Projektityön dokumentti

Henkilötiedot:

Rahan seuranta; Teerialho Veikko 735045, Kemiantekniikka, 060521

Yleiskuvaus:

Ohjelma, joka jaottelee käyttäjän kulutusdataa paremmin kuvaamaan muotoon

Käyttöohje

Aja main.py

Ohjelma kysyy käsiteltävän tiedoston polun, anna se sille

Ohjelma kysyy haluatko käyttää kovakoodattua enneltmäärättyjä ryhmiä luokittelussa.

Luokittele jäljellä olevat ohjelman avulla

Ohjelma kertoo kulutuksen ja antaa mahdollisuudet manuaalisesti muokata dataa ja kokeilla säästöohjelmaa

Manuaalimuokkaus:

Anna yläryhmä, jota haluat muokata ja anna summa(snt) jolla sitä haluat muokata

Säästöohjelma:

Anna välttämättämyysryhmät (necessity) ja matalan prioriteetin ryhmät sekä säästötavoite. Ohjelma kertoo säästöjakauman ja näyttää uuden tilanteesi.

Kirjastot:

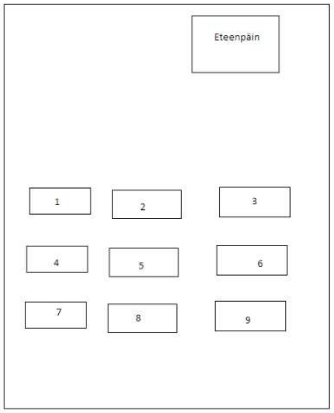
Pyqt5 ja sys

random

Ei ole vaadittu, mutta tekee ohjelmasta ulkonällisesti selkeämmän. Käytetään luomaan RBG-väreillle arvoja. Eri värien käyttö helpottaa kuvaajan tulkintaa ja randomilla on mahdollista luoda funktio värien luonnille.

from functools import partial

partial on korjaamaan nimenomainen ongelma, jossa painekkeita luodaan käyttämällä syötteenä string-listaa. Partialin avulla funktiot saadaan kohdistumaan jokaiseen erikseen ja mahdollistaa lisätoiminnot perustuen aiempaan dataan. Muilla rakenteilla komennot osuivat viimeiseen painikkeeseen.



Ohjelman rakenne:

