Python-ohjelma, joka pyytää käyttäjältä normaalikuukausihinnan, nettitikun hinnan ja sopimusajan kuukausina. Ohjelma laskee ja tulostaa mobiililaajakaistan hinnan kuukautta kohti, kun sekä kuukausimaksu (mahdollisine alennuksineen) että nettitikun hinta (ostohinta jaettuna sopimuskuukausien määrällä) otetaan huomioon. Jos operaattori antaa nettitikun ilmaiseksi, ei nettitikun hintaa lasketa kustannuksiin. (10 p.)

b) Erään yrityksen puhelinmyyjillä palkka lasketaan päivittäin sen mukaan, kuinka monta kappaletta myyjä on päivän aikana myynnyt yrityksen tuotetta. Jos myyjä on myynyt yrityksen tuotetta päivän aikana alle 20 kappaletta, hän saa jokaista myytyä kappaletta kohti määrätyn peruskorvauksen. Jos päivän aikana myyty määrä on 20-40 kappaletta, jokaisesta myydystä kappaleesta saa 1,1 kertaa peruskorvauksen. Jos päivän myynti on yli 40 kappaletta, jokaisesta myydystä kappaleesta saa 1,3 kertaa peruskorvauksen. Myyjällä on kuitenkin päiväkohtainen takuupalkka. Jos päivän myynnin mukaan laskettu päiväpalkka jäisi alle takuupalkan, myyjä saa tuosta päivästä takuupalkan (mutta ei siitä päivästä enää erikseen myyntimäärään perustuvaa korvausta). Yritys haluaa laskea yhdelle myyjälle pidemmältä ajalta kertyvän kokonaispalkan. Kirjoita tätä varten Python-funktio `laske_kokonaispalkka(myyntitiedot, peruskorvaus, takuupalkka)`. Funktion ensimmäisenä parametrina on lista, joka sisältää eri päivien myyntimäärät (listan yksi alkio on yhden päivän aikana myytyjen kappaleiden määrä) . Listan alkiot ovat kokonaislukuja, eikä listan pituutta ole määrätty etukäteen. Funktion toisena parametrina on yhden kappaleen myynnistä maksettava peruskorvaus euroina (desimaaliluku), ja kolmantena parametrina päiväkohtainen takuupalkka euroina. Funktion on laskettava ja palautettava puhelinmyyjälle listassa olevien päivien perusteella kuuluva kokonaispalkka. (20 p)

3. Erään yrityksen varastokirjanpito on tallennettu tiedostoon siten, että yhdellä rivillä on aina ensin tuotteen nimi, sitten tuotteen määrä varastossa tällä hetkellä, sitten tuotteen hälytysraja ja lopuksi tuotteen tilauserä. Tiedostossa olevaa tuotteen määrää päivitetään aina, kun tuotetta tuodaan varastoon lisää tai viedään varastosta pois. Tiedosto käydään läpi kerran päivässä, ja jos tällöin jonkin tuotteen määrän havaitaan olevan pienempi kuin tuotteen hälytysrajan, tilataan tätä tuotetta lisää tuotteen tilauserän verran. Tiedoston riveillä eri tiedot on erotettu toisistaan puolipisteellä. Kolme tiedoston riviä voisivat näyttää esim. seuraavalta:

```
Hilavitkutin;20;25;50
Hyttysöljy;15;16;32
Karhunpelotin;1;1;2
```

Kirjoita Python-ohjelma, joka pyytää käyttäjältä varastokirjanpidon sisältävän tiedoston nimen. Ohjelma lukee tästä tiedostosta kirjanpidon ja tulostaa tilauslistan eli niiden tuotteiden nimet, joita pitää tilata lisää sekä tilattavat määrät (kunkin tuotteen tilauserä). Esimerkkitapauksessa ohjelman pitäisi tulostaa

```
Hilavitkutin 50
Hyttysöljy 32
```

(Karhunpelotinta ei tilata, koska sen määrä varastossa ei ole hälytysrajan alapuolella).

Ohjelman on käsiteltävä seuraavat virhetilanteet:

- Annetun nimistä tiedostoa ei ole olemassa tai tiedoston lukeminen ei onnistu jostain muusta syystä
- Tiedoston jollain rivillä määrän, hälytysrajan tai tilauserän 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 neljä toisistaan puolipisteellä erotettua osaa. Ohjelman ei tarvitse osata käsitellä esimerkiksi sellaisia virhetilanteita, joissa rivi on tyhjä tai ei sisällä nimen lisäksi muuta tekstiä. (20 p)

4. Ystäväsi aikoo perustaa käytettyjen tavaroiden nettihuutokaupan. Kirjoita Python-kielellä luokka `Huutokohde` yhden myytävän tavaran (kohteen) tietojen käsittelyyn.

