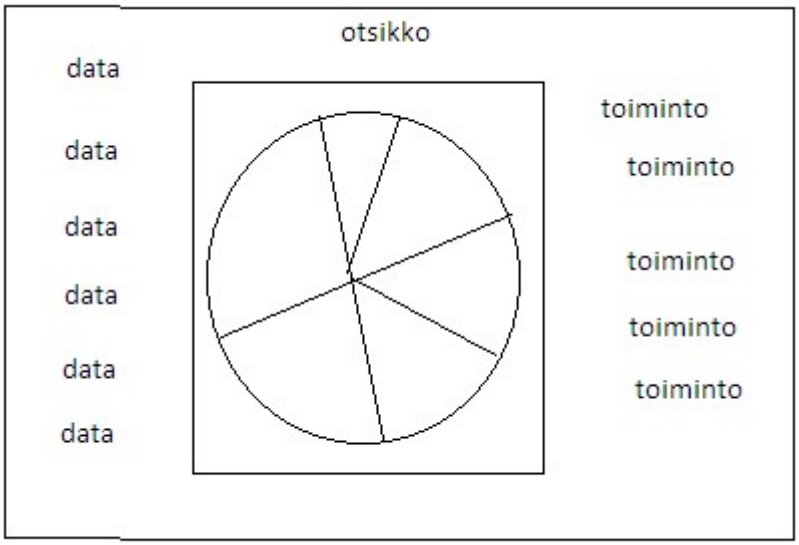
Projektisuunnitelma

Henkilötiedot
Rahan seuranta; Teerialho Veikko 735045, Kemiantekniikka, 2019, 20022021

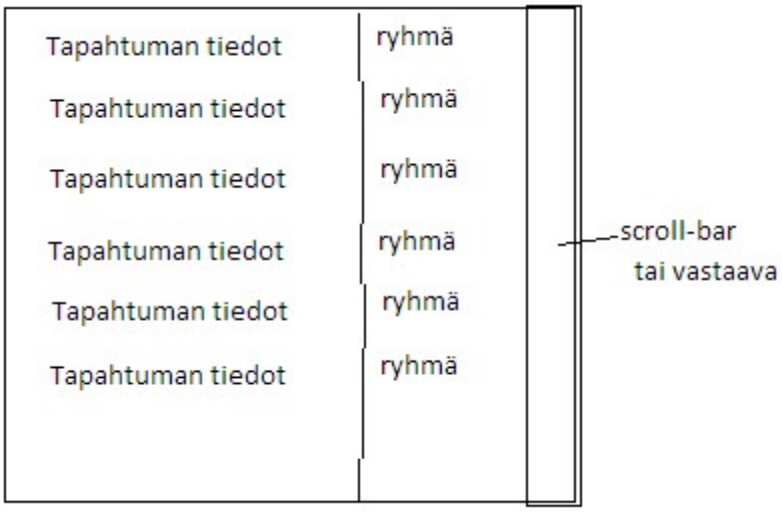
Yleiskuvaus ja vaikeustaso
Rahan seuranta tarkoittaa tässä ohjelmaa, joka purkaa tilitietoja ja kategorisoi niitä ryhmiin havainnollistaen miten paljon ja mihin rahaa kuluu. Ohjelmaan tulisi voida lisätä myös uusia tapahtumia käsin.

Tavoitteena on tehdä vähintään ns helpon ja keskivaikean osuus. Vaikean osuudessa on puhetta tekoälyn tekemästä tärkeysarvioista, niin se lienee projektin suurin kysymysmerkki.

Käyttötapauskuvaus ja käyttöliittymän luonnos
Käyttöliittymässä päätoimintona on datan havainnollistus. Eli tää on se pääajatus, mitä ajolla haetaan.



Tietoja pitänee ryhmitellä jotenkin. Sekä käytön että debuggauksen kannalta olisi fiksua pystyä selaamaan luettua ja käytettyä dataa jotakuinkin näin.