QT-olio

- näkyvät osat näkymä kerrallaan
- QT:n dataa tarvitsevat funktiot

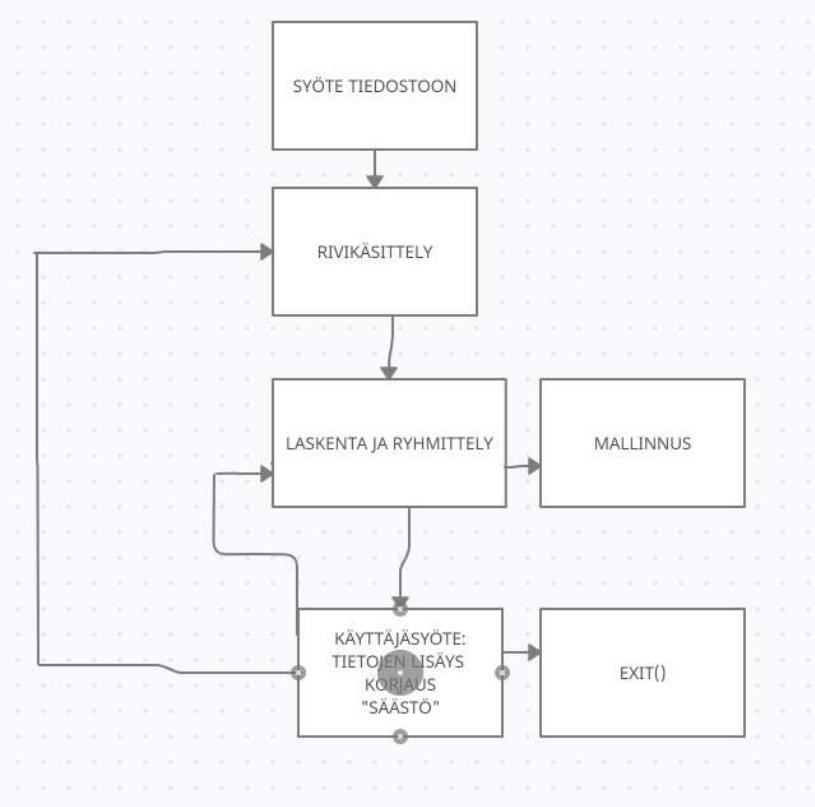
Datasäilö-olio

- Käytetään tapana siirtää dataa eri osien välillä

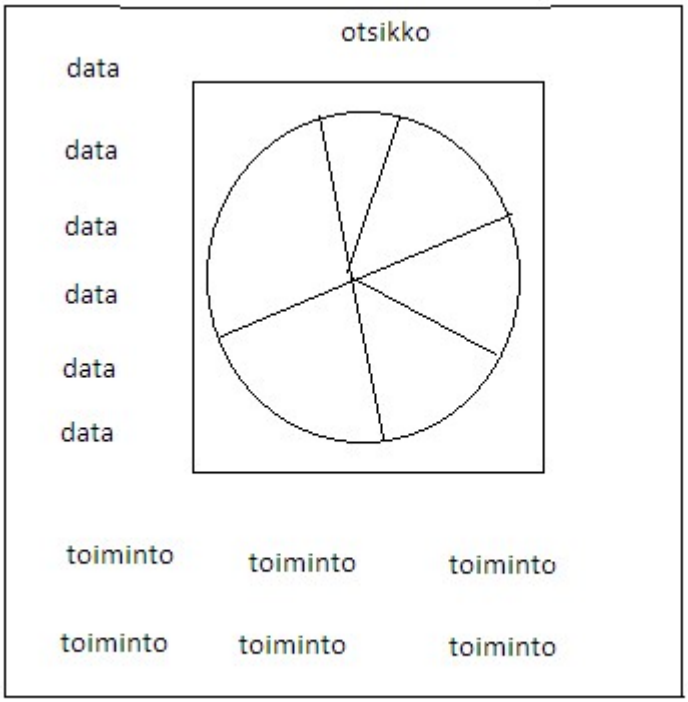
Tavalliset eli ei-QT-funktiot

- Käytännössä täällä luetaan ja muokataan sitä csv-dataa

Main funktio luo QT-olion, joka siirtyy vaiheeseen yksi. Tämä olio kutsuu QT:n ulkopuolisia osia saadakseen tietonsa. Käyttöliittymä on suurelta osin hyvin yksinkertaista QT-olioden luontia, painikkeita ja kirjoituskenttiä. Monimutkaisin osio on itse kuvaaja ja sitä käsittelemässä on funktio paintevent. Tämä on säädetty niin että se aktivoituu vain kun vaihe on oikea. Paintevent vaikuttaa olevan selkeästi raskaampi ajaa kuin muut QT:n osat saati tekstipohjainen koodi. Tämän vuoksi sitä on hyvä rajoittaa.



Ohjelman kulku



Kaaviovaiheen yksinkertaistettu versio

Algorytmit:

Ohjelma lukee pankin autogeneroimia csv-tiedostoja. Esimerkkirivi näyttää tältä:

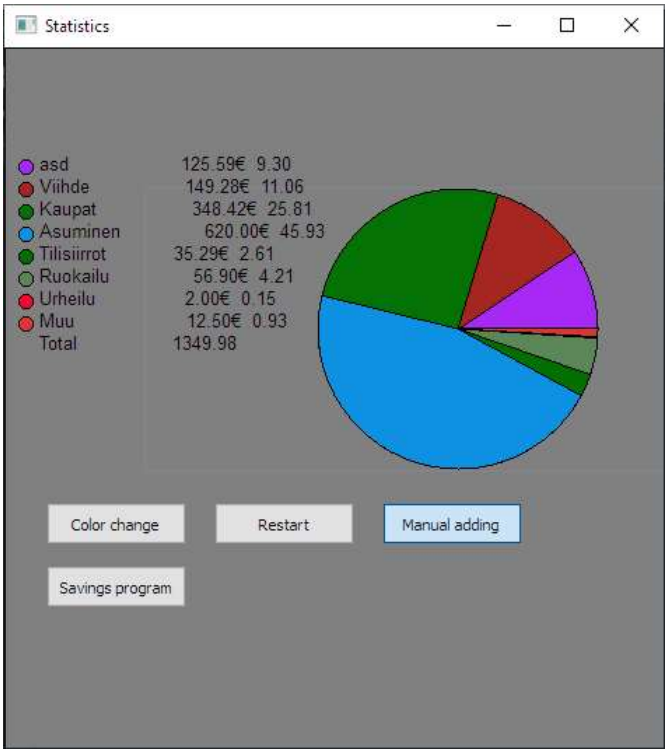
04.02.2021;04.02.2021;-580;"162";PKORTTIMAKSU;"Teknologforeningen ESPOO";;"";Viesti: 492057******8249 OSTOPVM 210203 MF NRO 74920541035165088157602 VARMENTAJA 400 ;20210204/5EQEO1/037630

