Rahan seuranta ohjelma

Kimi Kuru, 790695, kone- ja rakennustekniikka, 23.4.2021

2. Yleiskuvaus

Projektina on luotu rahan seuranta -ohjelma, joka lukee Osuuspankin tarjoamasta tilitapahtumat -tiedostosta kaikki tapahtumat, erottelee menot ja muodostaa niistä piirakkadiagrammin/Donitsidiagrammin. Käyttäjä pystyy jaottelemaan menoja kategorioihin. Käyttäjä saa säästö toiminnolla graafisen kuvan menojen jakautumisesta valitun säästösumman jälkeen.

3. Käyttöohje

3.1 Ohjelman käynnistys ja tiedostonluku

Ohjelma käynnistetään avaamalla GUI.py tiedosto. Ohjelman käynnistämisen jälkeen näkyy tyhjä kuvaaja-alue sekä oikealla nappeja kuvaajan ja datan hallitsemiseen. Kun tiedostoa ei ole vielä luettu kuvaaja-alueen yläpuolella lukee kehottava teksti: ”lisää tiedosto”. Tiedostonluku tapahtuu valitsemalla vasemmasta yläkulmasta ”Tiedosto” > ”Open File”, jolloin avautuu tiedostodialogi, jossa voi valita csv tyyppisiä tiedostoja. Jos tiedoston sisältö on oikeanlainen (OP:n tilitapahtumat tiedosto) tiedosto luetaan onnistuneesti. Jos valittu tiedosto on osittain virheellinen, luetaan tiedoston arvoja virhekohtaan saakka, jonka jälkeen ilmoitetaan virheestä virheikkunalla. Jos tiedosto on kokonaan virheellinen (ei yhtäkään oikein luettavaa kohtaa) ilmoitetaan tästä virheikkunalla.

3.2 Kuvaaja-alue sekä ”yhdistä kuluja”, ”erota kategoria” ja ”näytä jaottelu” toiminnot

Kuvaaja-alueella näkyy ylhäällä otsikko, vasemmalla menojen kuvaukset ja oikealla itse kuvaaja. Kuvaaja-alue on vuorovaikutteinen. Siirtämällä hiiren menon kuvauksen päälle, tummentuu menon vieressä olevan laatikon reunat. Samalla tavalla siirtämällä hiiri kuvaajan päälle tummentuu

kuvaajan palasen reunat. Kuvaajaa kaksoisklikkaamalla voidaan suurentaa kuvaajaa, ja uudestaan kaksoisklikkaamalla pienentää.

Edellisistä kohdista klikkaamalla voidaan valita haluttuja menoja. Menoa klikatessa menon vieressä olevan neliön reunat jäävät tummiksi ja menoa vastaava osa irtoaa muusta kuvaajasta, näistä tiedät menon olevan valittuna. Kun menoja on valittuna otsikossa näkyy valittujen menojen prosentuaalinen osuus koko summasta.

Valittuja menoja voi yhdistää kategorioiksi ensin valitsemalla vähintään yhden menon, sitten valitsemalla ”Yhdistä kuluja”. Nyt aukeaa ikkuna, jossa voit syöttää haluamasi kategorian nimen (Vai n kirjaimia, pituus 1-24). Painamalla ”ok” kategoria halutulla nimellä luodaan. Kun ”näytä jaottelu” ei ole vielä päällä, näkyy luotu kategoria pienellä punaisella tekstillä menon kuvauksen kohdalla. Jos haluat lisätä lisää menoja jo luotuun kategoriaan voit toistaa äskeisen prosessin eri menolla. Kun kirjoitat jo olemassa olevan kategorian nimen, meno lisääntyy jo olemassa olevaan kategoriaan.

Painamalla ”Näytä jaottelu” saa näkyviin kuvaajan, jossa näkyy luodut kategoriat ja loput menot. Jaottelun ollessa päällä näkyy ”Yhdistä kuluja” napin sijasta ”erota kategoria” nappi. Voit erottaa kategorioita valitsemalla kategorian ja painamalla ”erota kategoria”. Jos kategorioita ei ole enää jäljellä, ohjelma palautuu automaattisesti ei jaoteltuun tilaan.

3.3 Kuvaajan tyylit

Ohjelmassa on muutamia vaihtoehtoja muokata kaavion ulkoasua. Ohjelman oikealla ylhäällä sijaitsee valikko, josta painamalla voi valita ”Piirakkadiagrammin” tai ”Rengasdiagrammin”. Valinta on aluksi automaattisesti piirakkadiagrammissa, käyttäjä voi valita ”Rengasdiagrammi”, jolloin kaavion tyyli muuttuu donitsin muotoiseksi. Valikon vieressä sijaitsee ”vaihda värit” nappi, joka vaihtaa painalluksesta kaavion värit eri satunnaisiksi väreiksi.

3.4 Säästötoiminto

Ohjelmassa oikealla alhaalla on nappi ”säästä”, jota painamalla ohjelma luo automaattisesti ehdotetun säästöjärjestyksen, ja näyttää sen erillisessä ikkunassa. Ohjelma säästää menoista järjestyksessä ylhäältä alaspäin. Käyttäjä voi vielä halutessaan muuttaa ehdotettua järjestystä,

klikkaamalla ja raahaamalla menoja haluamaansa paikkaan. Käyttäjä voi syöttää ikkunassa olevaan kenttään haluamansa säästösumman (väliltä 1 – (kokosumma-0.1)) ja hyväksyä säästön painamalla ”Hyväksy”, jolloin ohjelma luo pääikkunan kaltaisen kuvaajan, jonka otsikkoon tulee lukemaan punaisella tekstillä säästetty summa, ja jossa näkyy kuvaaja säästetyillä menoilla. Säästökuvaajaa voi tarkastella samalla tavalla kun ”pääkuvaajaa”, sillä poikkeuksella, että kategorioita ei voi poistaa tai lisätä. Säästöikkunaa voi siis ajatella sen hetkisenä kuvana pääikkunasta.

4. Ulkoiset kirjastot

Ohjelmassa on käytetty ulkoisena kirjastoina tehtävänannossa määritettyä PyQt kirjastoa, jolla voidaan luoda käyttöliittymä ja siihen liittyviä toimintoja, sekä pyqtgraph kirjastoa.

