T-106.1208 Ohjelmoinnin perusteet Y (Python). Tentti 12.8.2013

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).**

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():
    tulos = 1500
    if tulos > 1400:
        print("Loistotulos!")
    if tulos < 1000:
        print("Meni kehnosti.")
    else:
        print("OK")

main()
```

b) (2 p)

```
def main():
    paiva = "perjantai"
    kello = 19
    if paiva == "lauantai" or paiva == "sunnuntai":
        print("Lippu maksaa 3 euroa.")
    else:
        if kello >= 16 and kello <= 8:
            print("Lippu maksaa 2 euroa.")
        else:
            print("Lippu maksaa 5 euroa.")

main()
```

c) (3 p)

```
def main():
    lista = [10, 20, 30, 10, 40]
    tulos = 0
    for luku in lista:
        if luku > 20:
            tulos = tulos + luku
            print(tulos)

main()
```

d) (5 p)

```
def muuta1(luku):
    luku = 0

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

def main():
    muuttuja = 18
    muuta1(muuttuja)
    luvut = [5, 10, 30]
    muuta2(luvut)
    i = 0
    print(muuttuja)
    while i < 3:
        print(luvut[i])
        i += 1

main()
```

e) Funktiolle annetaan parametreina kaksi desimaalilukuja sisältävää listaa, joilla on sama pituus. Tämä pituus on vähintään yksi. (4 p)

```
def mysteeri1(lista1, lista2):
    tulos = 0.0
    i = 0
    while i < len(lista1):
        tulos += lista1[i] - lista2[i]
        i += 1
    tulos = tulos / len(lista1)
    return tulos
```

f) Funktiolle annetaan parametrina merkkijono. (4 p)

```
def mysteeri2(mjono):
    uusi_jono = "*"
    i = 1
    while i < len(mjono) - 1:
        uusi_jono = uusi_jono + mjono[i]
        i = i + 1
    uusi_jono += "*"
    return uusi_jono
```

g) Funktiolle annetaan parametrina kokonaislukuja sisaltava lista, jossa on vähintään kaksi alkiota. (5 p)

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

2. a) Eräs liike tarjoaa vähintään 50 euron kertaostoksesta 5 %:n alennuksen, jos asiakkaalla on tämän liikkeen bonuskortti. Jos kertaostoksen arvo on alle 50 euroa tai asiakkaalla ei ole bonuskorttia, maksaa asiakas ostoksesta normaalihinnan. Kirjoita Python-ohjelma, joka kysyy ostoksen normaalihinnan ja sen, onko asiakkaalla bonuskortti. Tämän jälkeen ohjelman pitää tulostaa asiakkaan maksettavaksi tuleva hinta. Tässä ei ole tarkemmin määrätty ohjelman täsmällistä tulostusta ja käyttäjältä luettavien syötteiden muotoa (esimerkiksi sitä, miten bonuskortin olemassaolosta kerrotaan), vaan voit päättää ne itse, kunhan ohjelmasi toimii järkevästi. (10 p.)

   b) Yrityksen erään osaston kaikkien työntekijöiden kuukausipalkat on tallennettu listaan niin, että listan kukin alkio (desimaaliluku) kertoo yhden työntekijän kuukausipalkan. Kirjoita Python-funktio `monellako_suurempi(palkat)`, joka saa parametrina edellä kuvatun listan. Funktio palauttaa arvonaan tiedon siitä, kuinka monella osaston työntekijällä palkka on suurempi kuin osaston kaikkien työntekijöiden palkkojen keskiarvo. (20 p)

3. Tikanheittokilpailussa kukin osanottaja voi heittää karsintakierroksella joko yhden tai kaksi kertaa. Jos kilpailija heittää vain yhden kerran, on tämän heiton tulos hänen karsintatuloksensa. Jos hän heittää kaksi kertaa, on hänen karsintatuloksensa molempien heittokertojen keskiarvo. Karsintakierroksen tulokset on tallennettu teksitiedostoon siten, että yhdellä rivillä on aina jonkin osanottajan nimi ja sen jälkeen tämän osanottajan heittojen tulokset (kokonaislukuja), joita voi siis olla joko yksi tai kaksi kappaletta. Eri tiedot on erotettu toisistaan kaksoispisteellä. Tiedoston rivit voisivat näyttää esim. seuraavilta:

```
Teemu Teekkari:26:30
Tiina Teekkari:45
Katja Kemisti:30:18
```

   Jatkoon pääsevät ne heittäjät, joiden karsintatulos on vähintään 28. Kirjoita Python-ohjelma, joka pyytää käyttäjältä tulokset sisältävän tiedoston nimen. Ohjelma lukee tiedot tästä tiedostosta ja tulostaa kaikkien jatkoon päässeiden heittäjien nimet ja karsintatulokset. Esimerkkitapauksessa ohjelman tulostus olisi siis (tulostuksen muotoilun ei tarvitse olla sama kuin alla)