Se mitä ohjelma tästä lukee, on -580 ja "Teknologforeningen ESPOO". Sen jälkeen ohjelma ynnää kaikki "Teknologforeningen ESPOO":t sitä mukaa kun ne esiintyvät csv:ssä.

Tämän jälkeen spagulle pitää keksiä ryhmiä, johon se kuuluu ohjelma kysyy käyttäjältä ryhmää. Käyttäjä voi todeta että se kuuluu esimerkiksi ryhmään opiskelu tai terveys tai ruokailu sen perusteella mikä kulu on kyseessä ja mihin toimintaan se liittyy. Samaan ryhmään voisi kuulua esimerkiksi tapahtuma

27.01.2021;27.01.2021;-2,"60;""162"";PKORTTIMAKSU;""YLVA Palvelut Oy HELSINGIN YLI"";;""""";Viesti: 492057******8249 OSTOPVM 210126 MF NRO 74920541027443942197860 VARMENTAJA 400 ;20210127/5EQEO1/004072"

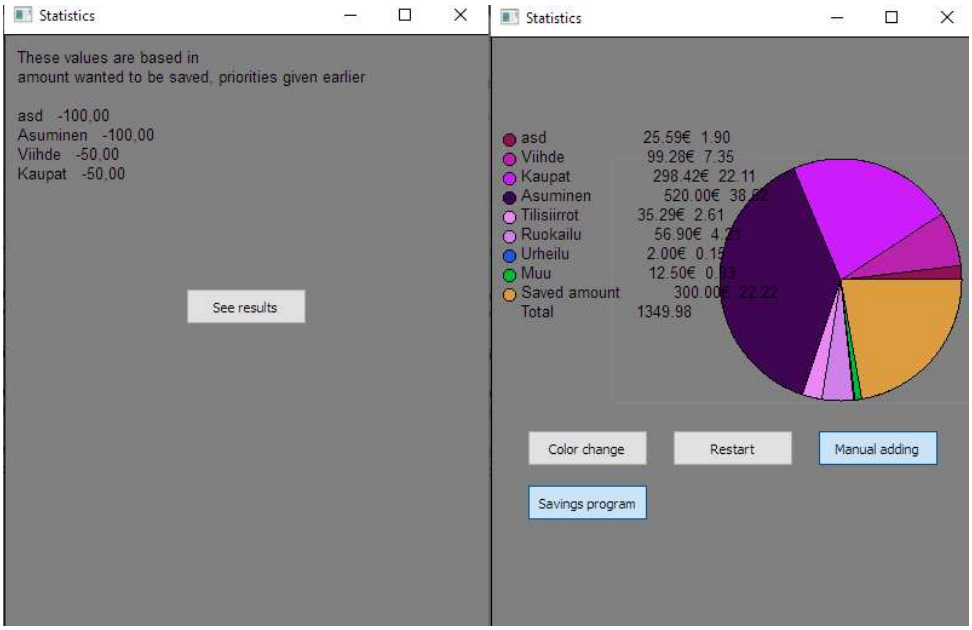
ja ohjelma laskee että Yläryhmä ruokailu kulutus on 580 + 260 = 740. Sitten piirakkakuvaajassa on dataa perusteella ruokailu 740.Kun luokittelu on valmis niin ohjelma siirtyy tekemään piirakkakaaviota (statistiikka). Koska joissakin funktiossa, joiden jälkeen palataan statistiikkaan, muokataan tietoja, on tämän vuoksi statistiikkaan siirryttäessä haettava tiedot uudelleen.