5. Ohjelman rakenne

5.1 Rakenne yleisesti

Ohjelma koostuu luokista ”Dataread”, joka on luokka tiedostonlukuun. Luokat ”Expense”, ”Expenseobj” ja ”Expensecategory” kuvaavat ja säilyttävät ohjelmassa käytettävän datan eri muotoja (ostotapahtuma, meno, kategoria), joiden avulla pysytään ajan tasalla nykyisten menojen jakautumisesta, ja kun lisätään dataa, kuvaajan muutosten lisääminen helpottuu. Luokka ”Series” mallintaa kuvaajaan syötettävää dataa. Luokka ”allExpenses” säilöö ja luo datan eri muodot sekä kuvaajaan syötettävän datan listoissa, sekä suorittaa säästötoiminnon laskut. Luokat ”WidgetWin” ja ”MainWin” luovat ohjelman koko käyttöliittymän.

5.2 Luokat selitetty

MainWin

MainWin luokka kuvaa ohjelman pääikkunaa. MainWin sisältää tiedostodialogin lukemisen ja aloittaa käyttäjän valitsemalla tiedostolla Dataread luokan, jossa tiedosto luetaan. MainWin saa Dataread olion luoman allExpenses olion, joka syötetään luotavalle WidgetWin oliolle.

Dataread

Dataread luokan tarkoitus on erotella menot käyttäjän valitsemasta tiedostosta, Aloittaa allExpenses olio ja luoda sinne Expense oliot eli ostotapahtumaa kuvaavat oliot.

allExpenses

AllExpenses luokan tarkoitus on luoda ostotapahtumista (Expense), yksittäisiä menoja (Expenseobj). AllExpenses luokka luo myös Expensecategoryja eli kategorioita. AllExpenses luo Series oliot, jotka sisältävät kuvaajaan piirrettävän datan (Jaoteltu ja ei jaoteltu). AllExpenses luokka säilyttää ja muokkaa listaa jaottelemattomista (expenseobjs) ja jaotelluista (ToPlot) menoista, sekä jaottelemattomia, että jaoteltuja Series olioita.

Se, että pidetään listaa Jaotelluista ja ei jaotelluista Expenseobj ja category olioista, ja luodaan erikseen oliot piirrettävistä ”serieksistä” helpottaa kuvaajan datan (eli juuri serieksen) muokkaamista ja datan lisäämistä.

Ohjelmassa siis luodaan ostotapahtumista samaan aikaan Expenseobj oliot (menot) ja lisätään samalla ostotapahtumat seriekseen. Oliot toimivat tietynlaisina tietorakenteina, jossa tieto nykyisen kuvaajan menoista säilyy hyvässä järjestyksessä ja helposti saatavana.

Vaihtoehtona toteutetulle ostotapahtumien käsittelylle (joka oli itseasiassa oma alkuperäinen toteutus) olisi ollut luoda niistä ensin Expenseobj oliot, ja niistä muokata ja kopioida kuvaajaan data. Kuitenkin, kun tiedostoja lisätään enemmän, ja listat kasvavat, niin tämän prosessin suorituskyky heikentyisi aina dataa lisätessä, koska kokonaisuudessaan kopioitavaa on enemmän. Toisinkuin toteutetussa tavassa, jossa suoritusnopeus riippuu ainoastaan lisättävän tiedoston koosta.

Expenseobj

Expenseobj luokka kuvaa menoa yhteen kauppaan. Sillä on nimi, arvo, päivämäärät, tärkeys ja tieto kategoriasta minkä sisällä on, jos on jossain kategoriassa.

Expense

Expense luokka kuvaa jokaista yksittäistä ostotapahtumaa. Sillä on nimi, arvo, päivämäärä

Expensecategory

Expensecategory luokka kuvaa käyttäjän luomaa kategoriaa. Expensecategory perii Expenseobj:n ominaisuudet. Lisäksi Expensecategory sisältää tiedon sisältämistään menoista (Expenseobj).

Series

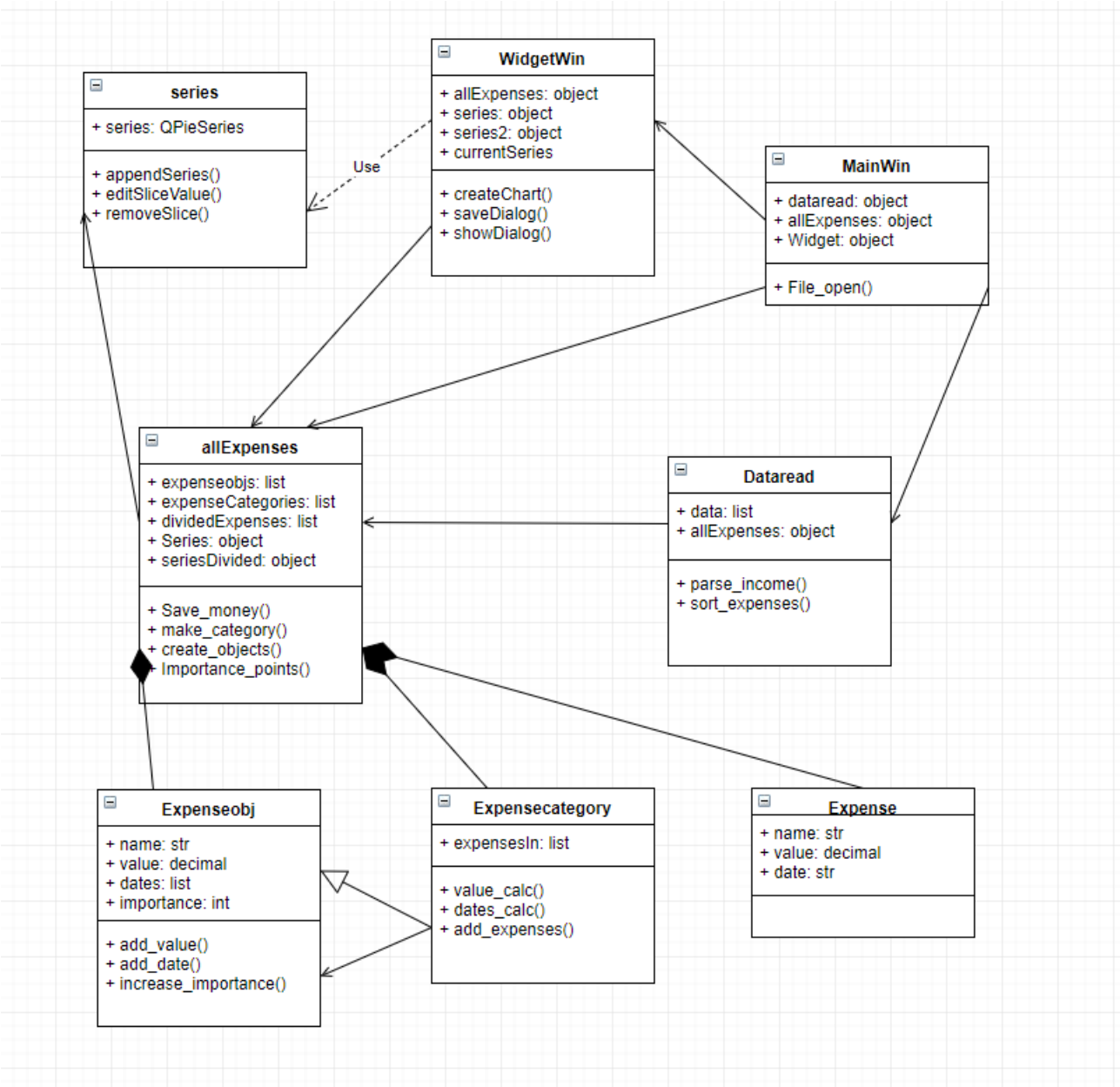
Series luokka kuvaa kuvaajaan lisättävää dataa. Luokassa luodaan PyQt:n series luokka. Samalla kun allExpenses oliossa luodaan Expenseobj:t, luodaan seriesDivided (jaoteltu data) ja Series (ei jaoteltu).