```
Teemu Teekkari 28.0
Tiina Teekkari 45.0
```

   Ohjelman on käsiteltävä seuraavat virhetilanteet:

   - Annetun nimistä tiedostoa ei ole olemassa tai tiedoston lukeminen ei onnistu jostain muusta syystä
   - Tiedoston jollain rivillä heittotuloksen 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 kaksi tai 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)

TENTTI JATKUU SEURAAVALLA SIVULLA

4. Ystäväsi aikoo perustaa pikavippiyrityksen ja tarvitsee tietokoneohjelman asiakasrekisteriä varten. Kirjoita ohjelmaa varten Python-luokka `Vippaaja` yhden asiakkaan tietojen kuvaamista varten.

`Vippaaja`-oliolla on oltava seuraavat kentät:

- `__nimi` asiakkaan nimi
- `__luottosaldo` asiakkaan velka yritykselle tällä hetkellä (positiivinen, jos asiakas on velkaa)
- `__pisteet` kokonaisluku (voi olla myös negatiivinen), joka kuvaa sitä, kuinka hyvä asiakas on kysymyksessä
- `__luottoraja` asiakkaalle voidaan yleensä antaa lainaa korkeintaan luottorajaan asti.

Määrittele luokkaan seuraavat metodit. (Jos metodin kuvauksessa ei ole kerrottu mitään metodin palauttamasta arvosta, metodin ei tarvitse palauttaa mitään.)

- `__init__(self, nimi1, alkusaldo, raja)` luo uuden `Vippaaja`-olion. Luotavan asiakkaan nimi, alkuperäinen luottosaldo ja luottoraja annetaan parametreina. Uudella asiakkaalla on nolla pistettä. Metodin ei tarvitse tarkistaa annettujen parametrien järkevyyttä.
- `kerro_luottosaldo(self)` palauttaa asiakkaan luottosaldon.
- `kerro_pisteet(self)` palauttaa asiakkaan pisteet.
- `muuta_luottorajaa(self, uusi_raja)` muuttaa asiakkaan luottorajan parametrin mukaiseksi, jos parametri on vähintään 0. Metodi ei tee muita tarkistuksia, joten esim. luottosaldo voi olla metodin suorituksen jälkeen suurempi kuin luottoraja.
- `tee_lyhennys(self, summa)` pienentää asiakkaan luottosaldoa. Asiakkaan maksama summa annetaan parametrina. Summaa ei kuitenkaan käytetä kokonaan lyhennykseen, vaan siitä vähennetään ensin 5 euroa luoton hoitokuluja sekä edellisen lyhennyksen jälkeen kertynyt korko, joka on 10 % luottosaldosta. Tämän jälkeen jäljelle jäänyt summa käytetään varsinaiseen lyhennykseen eli luottosaldoa pienennetään sen verran. Jos varsinainen lyhennys on vähintään 30 euroa, asiakkaan pisteitä kasvatetaan yhdellä. Jos varsinainen lyhennys jää vähennysten jälkeen negatiiviseksi, lisätään sen itseisarvo luottosaldoon (myös siinä tapauksessa, että luottosaldo ylittää tämän jälkeen luottorajan) ja pisteitä vähennetään yhdellä. Luottosaldo saa mennä myös negatiiviseksi.
- `ota_lisaa_luottoa(self, lisays)` kasvattaa asiakkaan luottosaldoa parametrina annetulla määrällä. Kasvatus onnistuu kuitenkin vain siinä tapauksessa, että asiakkaalla on vähintään viisi pistettä ja että luottosaldo on lisäyksen jälkeen korkeintaan luottoraja. Jos se onnistuu, asiakkaan pisteitä vähennetään viidellä ja metodi palauttaa arvon `True`. Jos luoton kasvatus ei onnistu, luottosaldoa ja pisteitä ei muuteta ja metodi palauttaa arvon `False`.
- `__str__(self)` palauttaa merkkijonon, joka sisältää asiakkaan nimen, luottosaldon, luottorajan ja pisteet.

Kirjoita lisäksi joko samaan tai toiseen moduuliin pääohjelma, joka luo kaksi `Vippaaja`-oliota ja sen jälkeen kutsuu toiselle niistä `kerro_luottosaldo`-metodia ja kaksi kertaa `tee_lyhennys`-metodia. Tämän jälkeen ohjelman pitää yrittää kasvattaa asiakkaan luottoa ja tulostaa joko "lisaluoton ottaminen onnistui" tai "lisaluoton ottaminen ei onnistunut" sen mukaan, onnistuiko luoton kasvattaminen. Lopuksi ohjelman on tulostettava molempien asiakkaiden tiedot (nimi, luottosaldo, luottoraja ja pisteet). Voit päättää asiakkaiden ja luottojen tiedot itse. Niitä ei tarvitse kysyä käyttäjältä. (25 p)