CSE-A1111 Ohjelmoinnin peruskurssi Y1 Tentti 1.12.2015

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 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). Nämä opiskelijat saavat halutessaan myös sekä suomen- että englannin- tai ruotsinkielisen kysymyspaperin.

Vastaa myös kurssin palautekyselyyn. Kyselyyn vastaamisesta saa 200 harjoitustehtävä-pistettä. Linkki kyselyyn on lähetetty kurssille ilmoittautuneille sähköpostitse.

1. a + b (4 p)
   Vastaa kurssin lopputestiin ja loppumotivaatiokyselyyn. Tämän tentin pisteistä 4 tulee siitä, että vastaat näihin molempiin. Pisteet tulevat testin tekemisestä ja kyselyyn vastaamisesta, ei siitä, mitä olet vastannut. Linkki kyselyihin on syksyn kurssin Goblinin etusivulla. (Myös kesäkurssille osallistuneet voivat kirjautua Aalto-tunnuksilla syksyn kurssin Gobliniin ja vastata näihin. Linkki syksyn kurssin Gobliniin on julkaistu kurssin Noppa-sivulla ja kesäkurssin omalla sivulla.)

   Kohdissa 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.

   c) (3 p)
```
def main():
    lista = [10, 30, 15, 40, 5]
    tulos = 100
    i = 0
    while i < len(lista):
        if lista[i] < 30:
            tulos = tulos - lista[i]
        i = i + 1
    print(tulos)

main()
```

   d) (5 p)
```
def tee_muutoksia(luvut, luku):
    luku = 15
    luvut[2] = luvut[2] + luku
    return luku

def main():
    arvo = 4
    lista = [10, 20, 30]
    tee_muutoksia(lista, arvo)
    print(arvo)
    for maara in lista:
        print(maara)

main()
```

e) Funktiolle annetaan parametreina kaksi listaa, jotka sisältävät kokonaislukuja ja joilla on sama pituus, sekä yksi positiivinen kokonaisluku. (4 p)

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

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

```
def mysteeri2(merkkijono1):
    uusi = "!!!"
    for alkio in merkkijono1:
        uusi += alkio
        uusi += "!"
    uusi += "!!"
    return uusi
```

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

```
def mysteeri3(lista, luku):
    i = 1
    tulos = lista[0]
    while i < len(lista):
        if lista[i] == lista[i - 1] + luku:
            tulos += lista[i]
        else:
            return -1
        i += 1
    return tulos
```

2. a) Kolme autovuokraamoa tarjoaa samanlaista autoa vuokralle erilaisilla ehdoilla. Ensimmäinen vuokraamo veloittaa 140 eur / vrk riippumatta autolla ajetuista kilometreistä. Toinen vuokraamo veloittaa 100 eur / vrk. Tähän hintaan sisältyy koko vuokrausaikana 1200 km. Jos kokonaiskilometrit menevät sen yli, pitää asiakkaan maksaa ylimenevistä kilometreistä 8 snt / km. Kolmas vuokraamo veloittaa kahdelta ensimmäiseltä vuorokaudelta 60 eur / vrk ja lisävuorokausilta 40 eur / vrk, mutta niiden lisäksi asiakkaan pitää maksaa 20 snt jokaisesta ajetusta kilometristä. Kaikissa vuokraamoissa asiakkaan pitää lisäksi maksaa tarvittava polttoaine itse. Kirjoita Python-ohjelma, joka kysyy käyttäjältä, kuinka moneksi vuorokaudeksi auto vuokrataan ja montako kilometriä sillä on tarkoitus ajaa yhteensä. Ohjelma laskee ja tulostaa, mistä autovuokraamosta auto kannattaa vuokrata ja mikä tämän vuokraamon kokonaishinnaksi (ilman polttoainetta) tulee. Jos hinta on sama useammalla vuokraamolla, ohjelman ei tarvitse luetella kaikkia vaihtoehtoja, vaan yhden (minkä tahansa) halvimman vaihtoehdon ja sen hinnan kertominen riittää. (10 p.)