WidgetWin

WidgetWin luokka luo kaikki PyQt:n widgetit, kuten napit ja kuvaajat. WidgetWin saa luodun allExpenses olion. WidgetWinissä yhdistetään myös nämä toiminnot allExpenses luokan toimintoihin esim. mitä tapahtuu kun nappia painaa.

5.3 Luokkakaavio

alla olevassa luokkakaaviossa esitetään tarkemmin luokkien väliset suhteet, ja tärkeimmät metodit.



(Kompositiot merkitty allExpensesille siksi, että se säilyttää eri ”expense..” olioita muuttujissaan (käytännössä koostuu niistä), ja metodeissa suorittaa niille operaatioita. Jos allExpenses luokka poistettaisiin, ei koko muulla ohjelmalla olisi enää tietoa yksittäisistä ”expenseistä”)

Tärkeimmät metodit

allExpenses luokka

Save_money() metodi laskee uudet ”säästöseriekset”, joista säästetty haluttu summa, ja jotka piirretään säästökuvaajaan.

Make_category() metodi luo uuden kategorian (Expensecategory), ja lisää sen ”jaoteltuihin” listaan ja ”seriekseen”, joista piirretään jaoteltu kuvaaja

Create_objects() metodi luo ostotapahtumista (Expense), yksittäisiä menoja (Expenseobj) ja lisää ne listoihin ja ”seriekseen”, joista piirretään kuvaajat.

Importance_points() metodi laskee menoille (Expenseobj) tärkeyspisteet, joiden mukaan niistä save_money() metodissa säästetään.

WidgetWin luokka

CreateChart() metodi päivittää aktiivisen kuvaajan (joko jaoteltu tai ei jaoteltu) kuvaukset ja otsikon kun dataa on lisätty.

SaveDialog() metodi luo säästöikkunan, jossa näkyy säästettävät menot järjestyksessä, järjestystä voi muuttaa raahaamalla. Metodissa on myös numerokenttä johon syötetään säästettävä summa, ja ”hyväksy” napista ohjelma luo säästökuvaajan.

ShowDialog() metodi luo ikkunan kategorian luomiselle. Metodissa on tekstikenttä, johon syötetään kategorian nimi, ja ”hyväksy” nappi

MainWin luokka

File_open() metodi avaa tiedostodialogin, ja syöttää käyttäjän valitseman tiedoston Dataread objektille, josta saadun allExpenses objektin syöttää luotavalla WidgetWin objektille

Dataread luokka

Parse_income() metodi erottelee tiedostosta kulut ja lisää ne listaan.

Sort_expenses() metodi erottelee kuluista nimet, arvot ja päivämäärät, ja luo niistä allExpenses objektiin ostotapahtumia (Expense)

Series luokka

AppendSeries() metodi lisää arvoja ”seriekseen”

editSliceValue() muokkaa ”serieksen” arvoja tietyllä indeksillä

removeSlice() metodi poistaa arvon serieksestä

Expenseobj luokka

add_value() lisää menolle arvon

add_date() lisää menolle päivämäärän

increase_importance() kasvattaa menon tärkeyttä

Expensecategory luokka

value_calc() metodi laskee kategorian sisältämien menojen (Expenseobj) arvot yhteen koko kategorian arvoksi.

dates_calc() metodi laskee kategorian sisältämien menojen (Expenseobj) päivämäärät yhteen listaan.

add_expenses() metodi lisää menon (Expenseobj) kategoriaan.

6. Algoritmit

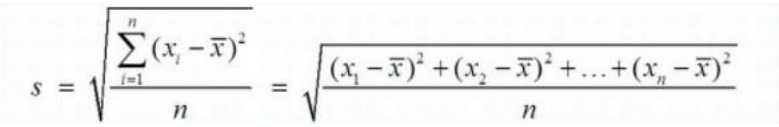
Tärkeyspisteiden laskeminen (säästämistekoäly)

Tarkoituksena oli luoda järjestys tärkeimmistä menoista. Tärkeyspisteiden laskemiseen käytetty algoritmi perustuu kaupan käyntikertojen määrään, käyntikertojen välien säännöllisyyteen ja kauppaan käytettyjen summien suuruuteen.

Niitä ominaisuuksia mistä tärkeyspisteitä saa tai mistä johtuen niitä kenties poistetaan voisi olla paljon useampia, mutta tässä työssä ei ole tarkoitus määrittää loputtoman tarkasti käyttäjän menojen tärkeyttä, vaan antaa edes suuntaa sille, miten ne voisi olla järjestetty. Toteutuksen ”tekoälyssä” on käytetty vain pisteitä lisääviä ominaisuuksia, koska ominaisuuksia on itsessään vain muutama.