Tämä on kuvaaja on nyt se mitä ohjelma hakee ja mihin se aina palaa. Kun siirrytään muokkaamaan dataa, manual adding on ns perusversio ja savings program on hienompi versio. Eli manual adding kysyy ryhmää (vaikka Asuminen) ja siihen lisättävää summaa (300). Tämän jälkeen palataan takaisin ja Asuminen on kasvanut 300:lla.

Savings program kysyy käyttäjältä ensin pakolliset kulut ja sen jälkeen vähäisen prioriteetin kulut sekä säästötavoitteen. Ohjelma kertoo säästöjakauman ja näyttää miltä kuvaaja

näyttäisi näillä säästöillä ja mikä osuus tästä säästö olisi. Savings program käyttää aivan samoja rakenteita kuin manual adding, mutta siinä on hieman enemmän ”älyä” mukana.



Kuvassa näkyy miten suhde on 1: 2 jos ryhmä on tai ei ole prioriteetti.

Tietorakenteet:

Ohjelma käyttää tietorakenteina listoja, sanakirjoja ja muuttujia. Kaikki on pythonin omia ja yksinkertaisia rakenteita.

Tiedostot:

Ohjelma käyttää autogeneroituja csv-tiedostoja. Tämän lisäksi on kolme .py-tiedostoa, jossa yhdessä on itse ohjelma, toisessa main ja kolmannessa testit.

Testit:

Ohjelma sisältää ilman GUI:ta olevia osia ja sen kanssa olevia osia.

Ilman GUI: yksikkötestit mallia assertEqual(funktio, paluuarvo). Hyvin yksinkertaiset ja käytännössä kokeilee ennekkotapauksella.

GUI:lla: Testiohjelma luo vähän erilaisen QT-olion. Rakenteet on samat, mutta tämä klikkailee ohjelmaa eteenpäin ja käy ns normi käytön läpi. Oliossa toteutettu mallilla

if self.test_run:

self.button1.click()

ja button1 jatkaa seuraavaan vaiheeseen. Tää tuntui ihan hirveältä rakenteelta (ks. 3 parasta ja 3 heikointa kohtaa), mutta mahdollisti testien rakennuksen niin että se huomioi myös QT:n osia. Itse testit ovat hyvin samanlaisia kuin ilman guita.

Ohjelman tunnetut puutteet ja viat:

- QT-olion alussa luodaan osia, joita ei vielä tarvitse.

- Testikattavuus on vajaa (alle 100 %)
- Toistuvat hide() käskyt
- def window_reset:ille on todennäköisesti olemassa parempi tapa tehdä asia
- QT-painikkeiden ja qLineEditien käskyt eivät ole johdonmukaisessa järjestyksessä
- Funktiot eivät ole fiksusti jaoteltuna, QT:n steppejä lukuunottamatta.
- if debug:- lauseet tekevät ei debug-ajoista epäoptimaalisia
- whichbtn() sisältö on kauheaa spagettia, mutta tekee mitä pitää
- self.savings_buttonlist = [ ]-alkuinen koodipätkä esiintyy useampaan kertaan
- keyPressevent ei sisällä enter ja numpad-enter nappeja
- Stats:in kahden askeleen tallennus on epäoptimaalinen, esim keywordlist ja keywordlistsave
- restartia ajamalla saa välillä Stats:in tietoja sekaisin
- Savings_programia ei voi ajaa takaisin
- osa muuttujanimistä toistuu, osa on epäjohdonmukaisia (”oppressor”)
- GUI:n sisäiset testit käyttävät ylimääräistä if self.test_run-rakennetta simuloidakseen ajoa

3 parasta ja 3 heikointa kohtaa:

Parhaat:

def start_saving_program_step1(self) rivit 557 – 565

Täällä on se partial, mistä oli jo maininta (ks ulkoiset kirjastot). Rakenne syö listan ja tuottaa jokaiselle alkiolle oman napin. Tämä oli luultavasti koko projektin vaikein osio, eikä ollut edes pakollinen. Tekee mielestäni ohjelmasta selkeämmän käyttää, vähentää kirjoitusviheistä johtuvia ongelmia ja on mun mielestä myös käyttömukavuuden kannalta parempi. Jos yläluokkia olisi paljon niin hyödyt vähenisi, mutta yläluokkien tarkoitus on että niitä ei ole paljoa.