Ohjelma lukee pankin autogeneroimia csv-tiedostoja. Esimerkkirivi näyttää tältä:
04.02.2021;04.02.2021;-580;"162";PKORTTIMAKSU;"Teknologforeningen ESPOO";;"";Viesti:
492057******8249 OSTOPVM 210203 MF NRO 74920541035165088157602 VARMENTAJA 400
;20210204/5EQEO1/037630

Mistä käytetyt tiedot olisi kirjauspäivä, summa ja saaja(28.01.2021, -580, "Teknologforeningen ESPOO”). Tapahtumat jaetaan alaryhmiin sen mukaan kuka rahan on saanut (Teknologforeningen) ja sen jälkeen yläryhmään minkälaisesta kulutuksesta on kyse (ruoka). Tehtävänannossa puhuttiin optionaalisesta toiminnosta, joka luokittelisi kulutusta käyttäjänsyöttämän säästötavoitteen mukaisesti. Tämä toteutunee yläryhmän osalta, jotta luokittelun määrä pysyy kohtuullisena. On helpompaa luokitella kuinka helppoa on säästää, jos luokiteltavia ryhmiä on vähemmän.

Ensisijaisesti luokitteluperusteena käytetään rahan saajaa. Jos rahan saaja on sama niin todennäköisesti sieltä ostetaan tavaraa tai lähtökohtaisesti kuuluu yhteen ryhmään. Ohjelmassa olisi hyvä olla toiminto, joka antaisi käyttäjän halutessaan itse muokata ryhmittelyä.

Ohjelman rakennesuunnitelma

Käyttäjäsyöte tiedostoon
Tiedoston luku
Rivien käsittely
Rahan kulutuksen summaaminen per saaja (alaryhmä)
Rahojen saajien ryhmittely(yläryhmä)
Käyttäsyöte ryhmittelyyn
Yläryhmien tietojen mallinnus
Optional: Käyttäjäsyöte muokkaus
Optional: Käyttäjäsyöte säästötavoite
Ryhmien priorisointi
Tietojen mallinnus

Suunnitelman tärkein osuus. Ohjelman erottelu tärkeimpiin osakokonaisuuksiinsa, suunnitellun luokkajaon esittely. Minkälaisilla luokilla kuvaat ohjelman ongelma-aluetta? Mitä ongelman osaa kukin luokka mallintaa? Mitkä ovat luokkien väliset suhteet? Entä millaisia luokkia tarvitaan ohjelman käyttöliittymän kuvaamiseen? Miettikää mahdollisia muita ratkaisumalleja ja perustele valittu ratkaisu. Liittäkää mukaan jonkinlainen graafinen luokkakaavio (esim. UML luokkadiagrammi). Esittele luokkien keskeiset metodit. Huom. oleellista on vain se, mitä metodeilla tehdään, ei se, miten ne sisäisesti toimivat.

Tietorakenteet

tapahtumat = [[tapahtuma-id, kohde, ryhmä, laatu]]
alaryhmät = {alaryhmä: ryhmänsaldo}
yläryhmät = {yläryhmä: [alaryhmä, alaryhmä]}
prioriteetit = {yläryhmä: arvo}

Tieto kasaantuu osissa, koska rivit luetaan yksittäin. Lisäksi käyttäjä voi korjaa tietoja jälkikäteen, jolloin muuttuvat ja joustavat ryhmät kuten listat ja sanakirjat ovat todennäköisesti hyviä. Rakenteen ideana on ensin tehdä alaryhmien tietojenkäsittely, sitten jaottelu yläryhmiin ja sitten mahdollinen priorisointi. Itse data ei vaadi mitään monimutkaisia rakenteita, fokus tulee olemaan liittymässä.

Tiedostot ja tiedostoformaatit
Ylempänä jo käytiin, että ensisijainen formaatti on pankin autogeneroimat csv:t. Näitä ensisijaisesti luetaan, mutta mikä ettei ohjelma voisi myös tehdä itselleen ns ”tallennusta” esimerkiksi lukien itse ensin useamman csv:tä ja sitten luomalla vapaamuotoisen tiedoston, jossa olisi kaikkien tiedot. Jos aikaa jää niin mikä ettei ohjelmaan voisi lisää jotain kivoja kuvia, mutta piirasdiagrammiin lienee (en vielä tarkistanut) jokin työkalu QT:n sisällä.