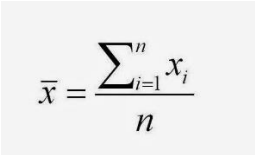
Kaavat:

Keskihajonta



$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}} = \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2}{n}}$$

keskiarvo



$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Algoritmi laskee kaikkien menojen arvojen keskiarvon ja keskihajonnan, kaikkien menojen käyntikertojen aikavälien keskihajonnan. Algoritmi antaa yksittäisistä käyntikerroista yhden (1) pisteen. Yksittäisten menojen käyntikertojen aikavälien keskihajontaa verrataan kaikkien menojen käyntikertojen aikavälien keskihajontaan, jos yksittäisen menon keskihajonta on pienempi kuin kaikkien menojen, saa meno kaksi (2) tärkeyspistettä. Jos yksittäisen menon arvo on suurempi kuin kaikkien arvojen keskiarvo + keskihajonta, saa meno kuusi (6) tärkeyspistettä.

Järjestyksen luomisessa käytetään lopuksi sort -algoritmia, joka järjestää menot pienimmästä tärkeyspistemäärästä suurimpaan.

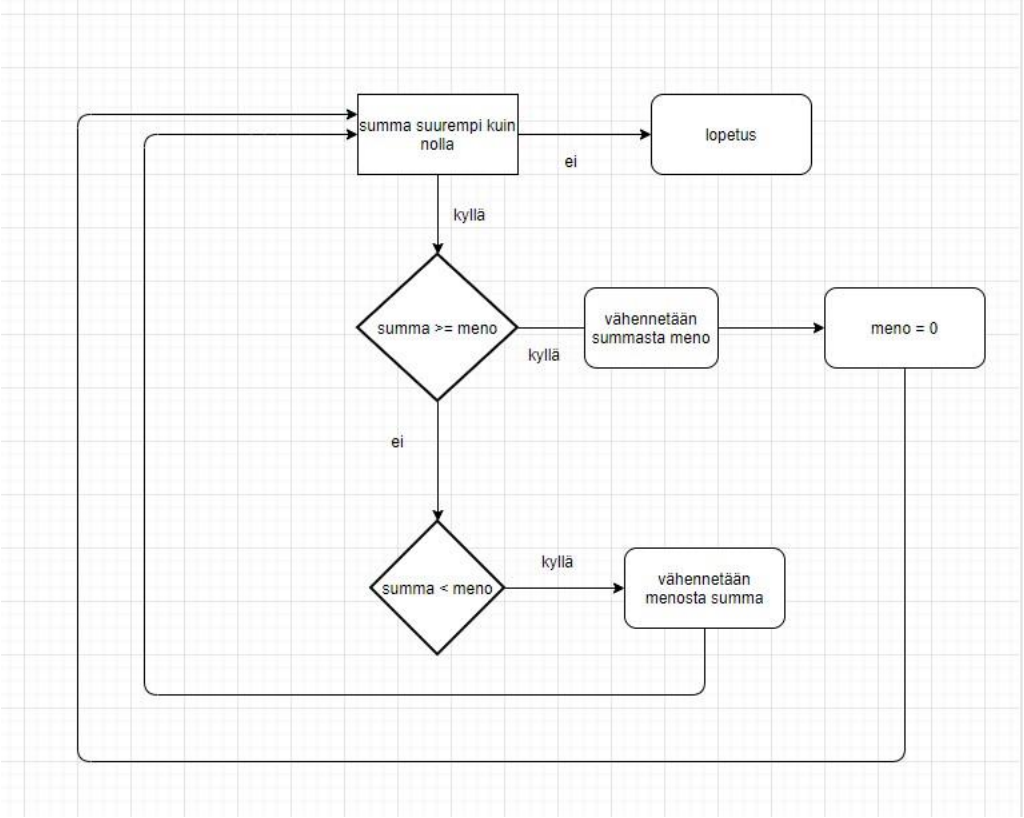
Menojen tärkeys on hyvin suhteellinen aihe, jonka takia tärkeyspisteet numeroina on kätevä toteutus. Tärkeyspisteet ovat yksinkertainen, mutta toisaalta suhteellisen tarkka tapa laittaa menot järjestykseen. Lisäksi se sopii hyvin tärkeyden ”suhteellisuuden” takia, koska se on herkkä ja helposti muutettava tyyli.

Edelleen suhteelliseen ongelmaan ”Menojen tärkeys” on turvallinen vaihtoehto käyttää keskiarvoa ja keskihajontaa, jotka ovat hyviä kuvaamaan epätarkkaa ongelmaa, jossa on vaikea antaa suoria arvoja ja joka riippuu useasta eri asiasta. Vaihtoehtoisesti olisi voinut käyttää esimerkiksi, varianssia tms. mutta se ei luo yhtä tasaista kuvaa koko tarkastelujoukosta.

Rahan säästäminen

Rahan säästämisessä algoritmin tarkoitus on vähentää säästettävää summaa yksittäisistä menoista tärkeysjärjestyksessä, kunnes säästettävä summa on nolla, ja sitten lopputuloksena saadaan säästetyt menot.

Toiminta kuvattu alla kaaviossa



Vaihtoehtoinen tapa olisi esimerkiksi laskea menoja yhteen, kunnes ne ylittävät halutun säästösumman, ja nollata menot viimeiseen yhteenlaskettuun asti, joka saisi arvon (yhteenlasketut menot – säästösumma). Toteutettu tyyli on yksinkertaisempi ja tehokkaampi, koska sen ei tarvitse ottaa huomioon esim. järjestystä vaan voidaan käydä kaikki kerran läpi, jonka jälkeen tulos saadaan ulos.

Ostotapahtumat (Expense) yksittäisiksi menoiksi (Expenseobj)

Menojen luominen ostotapahtumista, on yksinkertainen, mutta oleellinen prosessi ohjelmaa. Siinä lisätään yksitellen menoiksi ostotapahtumien nimiä (kauppoja). Jos samanniminen kauppa on jo olemassa, niin täydennetään menoihin saman nimen alle vain ostotapahtuman arvot ja päivämäärät, eikä luoda samalla nimellä uutta menoa.

Toinen vaihtoehto olisi esimerkiksi luoda aina datan lisäämisen jälkeen setti ostotapahtumien nimistä, jossa jokaista nimeä on vain yksi esiintymä. Sen jälkeen käydä läpi ostotapahtumat, ja lisätä niiden arvoja setin saman nimisen kohdalle. Kuitenkin kun datan määrä kasvaa, tämä toteutus hidastuu huomattavasti, eikä ole yhtä suoraviivainen ja vähävaiheinen kuin toteutettu tyyli.