b) Eräässä yrityksessä on menossa kuntokampanja, jossa työntekijä saa bonusta työntekijän askelmittarin mittaamien päivässä kuljettujen askelten määrän mukaan. Jos yhden päivän askelten määrä on pienempi kuin yhtiön määräämä raja1, ei työntekijä saa tästä päivästä bonuspisteitä. Jos taas päivän aikana kuljettujen askelten määrä on vähintään raja1, mutta pienempi kuin raja2, saa työntekijä päivästä yhden bonuspisteen. Jos askelten määrä on vähintään raja2, saa työntekijä päivästä 2 bonuspistettä sekä lisäksi yhden lisäbonuspisteen jokaista täyttä 1000 askelta kohti, jolla askelten määrä ylittää raja2:n. (Esimerkiksi jos raja2 on 9500 ja askelten määrä päivässä 12500, saa työntekijä päivästä 2 + 3 = 5 bonuspistettä.) Jokaiselle päivälle lasketaan siis omat bonuspisteensä sinä päivänä kuljettujen askelten määrän mukaan. Kirjoita Python-funktio `laske_askelbonukset(askeltiedot, raja1, raja2)` Funktio saa ensimmäisenä parametrina listan, joka sisältää erään työntekijän eri päivinä kulkemien askelten määrät kokonaislukuina (listan 1. alkio on 1. päivän askelten määrä, 2. alkio 2. päivän askelten määrä jne.) Toisena ja kolmantena parametrina on edellä kuvatut rajat. Funktion pitää laskea ja palauttaa työntekijälle eri päiviltä yhteensä kertyvien bonuspisteiden summa. Voit olettaa, että funktiolle on annettu järkevät parametrit. Tehtävässä ei tarvitse kirjoittaa muuta kuin pyydetty funktio. *Vinkki:* voit käyttää ratkaisussa hyväksesi katkaisevaa kokonaislukujen jakolaskua // (operaation tulos on jakolaskun tuloksen kokonaisosa kokonaislukuna). (20 p)

3. Erään seiväshyppykilpailun karsintojen tulokset on tallennettu tiedostoon. Yhden kilpailijan karsintatulos on tallennettu yhdelle riville seuraavasti: rivillä on ensin hyppääjän kilpailunumero, sitten nimi, tämän jälkeen suurin korkeus, jonka kilpailija ylitti karsinnassa ja lopuksi tieto siitä, kuinka monennella yrityksellä kilpailija ylitti tämän korkeuden. Eri tiedot on erotettu toisistaan pilkulla. Tiedoston rivit voisivat näyttää esim. seuraavilta:

```
543,Raphael Holzdeppe,560,2
776,U.N. Known,550,2
447,Renault Lavillenie,550,1
```

Voit olettaa, että yhden hyppääjän tiedot ovat tiedostossa vain yhteen kertaan. Loppukilpailuun pääsevät kaikki ne hyppääjät, jotka ovat ylittäneet vähintään 560 (millä tahansa yrityksellä) ja lisäksi ne hyppääjät, jotka ovat ylittäneet 550 ensimmäisellä yrityksellä. Esimerkin ensimmäinen ja viimeinen hyppääjä pääsisivät loppukilpailuun, mutta keskimmäinen ei. Kirjoita Python-ohjelma, joka pyytää käyttäjältä karsintatulokset sisältävän tiedoston nimen. Ohjelma lukee tulokset tästä tiedostosta ja tulostaa kaikkien loppukilpailuun päässeiden hyppääjien kilpailunumerot ja nimet.

Ohjelman on käsiteltävä seuraavat virhetilanteet:

- Annetun nimistä tiedostoa ei ole olemassa tai tiedoston lukeminen ei onnistu jostain muusta syystä.
- Tiedoston jollain rivillä korkeuden tai yrityskerran paikalla ei ole kokonaisluku.

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 tiedos-ton jokaisella rivillä on täsmälleen neljä toisistaan pilkulla erotettua osaa. Ohjelman ei tarvitse osata käsitellä esimerkiksi sellaisia virhetilanteita, joissa rivi on tyhjä tai ei sisällä kilpailunumeron ja nimen lisäksi muuta tekstiä. (20 p)

4. Kirjoita luokka `Kasinopelaaja` erään rahapelin pelaajan kuvaamiseen. Pelissä kukin pelaaja aloittaa ostamalla käyttöönsä haluamansa määrän pelimerkkejä. Päästäkseen peliin mukaan pelaajan on asetettava osa hallussaan olevista pelimerkeistä panoksiksi. Pelissä käydään useita korotuskierroksia, joiden aikana pelaajat voivat korottaa panoksiaan. Jos yksi pelaaja korottaa panoksiaan, muiden pelaajien pitää joko korottaa panoksiaan vastaavalla summalla tai luovuttaa. Jos pelaaja luovuttaa, hän menettää kaikki siihen asti asettamansa panokset, mutta hän ei häviä sen enempää. Pelin lopussa voittaja saa kaikkien pelissä vielä mukana olevien pelaajien panokset sekä lisäksi pelin luovuttaneiden pelaajien ennen luovutusta asettamat panokset. Hävinnyt pelaaja menettää pelin aikana asettamansa panokset, mutta voi pitää vapaana olevat pelimerkit. Jännityksen kasvattamiseksi kasino määrää aina jonkun pelaajan olevan pelissä vaarassa, mikä voi kasvattaa tämän pelaajan voittaessa saamien mutta myös hävitessä menettämien pelimerkkien määrää (selitetty tarkemmin metodien kuvauksissa). Tässä tehtävässä kirjoitettava luokka kuvaa vain yhtä pelaajaa, ei koko peliä. **JATKUU**

`Kasinopelaaja`-oliolla on oltava seuraavat kentät:

- `__nimi` pelaajan nimi
- `__pelimerkit` pelaajalla tällä hetkellä vapaana olevien pelimerkkien määrä (tähän ei siis lasketa mukaan niitä pelimerkkejä, jotka pelaaja on asettanut nykyisessä pelissä panoksiksi)
- `__panokset` niiden pelimerkkien määrä, jotka pelaaja on asettanut käynnissä olevan pelin panoksiksi
- `__vaarassa` kentän arvo on `True`, jos pelaaja on vaarassa ja muuten `False`.