def manual_adding_user_input_validation(self) rivit 762 – 791

Käyttäjän syötteen validointia. Tämän kun olisi tehnyt heti, niin se olisi säästänät monet kaatumiset. Ja epävirallista palautetta kysyessä kuulemma ”Random aasialainen merkki integer-kentässä kaataa sun koodin, tee jotain”, niin palautteeseen on reagoitu.

def changeColor_infowindow(self) rivit 829 – 844

QColor antaa mahdollisuudet RGB-väreillä säätää arvoilla 0 – 255. Yläryhmien määrää ei tiedetä ja jotenkin pitäisi erottaa piirakanpalat toisistaan. Mahdollisuus 1: Kovakoodaa kourallinen värejä, käytä niitä uudestaan.

Mahdollisuus 2: Anna sen päättää väreistä, joka niitä katsoo. Ohjelma luo eri värikokonaisuuksia randomilla ja käyttäjä voi ottaa uusia värikokonaisuuksia niin monta kertaa kunnes on tyytyväinen.

Pahimmat:

def keyPressEvent(self , e) 915-920

Tällä oli 2 tarkoitusta, jotka sen piti toteuttaa. 1: Esc heittää sut ulos ja 2: Enter vie eteenpäin. No Esc heittää sut ulos ja enterille saatiin toiminnallisuus ”kaataa ohjelman”.

class Stats(): rivit 924 – 941

Staattinen tiedonsäilytys on vähän huono ja nykyinen malli on lisäksi julkinen. Toisaalta kun projekti on noin 1 TLOC ja käsiteltävän tiedon määrä on kohtalaisen vähäinen niin tän ongelmat eivät juuri näy.

Stats.reset() rivit 942 – 957

Kun ohjelmaa testaa manuaalisesti, niin välillä tää toimii osin ja välillä ei. Tai ei ainakaan tee kaikkea sitä mitä pitäisi.

Bonus:

```
if self.test_run:
    do()
```

Tämä voisi periaatteessa olla kumpaa vain. Hirveää spagettia ja 4/5 epäluotettava. Toisaalta mahdollistaa GUI:n sisällä oleville testien kirjoittamisen. Jos osaisin sanoa tällä hyvän vaihtoehdon niin se olisi koodissa.

Poikkeamat suunnitelmasta:

Ohjelmaan luottiin tylsä harmaa taustaväri korjaamaan ongelmaa kuvanrajauksessa.

Säästökohteiden luokittelun periaate muuttui lennossa osa automaattisesta manuaaliseksi.

Toteutunut työjärjestys ja aikataulu:

Toteutusjärjestys oli sama kuin suunniteltu, mutta jälkikäteen se oli huono. Ohjelman teko olisi sujunut jouhevammin kun olisi aloittanut GUI:sta ja täydentänyt engineä tarpeen mukaan. Nyt engine määräsi GUI:n muodostumista ja jos yksinkertainen määrää monimutkaisempaa niin menee väärin päin.

Aikataulu suurinpiirtein piti. Vähän venyi lopussa, muttei kriittisesti.

Arvio lopputuloksesta:

Ohjelma jotenkin ei ole kovin vakaa ja tietorakenteet on tehty mallilla ”No tää toimii, katsotaan uusiksi jos tulee ongelmia.” ja osa tiedoista päivittyy epäluotettavasti. Koodissa on paljon ns spagettiratkaisuja, mm if-lauseketjuja. Sekä QT:n että enginen osalta osa koodista tuntuu ns legacy-koodilta, johon ei oikein uskalla koskea koska se toimii edes jotenkin tällä hetkellä.

Ohjelma tekee mitä piti ja sitä voisi ajaa ilman consolea ihan hyvin. GUI:n sisäisestä testauksesta ei oikein ollut kurssilla juuri mitään ja testaustakin käytiin mielestäni vähän kevyesti läpi, mutta silti sekin onnistui jotenkin lopulta. Ei tämä sellaisessa kunnossa ole, että voisi myydä, mutta ei sitä ihan häpeäkään.