7. Tietorakenteet

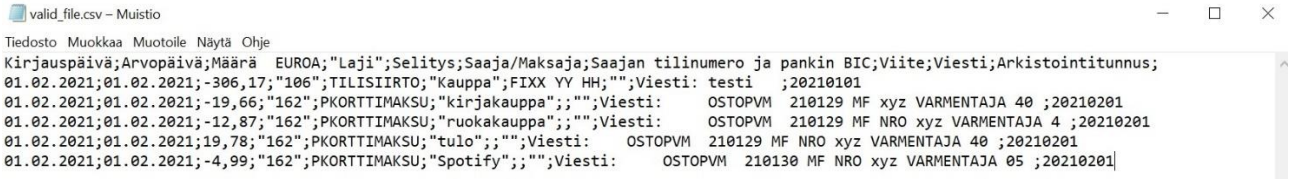
Ohjelmassa käytetään paljon listoja, olioita ja hieman myös sanakirjaa. AllExpenses luokassa käytetään paljon listoja olioiden säilyttämiseen. Olioita voi luoda monentyyppisiä, joten niissä voi säilyttää helposti monentyyppistä dataa, kuten toteutetussa ohjelmassa luodaan menoista ja kategorioista oliot, jotka sisältävät tiedon mm. nimestä, arvosta ja päivämääristä. Oliot ovat paljon laajempia kuin esimerkiksi listat ja sanakirjat ja tekevät esimerkiksi datan käyttämisestä ja löytämisestä yksinkertaisempaa.

Listat ovat ohjelman kannalta hyödyllisiä. Listoja voi muuttaa suhteellisen helposti ja ne ovat yksinkertaiseen datan säilytystarpeeseen kepeitä rakenteita. Työssä ei käytetä esimerkiksi settejä, koska monessa kohdassa listojen dataa halutaan muokata.

Sanakirjaa käytettiin pisteiden laskemisessa, kun piti järjestää meno oliot (Expenseobj), tärkeyspisteiden perusteella. Sanakirja oli selvä valinta, koska sille pystyy antamaan kaksi arvoa (key ja value), ja täten se oli hyvä säilyttämään tiedon oliosta ja pisteistä, ja se voidaan järjestää helposti pythonin tarjoamilla funktioilla.

8. Tiedostot

Ohjelma käsittelee OP:n tarjoamia csv muotoisia tilitapahtumat tiedostoja. Tiedoston rakenne on varsin pitkä, joten se on kuvattu alla olevassa kuvassa.



9. Testaus

Ohjelma läpäisee kaikki suunnitelmassa esitetyt testit. Esimerkiksi jaoteltua kuvaajaa ei saa näkyviin, jos ei ole jäljellä yhtään kategoriaa, koska käyttöliittymään on luotu toiminnot, jotka ennakoivat kyseisiä virhetilanteita mm. asettamalla napin pois käytöstä, kun tietty ehto täyttyy.

Ohjelmaa rakennettaessa, sitä testattiin mm. debuggaamalla ja print komennoilla.

Ohjelmaa on testattu käyttöliittymän kautta esimerkiksi luomalla kategoria, ja valitsemalla klikkaamalla meno ja kategoria, jolloin ”yhdistä kuluja” napin ei tulisi olla painettavissa. On tarkistettu, että kun jaotellusta näkymästä poistaa kaikki kategoriat, niin ohjelma menee automaattisesti ei jaoteltuun näkymään ja ei päästä jaoteltuun näkymään ennen, kun on vähintään yksi kategoria olemassa. Lisäksi on testattu luoda kategorioita väärällä syötteellä ja säästää väärällä syötteellä, joista kumpikaan ei onnistu, koska käyttöliittymä ennakoi ja sallii tietyissä kohdissa vain tietynlaista syötettä. On testattu myös että toiminnot ja napit toimii odotetulla tavalla.

Ohjelmaa testattiin myös yksikkötesteillä. Yksikkötestejä luotiin datan lukemiselle, jossa testattiin oikeaa syötettä, väärää ja tyhjää syötettä, joiden pitää johtaa virheilmoitukseen, osittain oikeaa syötettä, jossa syötettä luetaan ohjelmaan normaalisti, kunnes virheellinen rivi tulee vastaan ja nostetaan virhe.

Myös allExpenses luokan ominaisuuksia testattiin yksikkötesteillä. Testattiin esimerkiksi olioiden luomista, serieksen täyttämistä ja muokkaamista, kategorioiden luomista, muokkaamista ja poistamista.

10. Ohjelman tunnetut puutteet ja viat

Vaikka tehtävänannossa ei suoraan vaadittu tiedostojen virhetilanteiden käsittelemistä, niin ohjelma käsittelee vain muutaman virhetilanteen tiedostonlukemisen suhteen.

11. 3 parasta ja 3 heikointa kohtaa

3 parasta

1. Vuorovaikutteinen käyttöliittymä

Käyttöliittymää on hauska ja helppo käyttää. Kuvaaja on muokattavissa ja visuaaliset efektit ovat toimivia. Lisäksi käyttöliittymä ennakoi virhetilanteita monella tapaa esimerkiksi poistamalla painikkeita käytöstä tietyssä tilanteessa.

2. Säästötoiminto

Säästötoimintoon on luotu hyviä ominaisuuksia ja se toimii suhteellisen hyvin.

3. Virhetilanteiden käsittely

Ohjelma ennakoi ja käsittelee aika hyvin suuren osan virhetilanteista ja monet osat on rakennettu hyvin virhetilanteita ajatellen

3 heikointa

1. Ohjelmakoodin rakenteet ovat osittain suhteellisen pitkiä ja epäselkeitä varsinkin WidgetWin luokassa. (Kuitenkin vain osa, ja osittain myös johtuu siitä, että on vuorovaikutteinen käyttöliittymä, joka vaatii enemmän koodia. Silti voisi olla pienemmissä paloissa)
2. Ohjelmakoodin laajennettavuudessa voisi olla joissain kohdissa parantamisen varaa, esim. kirjoittamalla yleisempää koodia.
3. Ohjelma ei ehkä neuvo vaiheitaan tarpeeksi joillekin käyttäjille. Voisi lisätä vinkkejä ohjelman toimintaan.

12. Poikkeamat suunnitelmasta

Suunnitelman mukaan luotiin paljon luokkia ja niiden metodeita, mutta rakenteesta poikettiin hieman, ehkä jakamalla pienempiin osiin. Suunnittelutilanteessa ei nähty kaikkia pienimpiä osia mistä ohjelma koostuu. Toteutusjärjestys oli muuten suhteellisen samanlainen paitsi käyttöliittymä luotiin ensimmäiseksi, koska PyQt oli vielä tuntematon, eikä sitä kannattanut jättää viimeiseksi.

