Ohjelmoinnin perusteet
Harjoitustyö

Harjoitustyön toteutus

1. Tehtävä toteutetaan yksin tai ryhmässä.
2. Valitkaa aihe tämän dokumentin lopusta löytyvästä listasta. Huomaa, että myös oma aihe on sallittu.
3. Työstä kirjoitetaan suunnitelma (kuvaus alempana), joka palautetaan ViLLEstä löytyvän harjoitustyöpalautuksen avulla. Kun suunnitelma on palautettu, työlle nimetään ohjaaja, joka joko hyväksyy suunnitelman sellaisenaan tai pyytää siihen tarvittavat muutokset.
4. Ryhmä voi tarvittaessa sopia erillisiä ohjaustapaamisia työn ohjaajan kanssa.
5. Lopullinen työ (kuvaus alempana) palautetaan ViLLEen.

Suunnitelma

Suunnitelman tulee sisältää:

- Aiheen numero ja mahdollinen lisäselvitys aiheesta (esim. valittu peli sääntöineen); mikäli kyseessä on oma aihe, täytyy se kuvata tarkemmin.
- Lyhyt kuvaus ratkaisuperiaatteesta (n. ½ - 1 sivua), eli minkälaisia toteutusongelmia odotetaan tulevan ja miten näihin varaudutaan. Esimerkiksi, miten toteutetaan pelin vuoropohjaisuus tai kalenterin tiedontallennus?
- Ratkaisussa tarvittavat tärkeimmät metodit; eli millä tavalla ohjelman suunnitellaan rakentuvan.

Lopullinen työ

Harjoitustyön tekniset vähimmäisvaatimukset:

- 100 lausetta kerrottuna ryhmän koolla (2hlö = 200 lausetta, jne.). Kts. määritelmä seuraavalta sivulta
- Kolme metodia
- Käytössä vähintään yksi lista.
- Käytössä vähintään yksi hajautustaulu.
- Ohjelman aloitusdata (jos sellaista on) pitää lukea tiedostosta. Kovakoodatut tietorakenteet ei sallittuja.
- Käyttöliittymä
- Tiedosto-operaatioita (luku ja kirjoitus)

Lopullinen työ palautetaan ViLLEen. Palautuksen tulee sisältää (pakattuna zip-, rar- tai vastaavaan pakettiin):

- Kaikki työhön kuuluva python-lähdekoodi .py-tiedostoina
- Mahdolliset muut suoritukseen tarvittavat tiedostot (kuvat, tekstitiedostot jne.)
- Harjoitustyön dokumentti, jonka tulee noudattaa IT-laitoksen yleisiä harjoitustyön dokumentointiohjeita (moodlessa)
- Maksimissaan yhden sivun ohje työn ajamiseksi
- **Lista ryhmän jäsenten vastuualueista työn toteuttamisessa**

Työt arvostellaan asteikolla hylätty / hyväksytty.

Harjoitustyön aiheet

Peli

Toteuta mikä tahansa peli. Pelissä tulee olla käyttöliittymä (komentorivipohjainen tai graafinen) ja sen tulisi mahdollistaa pelin tilan tallentaminen ja lataaminen. Lisäksi peli voi esimerkiksi pitää kirjaa pelaajista, jotka ovat saavuttaneet parhaat pisteet.

Kalenteri

Kalenterisovelluksen tulisi mahdollistaa erilaisten merkintöjen lisääminen kalenteriin (tapahtumat, tehtävät, jne.), ja niihin pitäisi pystyä lisäämään muistutuksia. Sovelluksen käyttöliittymän ei tarvitse olla perinteisen kalenterinäkymän näköinen, vaan riittää, että kaikki toiminnallisuudet löytyvät ja niitä on mahdollista käyttää tekstipohjaisen (tai graafisen) käyttöliittymän kautta. Kalenteri tila pitää voida tallentaa tiedostoon; käynnistettäessä sovellus sen pitää automaattisesti hakea tila tiedostosta.

Oma aihe

Jokin ohjelma omaan tarpeeseen. Huomioi aiheen valinnassa työn minimivaatimukset. Kannattaa valita jokin yksinkertainen ohjelma, johon voi tarpeen mukaan lisätä lisää toiminnallisuutta, jotta minimivaatimukset täyttyvät.

Lauseen määritelmä

Lauseitahan ovat metodikutsut, asetus-,valinta-,toistolauseet, jne. Seuraavassa koodiesimerkissä lasketaan olevan 4 lausetta, jokainen omalla rivillään:

```
x = ”Anna syöte:” #Asetuslause
mjono = input(x) #Asetuslause; metodikutsu katsotaan lausekkeeksi
for i in range(3): #for-lause
    print(i) #Metodikutsu


#Allaolevat ovat kovakoodattuja tietorakenteita,
#eli tietorakenteen sisältö syötetään ohjelmakoodissa.
#Jos arvot tiedetään ohjelman alussa, ne pitää lukea tiedostosta.
elain = {}
elain[’nimi’] = ”Toffo”
elain[’paino’] = 2000
elain[’laji’] = ”Kissa”
elain[’vari’] = ”Oranssi”

matriisi =[
[1,2,3],
[4,5,6],
[7,8,9]
]

#Pitkät lauseet jaettu monelle riville on edelleen vain yksi lause

if x == 0 or x == 1\
or x == 2 or x == 3 \
or x == 4 or x == 5 or x == 7: #yksi if-lause
    print(“Oikein”) #yksi metodikutsu
```