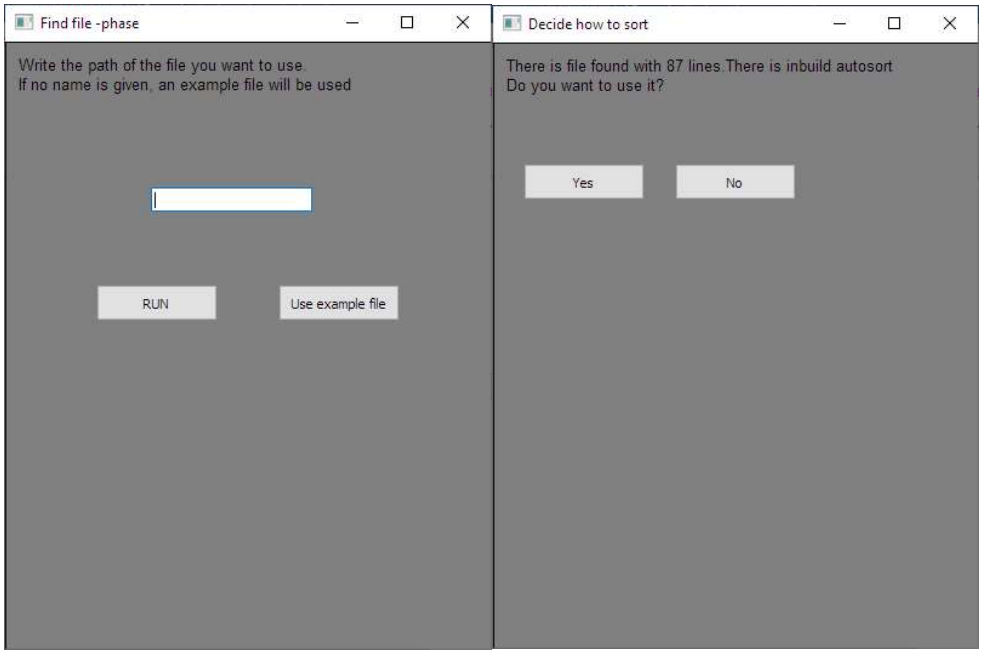
Viitteet

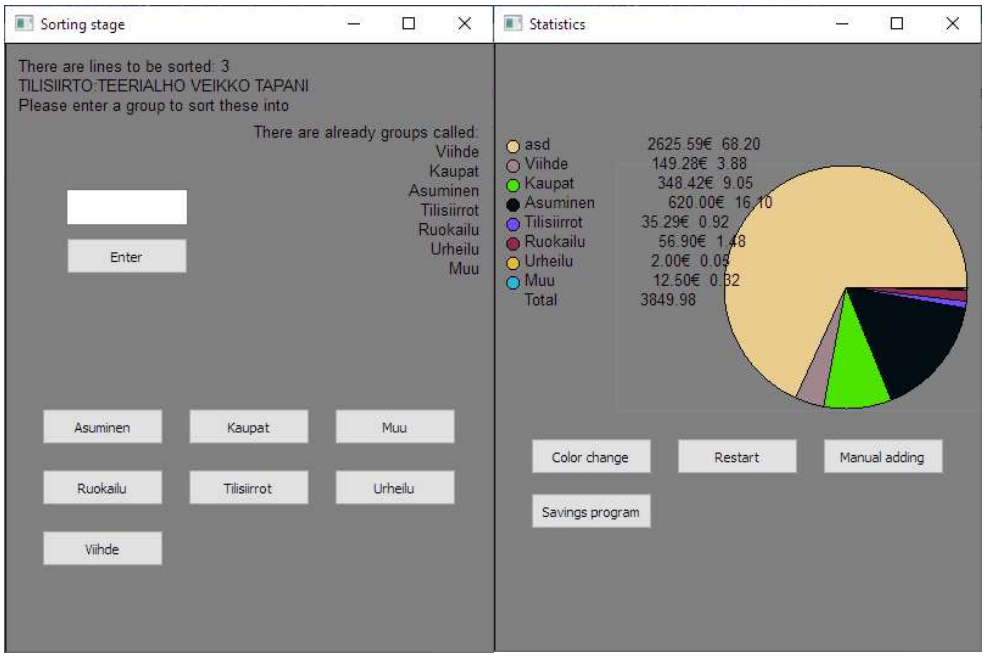
Kurssimateriaali Y1 ja Y2, saatavilla plus.cs.aalto.fi