`Huutokohde`-oliolla on oltava seuraavat kentät:

- `__nimi` myytävän kohteen nimi
- `__hintavaraus` myyjän asettama vähimmäishinta, jota pienemmällä hinnalla myyjä ei myy kohdetta. Ostajat voivat kuitenkin tehdä tätä pienempiä tarjouksia.
- `__onko_auki` kentän arvo on `True`, jos kohde on tällä hetkellä huudettavana (siitä voidaan tehdä tarjouksia) ja `False`, jos kohteesta ei voi tehdä tarjouksia, koska sen myyntiaika ei ole vielä alkanut tai myyntiaika on jo päättynyt.
- `__korkein_tarjous` kohteesta tähän mennessä tehty korkein tarjous (summa euroina)
- `__tarjoaja` tähän mennessä korkeimman tarjouksen tekijä (nimi)

Huomaa, että kohteen yhteydessä säilytetään tietoa vain korkeimmasta tarjouksesta ja aikaisemmat tarjoukset tavallaan unohdetaan.

JATKUU SEURAAVALLA SIVULLA

Määrittele luokkaan seuraavat metodit. (Halutessasi saat määritellä muitakin metodeita. Jos metodin kuvauksessa ei ole kerrottu mitään metodin palauttamasta arvosta, metodin ei tarvitse palauttaa mitään.)

- `__init__(self, nimi, hintavaraus)` luo uuden `Huutokohde`-olion. Luotavan kohteen nimi ja hintavaraus annetaan parametreina. Jos viimeinen parametri on negatiivinen, hintavaraukseksi asetetaan `0.0`. Uusi kohde ei ole auki, korkein tarjous on `0.0` ja tarjoaja tyhjä merkkijono (`""`).
- `kerro_nimi(self)` palauttaa kohteen nimen.
- `kerro_hintavaraus(self)` palauttaa kohteen hintavarauksen.
- `avaa_kohde(self)` avaa kohteen huudettavaksi.
- `sulje_kohde(self)` sulkee kohteen.
- `tee_tarjous(self, summa, tekija)` Tätä metodia käytetään, kun joku tekee kohteesta uuden tarjouksen. Metodin parametreina annetaan uuden tarjouksen määrä (euroina) ja tarjouksen tekijän nimi. Metodi tarkistaa, että kohde on auki ja että uusi tarjous on vähintään 2 euroa suurempi kuin kohteesta aikaisemmin tehty korkein tarjous. Jos molemmat ehdot pitävät paikkansa, metodi muuttaa kenttien `__korkein_tarjous` ja `__tarjoaja` arvoja ja palauttaa arvon `True`. Muussa tapauksessa metodi ei muuta minkään kentän arvoa ja palauttaa arvon `False`.
- `maaraa_myyntihinta(self)` palauttaa kohteen lopullisen myyntihinnan (korkeimman tarjouksen arvon), jos kohde on jo suljettu ja siitä tehty korkein tarjous on vähintään yhtäsuuri kuin kohteen hintavaraus. Muussa tapauksessa metodi palauttaa arvon `-1.0` kertomaan, että kohteen myynti ei (ainakaan vielä) onnistu.
- `maaraa_ostaja(self)` palauttaa kohteen ostajan (korkeimman tajouksen tekijän nimen), jos kohde on jo suljettu ja siitä tehty korkein tarjous on vähintään yhtäsuuri kuin kohteen hintavaraus. Muussa tapauksessa metodi palauttaa tyhjän merkkijonon (`""`) kertomaan, että kohteen myynti ei (ainakaan vielä) onnistu.
- `__str__(self)` palauttaa merkkijonon, joka sisältää kohteen nimen, hintavarauksen, korkeimman tarjouksen ja sen tekijän nimen ja joko tekstin "kohde on auki" tai "kohde on suljettu" sen mukaan, onko kohde auki vai ei.

Kirjoita lisäksi pääohjelma, joka luo kaksi `Huutokohde`-oliota, avaa ne molemmat ja tekee niistä molemmista kaksi tarjousta. Yhden tarjouksen kohdalla ohjelman pitää tulostaa, onnistuiko tarjous (oliko sen summa tarpeeksi suuri niin, että tarjous hyväksyttiin uudeksi korkeimmaksi tarjoukseksi). Tämän jälkeen ohjelman pitää sulkea toinen kohteista ja tulostaa sen ostajan nimi ja myyntihinta tai tieto siitä, että kohteen myynti ei onnistunut. Lopuksi ohjelman on tulostettava molemmista kohteista nimi, hintavaraus, korkein tarjous ja sen tekijä sekä tieto siitä, onko kohde auki. Voit päättää kohteiden alkutiedot ja tarjousten tiedot 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)