Määrittele luokkaan seuraavat metodit. (Jos metodin kuvauksessa ei ole kerrottu mitään metodin palauttamasta arvosta, metodin ei tarvitse palauttaa mitään. Jos haluat, saat määritellä luokkaan myös muita metodeja, mutta se ei ole pakollista.)

- `__init__(self, nimi1, merkkeja)` luo uuden `Kasinopelaaja`-olion. Pelaajan nimi ja vapaiden pelimerkkien määrä alussa (pelaajan ostamien pelimerkkien määrä) annetaan parametreina. Uusi pelaaja ei ole vielä asettanut panoksia eikä hän ole vaarassa.
- `kerro_nimi(self)` palauttaa pelaajan nimen.
- `kerro_pelimerkit(self)` palauttaa pelaajalla olevien vapaiden pelimerkkien määrän.
- `aseta_vaaraan(self)` vaihtaa pelaajan olemaan vaarassa.
- `poista_vaarasta(self)` vaihtaa pelaajan pois vaarasta.
- `kasvata_panoksia(self, lisapanos)` siirtää parametrina annetun määrän pelaajan pelimerkkejä vapaista pelimerkeistä panoksiin. Tämä onnistuu vain siinä tapauksessa, että parametri on positiivinen ja korkeintaan yhtä suuri kuin vapaiden pelimerkkien määrä tällä hetkellä. Jos parametri ei täytä näitä ehtoja, metodi ei muuta mitään ja palauttaa arvon `False`. Jos panoksien kasvattaminen onnistuu, metodi palauttaa arvon `True`. Tätä metodia käytetään myös panosten asettamiseen pelin alussa.
- `luovuta(self)` metodia kutsutaan, kun pelaaja luovuttaa pelin. Se muuttaa asetettujen panosten määrän nollaksi, mutta ei muuta vapaina olevien pelimerkkien määrää. Metodi palauttaa tiedon siitä, mikä oli panosten määrä ennen nollaamista. Vaarassa olemisella ei ole vaikutusta tämän metodin toimintaan.
- `voitto(self, voitetut_panokset, luovutetut_panokset)` metodia kutsutaan, jos pelaaja voittaa pelin. Pelaajan omat panokset siirretään takaisin vapaina oleviin pelimerkkeihin. Sen lisäksi vapaina oleviin pelimerkkeihin lisätään pelissä voitetut muiden pelaajien panokset (määrä annettu toisena parametrina) ja pelissä luovuttaneiden pelaajien asettamat panokset (määrä annettu kolmantena parametrina). Jos pelaaja on vaarassa, voitetut panokset (toinen parametri) lisätään vapaisiin pelimerkkeihin kaksinkertaisina.
- `havio(self)` metodia kutsutaan, jos pelin päättyessä pelaaja ei voita peliä. Tällöin pelaajan asettamien panosten määrä nollataan. Jos pelaaja on vaarassa, häneltä vähennetään lisäksi vapaita pelimerkkejä yhtä paljon kuin mitä hänellä oli asetettuja panoksia, korkeintaan kuitenkin niin paljon kuin pelaajalla on vapaita pelimerkkejä. Metodi palauttaa arvonaan tiedon siitä, kuinka monta pelimerkkiä pelaajalta yhteensä vähennettiin (sekä panoksista vähennetyt että mahdollisesti vapaista pelimerkeistä vähennetyt yhteensä).
- `__str__(self)` palauttaa merkkijonon, joka sisältää pelaajan nimen, vapaiden pelimerkkien määrän, pelaajan tällä hetkellä asettamien panosten määrän ja joko tekstin "on vaarassa" tai "ei ole vaarassa" sen mukaan, onko pelaaja tällä hetkellä vaarassa.

Kirjoita lisäksi pääohjelma, joka luo kaksi `Kasinopelaaja`-oliota, kasvattaa kaksi kertaa kummankin pelaajan panoksia ja tulostaa jälkimmäisen pelaajan viimeisestä kasvatuskerrasta, onnistuiko panosten kasvatus. Sitten pääohjelma kutsuu ensiksi luodulle pelaajalle havio-metodia ja tulostaa, montako pelimerkkiä pelaaja menetti. Lopuksi ohjelman pitää tulostaa molempien pelaajien tiedot (nimi, vapaiden pelimerkkien ja asetettujen panosten määrä ja tieto siitä, onko pelaaja vaarassa). Pääohjelman ei tarvitse kysyä mitään käyttäjältä, vaan voit päättää olioiden luonnissa ja metodien kutsuissa tarvittavat tiedot itse. Tehtävän yksinkertaistamiseksi pääohjelma ei siis sisällä koko pelin pelaamista, vaan vain osan siitä.
(25 p)