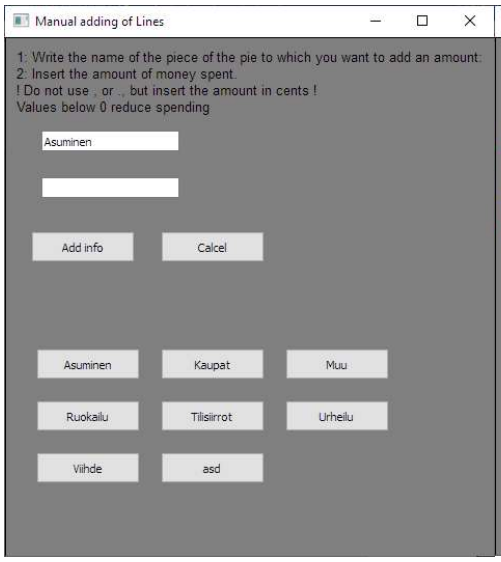
QT Documentation, saatavilla https://doc.qt.io/qt-5/qpushbutton.html

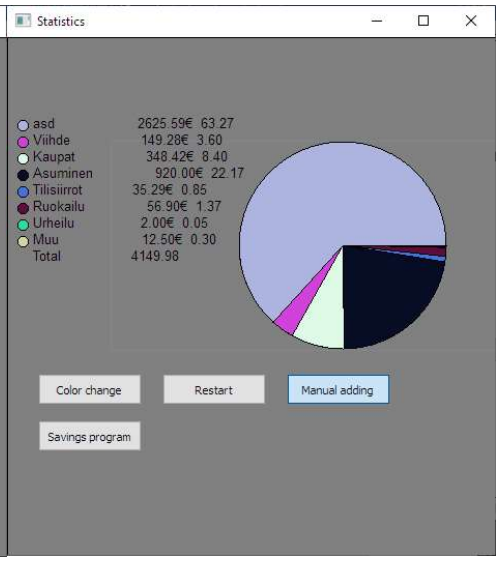
StackOverflow: How to pass arguments to functions by the click of button in PyQt? saatavilla https://stackoverflow.com/questions/6784084/how-to-pass-arguments-to-functions-by-the-click-of-button-in-pyqt/42945033