13. Toteutunut työjärjestys ja aikataulu

Ohjelmaan luotiin ensin käyttöliittymää (13.3.2021). Sen jälkeen luotiin Dataread luokkaa (15.3.2021). Jonka jälkeen loput allExpenses luokka ja meno, kategoria, ostotapahtuma luokat (24.3.2021). Sen jälkeen luotiin lisää ominaisuuksia ja parannuksia, kunnes luotiin Series luokka (18.4.2021) ja yksikkötestit luotiin (22.4.2021). Suunnitelmasta poikettiin ainakin siinä, että aloitettiin käyttöliittymästä, koska ei tunnettu PyQt:n toimintaa vielä silloin.

14. Arvio lopputuloksesta

Ohjelma on kokonaisvaltaisesti OK. Hyviä puolia ovat käyttöliittymän vuorovaikutteisuus ja muokattavuus, säästötoiminnon toimivuus ja virhetilanteiden huomioiminen. Ohjelmassa voisi olla tiedostoille enemmän virhetilannekäsittelyitä. Ohjelman rakenne on suhteellisen hyvä, mutta paikoittain olisi parantamista esim. selkeyteen ja paloitteluun. Samoin ohjelman uudelleenkäytettävyys vaihtelee hieman osien välillä, ja sitä ehkä voisi joissain osissa parantaa (toisaalta uudelleenkäytettävyys vaihtelee luonnollisestikin erilaisten ominaisuuksien välillä). Joissain kohdissa voisi myös opetella kirjoittamaan ”yleisempää” koodia.

