CS/CSE-A1111 Ohjelmoinnin peruskurssi Y1 Tentti 24.10.2016

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. Jos sisennyksiä ei ole käytetty tai niistä ei saa selvää, vähennetään siitä pisteitä. Kirjoitettavaan ohjelmakoodiin ei tarvitse lisätä kommentteja. Missään tehtävässä tulostuista 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). Nämä opiskelijat saavat halutessaan myös sekä suomen- että englanninkielisen kysymyspaperin.

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():
    bakteeripitoisuus = 200
    if bakteeripitoisuus > 10:
        print("Veden laatu on kohtuullinen.")
    elif bakteeripitoisuus > 150:
        print("Veden laatu on huono.")
    else:
        print("Veden laatu on hyvä.")

main()
```

b) (2 p)

```
def tuplaa(luku):
    print(2 * luku)

def main():
    arvo = 15
    luku = 4
    tuplaa(arvo)

main()
```

c) (3 p)

```
def main():
    tulos = 100
    lista = [35, 10, 50, 10, 5]
    for luku in lista:
        if tulos > 30:
            tulos = tulos - luku
    print(tulos)

main()
```

d) (5 p)

```
def muuta(luku, luvut):
    luvut[0] = 2 * luvut[1]
    luku = 5 * luku
    return luku

def main():
    lista = [2, 12, 15]
    tulos = 10
    muuta(tulos, lista)
    print(tulos)
    for alkio in lista:
        print(alkio)

main()
```

e) Funktiolle annetaan parametreina kaksi listaa, jotka sisältävät kokonaislukuja ja joissa on sama määrä alkioita. (4 p)

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

f) Funktiolle annetaan parametrina merkkijono, joka sisältää vähintään yhden merkin. (4 p)

```
def mysteeri2(merkkijono1):
    i = len(merkkijono1) - 1
    merkkijono2 = ""
    while i >= 0:
        merkkijono2 += merkkijono1[i]
        i -= 1
    return merkkijono2
```

g) Funktiolle annetaan parametrina lista, joka sisältää kokonaislukuja ja jossa on vähintään 2 alkiota. (5 p)

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

2. a) Eräs liike tarjoaa vähintään 100 euron kertaostoksesta 5 %:n alennuksen, jos asiakkaalla on tämän liikkeen bonuskortti. Jos kertaostoksen arvo on alle 100 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 euroina. Kirjoita Python-funktio `monellako_suurempi(palkat, vali)`, joka saa ensimmäisenä parametrina edellä kuvatun listan ja toisena parametrina desimaaliluvun. Funktio palauttaa arvonaan tiedon siitä, kuinka monella osaston työntekijällä palkka on vähintään toisena parametrina annetun luvun verran suurempi kuin osaston kaikkien työntekijöiden palkkojen keskiarvo. Jos esimerkiksi toinen parametri on 100.0, funktio palauttaa tiedon siitä, kuinka monella osaston työntekijällä palkka on vähintään 100 euroa suurempi kuin osaston kaikkien työntekijöiden palkkojen keskiarvo. (20 p)

3. Kauppa on tallentanut tietoja sen kanta-asiakkaiden ostoksista tekstitiedostoon niin, että yhdellä rivillä on yhden asiakkaan yhden ostoksen tiedot. Rivillä on ensin päivämäärä, sitten asiakkaan asiakasnumero ja lopuksi ostoksen arvo. Eri tiedot on erotettu toisistaan puolipisteellä. Tiedoston rivit voisivat näyttää esim. seuraavilta:

```
14.1.2015;A11223;55.30
14.1.2015;B7755;63.75
15.1.2015;A11223;75.00
18.1.2015;A11223;25.80
```

   Kirjoita Python-ohjelma, joka lukee tiedoston läpi ja laskee, kuinka monta yli 50 euron ostosta on käyttäjän haluamalla asiakkaalla. Ohjelma pyytää käyttäjältä tiedoston nimen ja sen asiakkaan asiakasnumeron, jolta yli 50 euron ostosten määrä lasketaan. Jos esimerkiksi käyttäjä antaa asiakkaaksi `A11223` yllä olevassa esimerkissä, pitää ohjelman tulostaa, että asiakkaalla on 2 yli 50 euron arvoista ostosta. Jos halutulla asiakkaalla ei ole tiedostossa yhtään ostosta, ohjelman pitää tulostaa, että asiakkaalla on 0 yli 50 euron arvoista ostosta.

   Ohjelman on käsiteltävä seuraavat virhetilanteet:
   - Annetun nimistä tiedostoa ei ole olemassa tai tiedoston lukeminen ei onnistu jostain muusta syystä
   - Tiedoston jollain rivillä ostosten arvon paikalla ei ole desimaaliluku.

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

4. Eräs nettitiliyritys tarjoaa käyttöön maksutilin, jonka avulla netissä kauppaa käyvät yksityishenkilöt voivat huolehtia tavaroiden maksusta luotettavalla tavalla. Tili toimii siten, että tavaran ostaja perustaa maksutilin ja tallettaa sille rahaa. Kun hän haluaa tehdä oston, ilmoittaa hän maksutilille, että sille tehdään varaus halutulle summalle, joka vastaa myytävän tavaran hintaa. Tiliyritys välittää myyjälle tiedon siitä, että tarvittava varaus on tehty ostajan tililtä. Tällöin myyjä uskaltaa lähettää tavaran ostajalle. Kun ostaja ilmoittaa tiliyritykselle saaneensa tavaran, tiliyritys nostaa varatut rahat ostajan tililtä ja välittää ne myyjälle. Näin ostajan ei tarvitse varsinaisesti maksaa tavaraa ennen sen vastaanottamista, ja toisaalta myyjän ei tarvitse tavaraa lähettäessään ottaa sitä riskiä, että ostaja saa tavaran, mutta ei maksa sitä. Ostajalla voi olla samanaikaisesti kesken useita kauppoja, eli tililtä voi olla varattu rahoja useaan eri tarkoitukseen.

JATKUU

Kirjoita maksutilin tietojen säilyttämiseen luokka `Maksutili`. `Maksutili`-oliolla on oltava seuraavat kentät:

- `__omistaja` tilin omistajan (ostajan) nimi
- `__saldo` tilillä tällä hetkellä oleva rahasumma (sisältää myös voimassa olevien varausten arvon)
- `__varattu` tilillä tällä hetkellä voimassa olevien varausten yhteissumma (tehtävän helpottamiseksi eri ostoja varten tehtyjä varauksia ei tässä mitenkään eritellä, vaan pidetään yllä tietoa vain siitä, mikä on varausten summa yhteensä).

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

- `__init__(self, nimi)` luo uuden `Maksutili`-olion. Tilin omistajan nimi annetaan parametrina. Uuden tilin saldo on nolla eikä tilillä ei ole varauksia.
- `kerro_omistaja(self)` palauttaa tilin omistajan nimen.
- `talleta_rahaa(self, summa)` tallettaa tilille rahaa (eli kasvattaa tilin saldoa) parametrina annetun summan, jos parametri on postitiivinen. Jos parametri on negatiivinen tai nolla, metodi ei tee mitään.
- `tee_varaus(self, summa)` tekee tililtä varauksen parametrina annetulle summalle, jos tilin saldo riittää varauksen tekemiseen ja annettu summa on positiivinen. Tilillä voimassa olevien varausten yhteissumma ei saa varauksen tekemisen jälkeen ylittää tilin saldoa. Jos varauksen tekeminen onnistuu, metodi palauttaa arvon `True`. Jos tilin saldo ei riitä varauksen tekemiseen tai metodille annettu parametri on negatiivinen tai nolla, metodi palauttaa arvon `False`.
- `maksa_varaus(self, summa)` maksaa tililtä aikaisemmin tehdyn, parametrina annetun summan suuruisen varauksen. Käytännössä varauksen maksaminen tarkoittaa sitä, että sekä tilin saldoa että nykyisten varausten yhteissummaa vähennetään annetulla summalla. (Maksun välittäminen myyjälle hoidetaan jotenkin muuten eikä se näy tässä metodissa.) Varauksen maksu onnistuu vain siinä tapauksessa, että parametrina annettu summa on positiivinen ja että se on korkeintaan yhtäsuuri kuin tilillä tällä hetkellä voimassa olevien varausten summa. Jos varauksen maksu onnistuu, metodi palauttaa arvon `True`. Jos varauksen maksu ei onnistu, metodi ei muuta lainkaan tilin saldoa eikä varausten summaa. Tällöin metodi palauttaa arvon `False`. (Tehtävän helpottamiseksi tässä ei siis tarkisteta sitä, että aikaisemmin on tehty varaus, joka on juuri samansuuruinen kuin nyt suoritettava maksu. Riittää, että voimassa olevia varauksia on yhteensä vähintään yhtä suuresta arvosta kuin mitä on nyt suoritettava maksu. Järjestelmä tarkistaa jotenkin muuten, että kukaan myyjä ei saa enempää maksuja kuin mitä häntä varten on tehty varauksia. Tätä tarkistusta ei kirjoiteta tässä tehtävässä.)
- `peru_varaus(self, summa)` peruu tililtä aikaisemmin tehdyn, parametrina annetun summan suuruisen varauksen, jota ei vielä ole maksettu. Varauksen peruminen tarkoittaa käytännössä sitä, että varausta vastaava summa vähennetään voimassa olevien varausten summasta. Varauksen peruminen onnistuu vain, jos annettu summa on positiivinen ja korkeintaan yhtäsuuri kuin voimassa olevien varausten summa. Jos peruminen onnistuu, metodi palauttaa arvon `True`. Jos peruminen ei onnistu, metodi ei muuta mitään ja palauttaa arvon `False`. Tätä metodia voidaan käyttää esimerkiksi kaupan peruuntuessa.
- `__str__(self)` palauttaa merkkijonon, joka sisältää maksutilin omistajan nimen, tilin saldon ja tällä hetkellä voimassa olevien varausten summan.

Kirjoita lisäksi pääohjelma, joka luo kaksi `Maksutili`-oliota ja sen jälkeen tallettaa molemmille niistä rahaa. Sitten pääohjelman pitää tehdä varaus ensiksi luodulle tilille ja tulostaa, onnistuiko varauksen tekeminen. Sen jälkeen ohjelman pitää maksaa varaus ensiksi luodulta tililtä ja kertoa, onnistuiko se. Lopuksi ohjelman pitää tulostaa molempien maksutilien tiedot (omistajan nimi, tilin saldo ja tällä hetkellä voimassa olevien varausten summa). Voit päättää tilien omistajat sekä talletuksissa, varauksessa ja maksussa tarvittavat rahasummat itse. Pääohjelman ei siis tarvitse kysyä mitään käyttäjältä. (25 p)