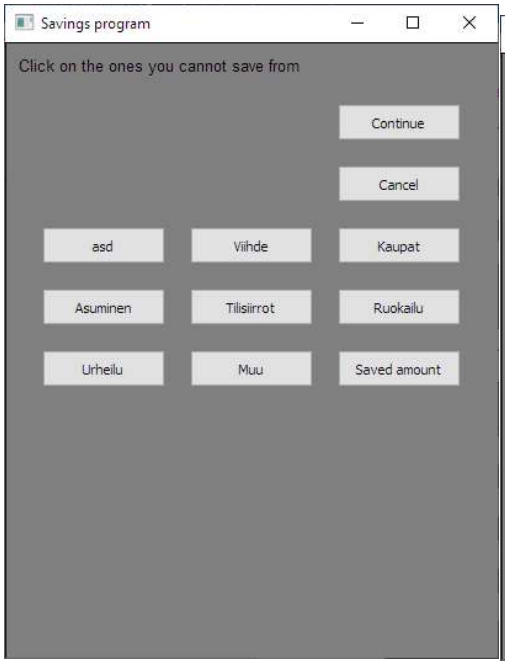
Liitteet

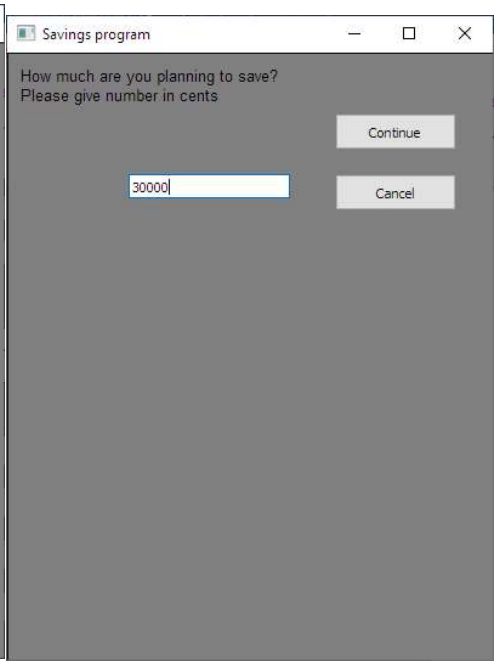


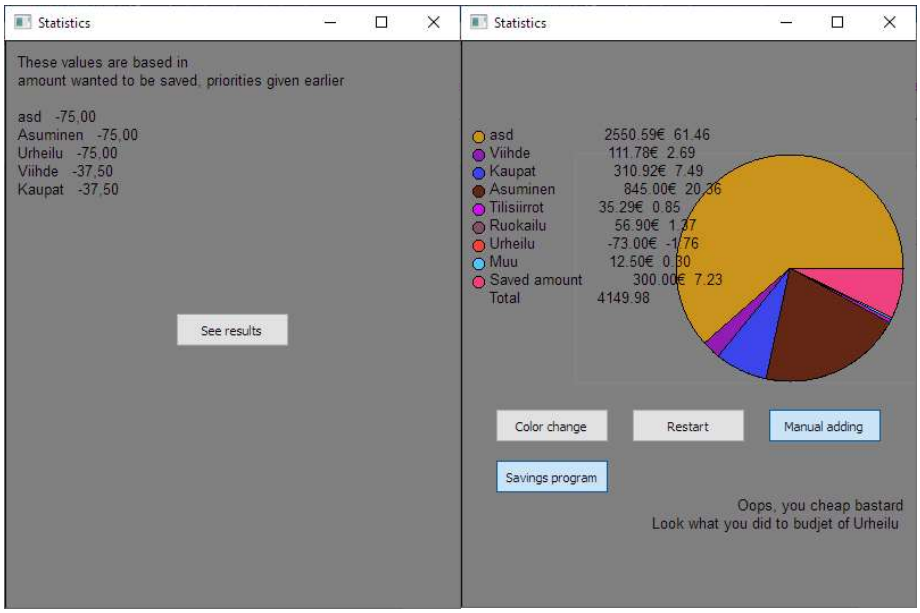



Manual adding of Lines
1: Write the name of the piece of the pie to which you want to add an amount.
2: Insert the amount of money spent.
! Do not use , or . but insert the amount in cents !
Values below 0 reduce spending
Asuminen
Add info
Calcel
Asuminen
Kaupat
Muu
Ruokailu
Tilisiirrot
Urheilu
Viihde
asd


Statistics
asd 2625.59€ 63.27
Viihde 149.28€ 3.60
Kaupat 348.42€ 8.40
Asuminen 920.00€ 22.17
Tilisiirrot 35.29€ 0.85
Ruokailu 56.90€ 1.37
Urheilu 2.00€ 0.05
Muu 12.50€ 0.30
Total 4149.98
Color change
Restart
Manual adding
Savings program


Savings program
Click on the ones you cannot save from
Continue
Cancel
asd
Viihde
Kaupat
Asuminen
Tilisiirrot
Ruokailu
Urheilu
Muu
Saved amount


Savings program
How much are you planning to save?
Please give number in cents
Continue
30000
Cancel


Statistics
These values are based in
amount wanted to be saved, priorities given earlier
asd -75,00
Asuminen -75,00
Urheilu -75,00
Viihde -37,50
Kaupat -37,50
See results
Statistics
asd 2550.59€ 61.46
Viihde 111.78€ 2.69
Kaupat 310.92€ 7.49
Asuminen 845.00€ 20.36
Tilisiirrot 35.29€ 0.85
Ruokailu 56.90€ 1.37
Urheilu -73.00€ -1.76
Muu 12.50€ 0.30
Saved amount 300.00€ 7.23
Total 4149.98
Color change
Restart
Manual adding
Savings program
Oops, you cheap bastard
Look what you did to budjet of Urheilu