15. Viitteet

Kurssimateriaali: https://plus.cs.aalto.fi/y2/2021

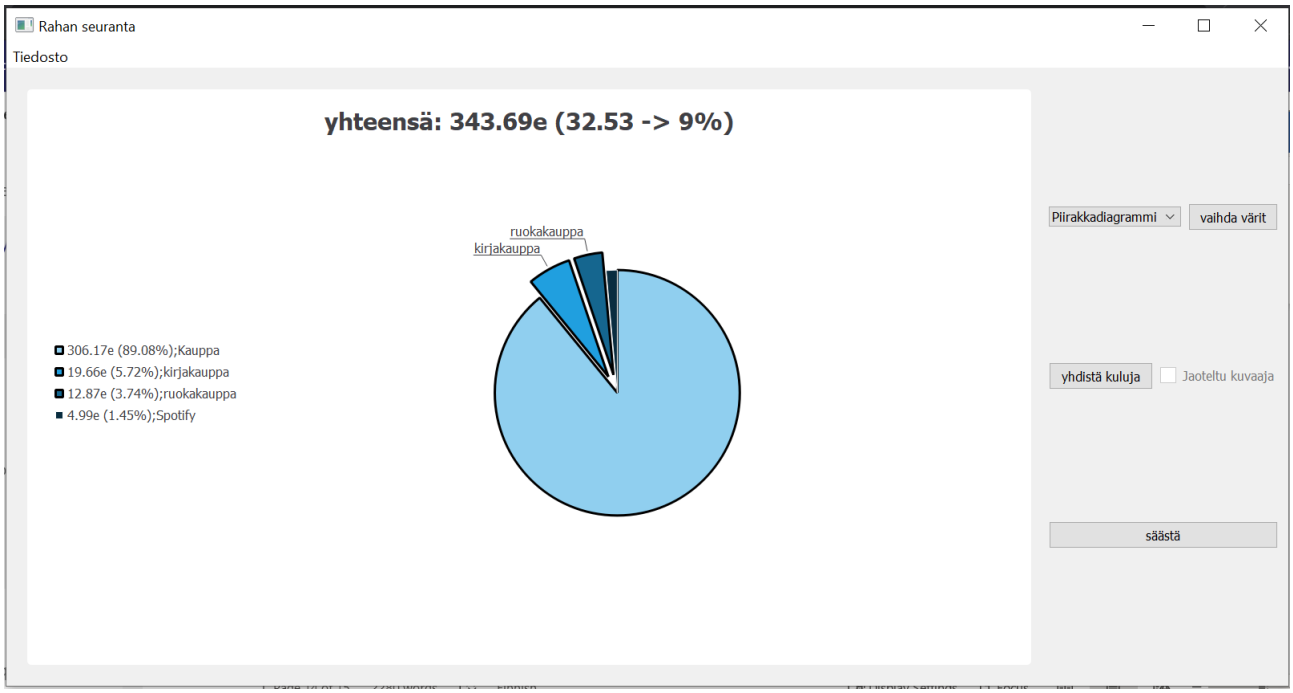
pyqt dokumentit: https://doc.qt.io/qt-5

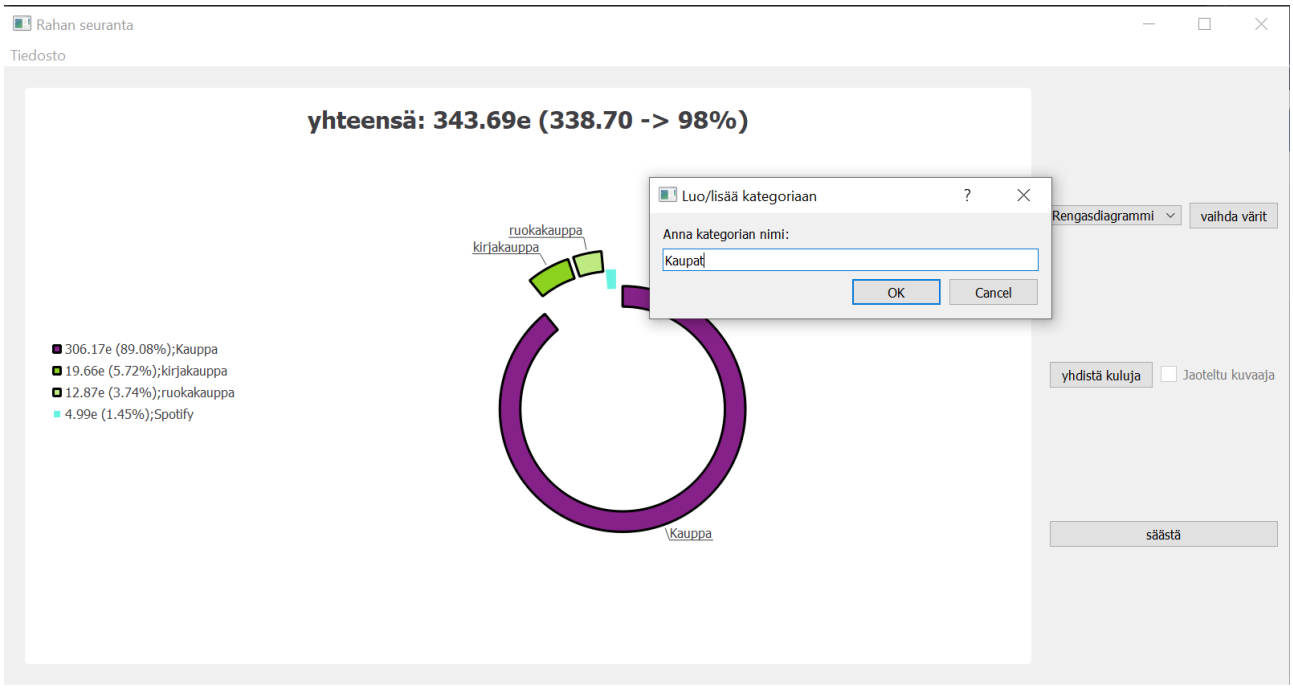
https://stackoverflow.com/

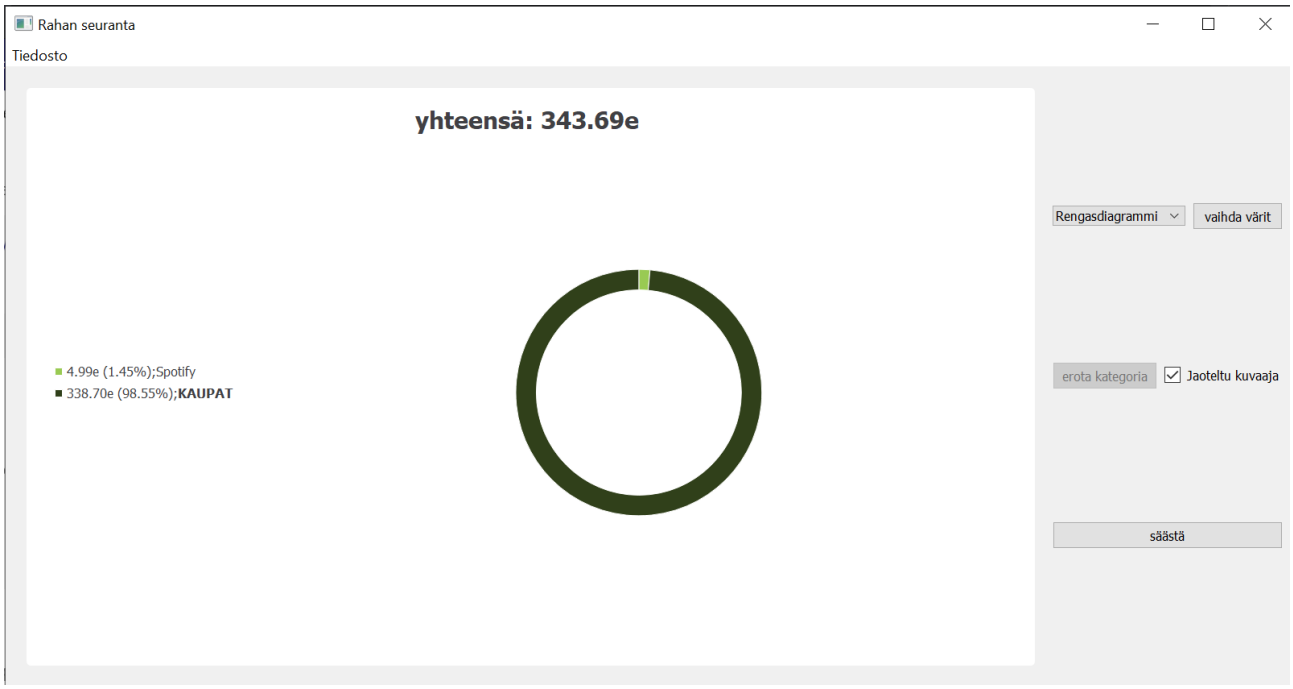
https://docs.python.org/3

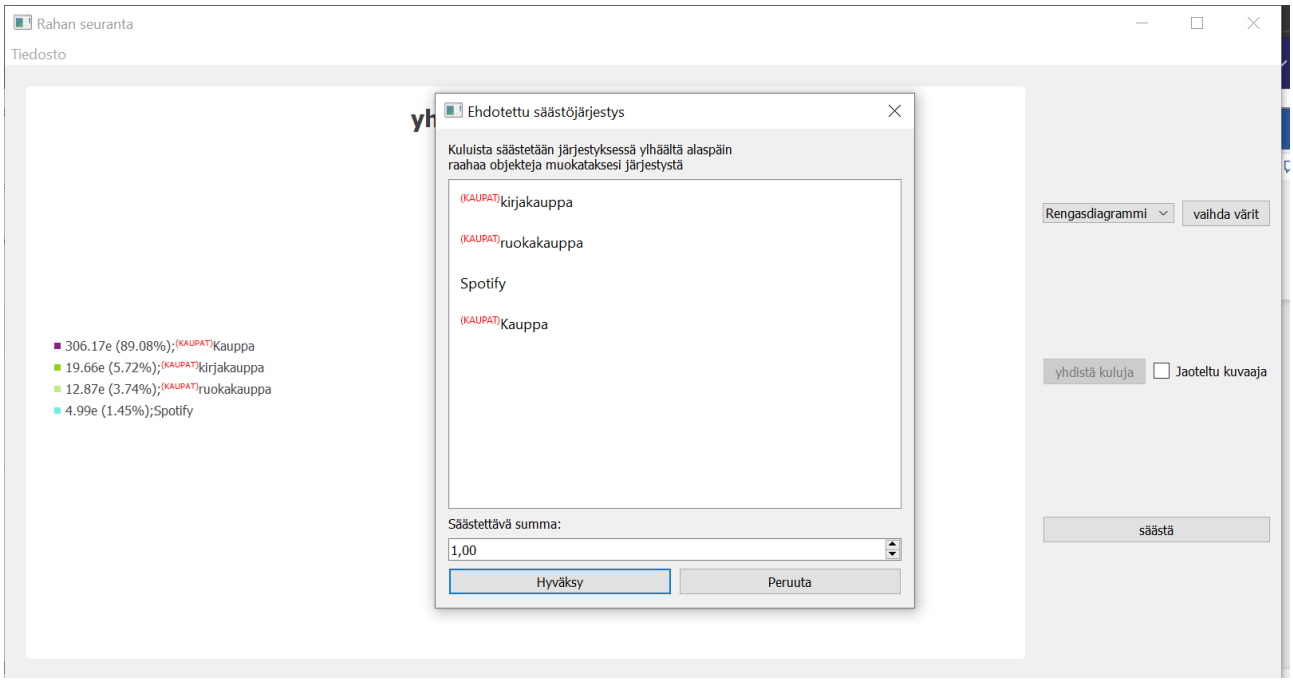
16. Liitteet **(Kuvia käyttöliittymän ajosta)**