Algoritmit

Itse ohjelman päätoiminto on lähtökohtaisesti pluslaskuja. Säästämistoiminnon osalta näkisin mahdollisena toimintatavan, jossa prioriteetit ovat integerejä esimerkiksi välillä 0 – 2 ja arvoilla 0 niistä ei voi säästää, arvolla 1 niistä voi periaatteessa säästää, mutta ensin säästetään ryhmää 2. Toinen mahdollinen toteutustapa olisi että jos pitää säästää 30 euroa niin se säästetään ryhmien 1 ja 2 välillä suhteessa 1:2, jolloin ryhmän 1 kulutuksesta säästetään 10 euroa ja 2 kulutuksesta 20 euroa. Yläryhmien prioriteetit voi kovakoodata sisään. On yleisesti hyväksyttyä, että esim ruoasta tai asumisen kuluista ei voi säästää samaan tapaan kuin vaikka liikunnasta tai viihteestä.

Testaussuunnitelma

Kehitysvaiheessa voi tehdä yksittäisten funktioiden testejä, joissa ideana on estää ettei koodia vahingossa rikota kesken kehityksen. Tärkeämpänä itse näen ns käyttäjätoimintojen toimintaa testaavat testit. Ohjelma ohjelmalla käsitellään paljon float-dataa, jolloin luvut pidetään lähtökohtaisesti koko ajan floattina. Käyttäjäsyötteellä keksitään arbitääriset rajat estämään esimerkiksi rolleveria tmv. Lisäksi syötteiden anto rajataan vahvasti eli syötteitä voi antaa vain ja ainoastaan rajatuissa paikoissa, niiden muotoa rajataan ja ensisijaisesti ohjelma toimii ”putki”-meiningillä, eli ohjelman suoritus menee tiettyä rataa pitkin ja kun ollaan edetty tiettyyn kohtaan niin siitä ei enää palata.

Käyttäjätoimintojen testaukseen tulee vakiotiedosto, joka on vastaava kuin mihin ohjelmaa tullaan käyttämään. Tästä katsotaan valmiiksi mitä arvoja ohjelman tulee saada ja nämä kovakoodataan ohjelmaan sisään. Tiedosto voi olla hyvinkin lyhyt, kunhan se kattaa tärkeimmät rakenteet. Esimerkiksi riittää että yläryhmään kuluu 2 alaryhmää, niitä ei tarvitse olla 20.

Kirjastot ja muut työkalut
Kielletyt kirjastot: MatPlotLib, QT Designer

Koska kurssista halutaan päästä läpi, niin näitä ei käytetä. Lähtökohtaisesti mennään QT:lla. Jos hätä tulee niin diagrammi vaihdetaan piiraasta pylvääseen, niin sen jälkeen pärjäisi jo melkein pelkällä tekstipohjalla.

Aikataulu

Yrittäkää hahmotella itsellenne aikataulu ja karkea arvio eri vaiheissa kuluvista työtunneista. Se on vaikeaa, mutta opettavaista, kun aikataulua vertaa käytännössä eri vaiheisiin kuluvaan todelliseen aikaan. Älä heitä suunnitelmaan mitä tahansa ensiksi mieleen tulevia lukuja vaan yritää arvioida tilannetta realistisesti.

Kuvaa myös suunniteltu etenemisjärjestys, eli missä järjestyksessä ohjelman aiotte toteuttaa. Tässä kannattaa jo miettiä ohjelman testausta, ja sitä, mitkä osat muusta koodista käyttävät muita osia. Yleensä ohjelmassa on keskeisiä ominaisuuksia joista kannattaa aloittaa, sekä ominaisuuksia, joiden toteutus ei ole heti olennaista. Muista että kehitettäessä jotkin asiat voidaan korvata tynkäluokilla tai tynkämetodeilla, jotta ohjelma saadaan kääntymään, ja jotta toteutettua koodia voi testata. Kokonaisia luokkia ei myöskään tarvitse toteuttaa alusta loppuun vaan tarkentaa toteutusta aste asteelta, kun työ etenee.

Kirjallisuusviitteet ja linkit

QT-kirjastot ja pythonin oman rakenteen kirjastot. Allekirjoittaneella on Python 3.7, joten se lienee ensisijainen.

Liitteet

””