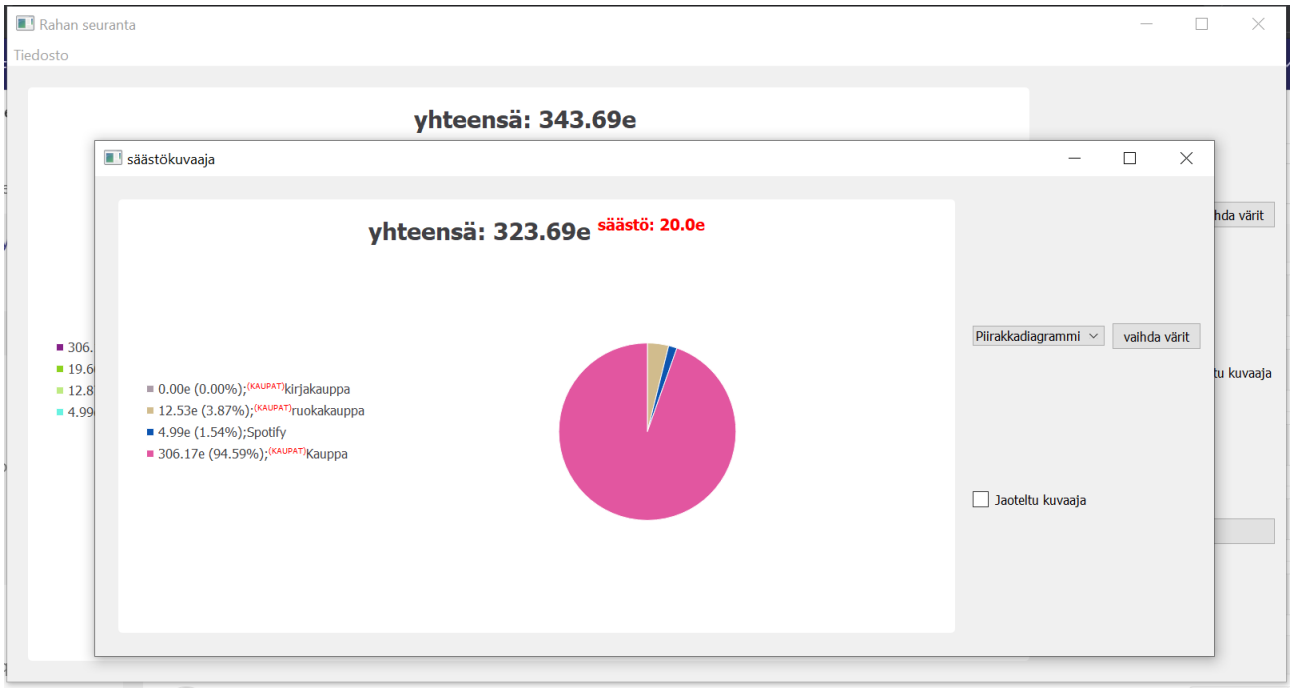

Rahan seuranta
Tiedosto
Lisää tiedosto
Piirakkadiagrammi
vaihda värit
yhdistä kuluja
Jaoteltu kuvaaja
säästä


Rahan seuranta
Tiedosto
yhteensä: 343.69e (32.53 -> 9%)
ruokakauppa
kirjakauppa
306.17e (89.08%);Kauppa
19.66e (5.72%);kirjakauppa
12.87e (3.74%);ruokakauppa
4.99e (1.45%);Spotify
Piirakkadiagrammi
vaihda värit
yhdistä kuluja
Jaoteltu kuvaaja
säästä


Rahan seuranta
Tiedosto
yhteensä: 343.69e (338.70 -> 98%)
Luo/lisää kategoriaan
Anna kategorian nimi:
Kaupat
OK
Cancel
ruokakauppa
kirjakauppa
Kauppa
306.17e (89.08%);Kauppa
19.66e (5.72%);kirjakauppa
12.87e (3.74%);ruokakauppa
4.99e (1.45%);Spotify
Rengasdiagrammi
vaihda värit
yhdistä kuluja
Jaoteltu kuvaaja
säästä


Rahan seuranta
Tiedosto
yhteensä: 343.69e
4.99e (1.45%);Spotify
338.70e (98.55%);KAUPAT
Rengasdiagrammi
vaihda värit
erota kategoria
Jaoteltu kuvaaja
säästä


Rahan seuranta
Tiedosto
Ehdotettu säästöjärjestys
Kuluista säästetään järjestyksessä ylhäältä alaspäin
raahaa objekteja muokataksesi järjestystä
(KAUPAT)kirjakauppa
(KAUPAT)ruokakauppa
Spotify
(KAUPAT)Kauppa
Säästettävä summa:
1,00
Hyväksy
Peruuta
306.17e (89.08%);(KAUPAT)Kauppa
19.66e (5.72%);(KAUPAT)kirjakauppa
12.87e (3.74%);(KAUPAT)ruokakauppa
4.99e (1.45%);Spotify
Rengasdiagrammi
vaihda värit
yhdistä kuluja
Jaoteltu kuvaaja
säästä


Rahan seuranta
Tiedosto
yhteensä: 343.69e
säästökuvaaja
yhteensä: 323.69e säästö: 20.0e
0.00e (0.00%);(KAUPAT)kirjakauppa
12.53e (3.87%);(KAUPAT)ruokakauppa
4.99e (1.54%);Spotify
306.17e (94.59%);(KAUPAT)Kauppa
Piirakkadiagrammi
vaihda värit
Jaoteltu kuvaaja