

# Y2-projektin dokumentti: Kalori- ja ravintoainelaskuri

Rami Lahtinen, 916929, Bioinformaatioteknologia, kolmas vuosikurssi, 8.5.2024

## 1. Yleinen kuvaus

Tämän ohjelman avulla käyttäjä voi seurata ravintoaineiden ja energian saantia. Se arvioi ensin käyttäjän kysymällä tältä lyhyen sarjan kysymyksiä ja laskemalla käyttäjälle eräänlaisen ravintoainetavoitteen. Sitten käyttäjä siirtyy seurantaikkunaan, jossa tavoite näytetään sekä tekstinä että graafisesti. Käyttäjä voi sitten syöttää syötettyjä ruokia, jotka päivittävät näyttöä. Olen saanut projektin valmiiksi ”keskitasolla” suunnitelmien mukaisesti: kaikkien ”helpon” tason vaatimusten lisäksi ohjelmassa on toimiva graafinen käyttöliittymä ja testit joillekin ohjelman osille.

## 2. Käyttöohjeet

Ohjelma käynnistetään tiedostosta ”app.py” (entinen ”main.py”) painamalla painiketta run mistä tahansa IDE:stä. Käyttäjälle esitetään tämän jälkeen joukko kysymyksiä, minkä jälkeen häntä kehoitetaan painamaan painiketta ”Submit”. Vaihtoehtoisesti käyttäjä voi ladata tavoitteensa tekstitiedostosta, jolloin hän pääsee myös seurantaikkunaan. Seurantaikkunassa käyttäjä voi halutessaan kirjautua sisään ja viedä tavoitteensa tekstitiedostoon.

Nutrient ...

What is your name?

Rami

What is your gender? (male/female)

male

How old are you?

22

How much do you weigh in kilograms?

100

How tall are you in centimeters?

182

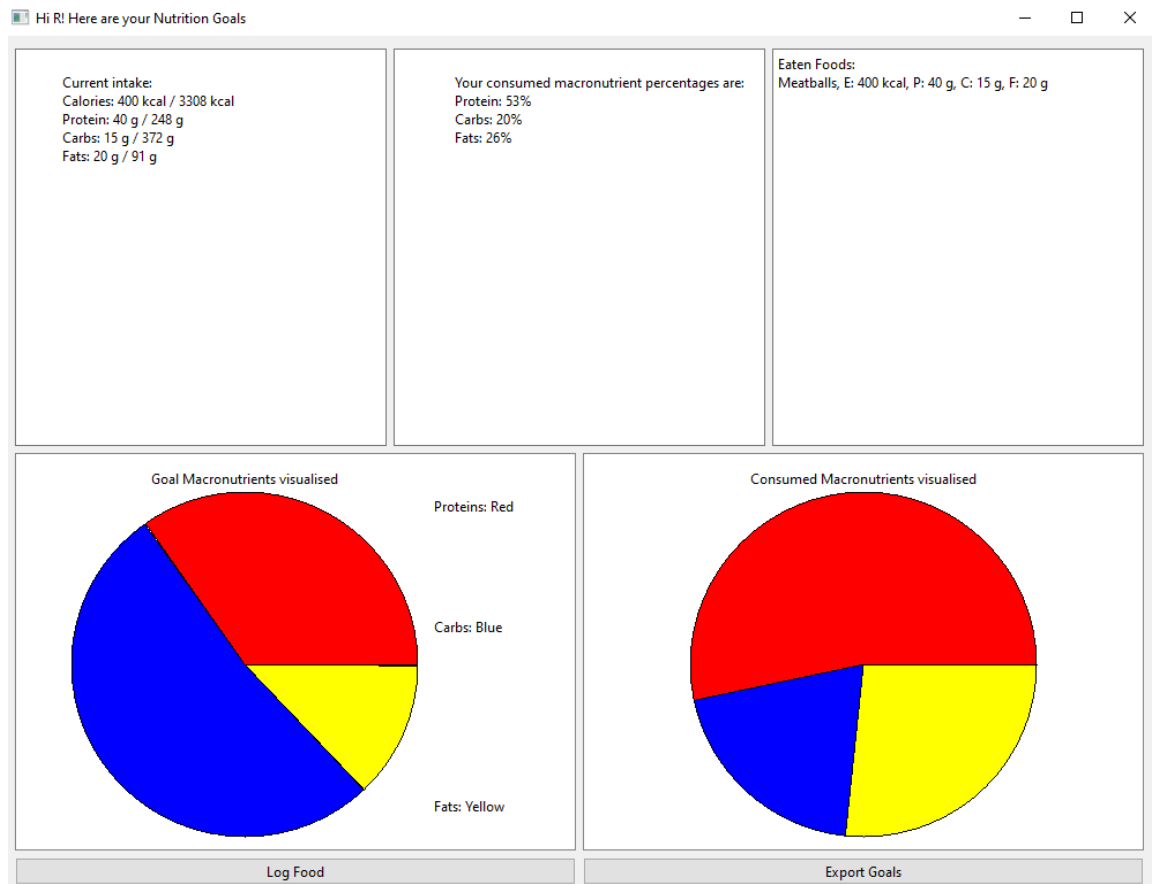
What type of person are you?

Student

Submit

Load Goals from file

Jos käyttäjä painaa painiketta ”Load goals from file”, häneltä kysytään käyttäjänimi ja tiedostopolku tekstitiedostoihin, jotka sisältävät hänen ravintoainetavoitteensa.



Huomio: ennen ensimmäisen ruoka-aineen syöttämistä keskimmäisen widgetin teksti näyttää sen sijaan ”No data yet, input something!” ja oikeanpuoleinen piirakkakaavio ei näy.

### 3. Ulkoiset kirjastot

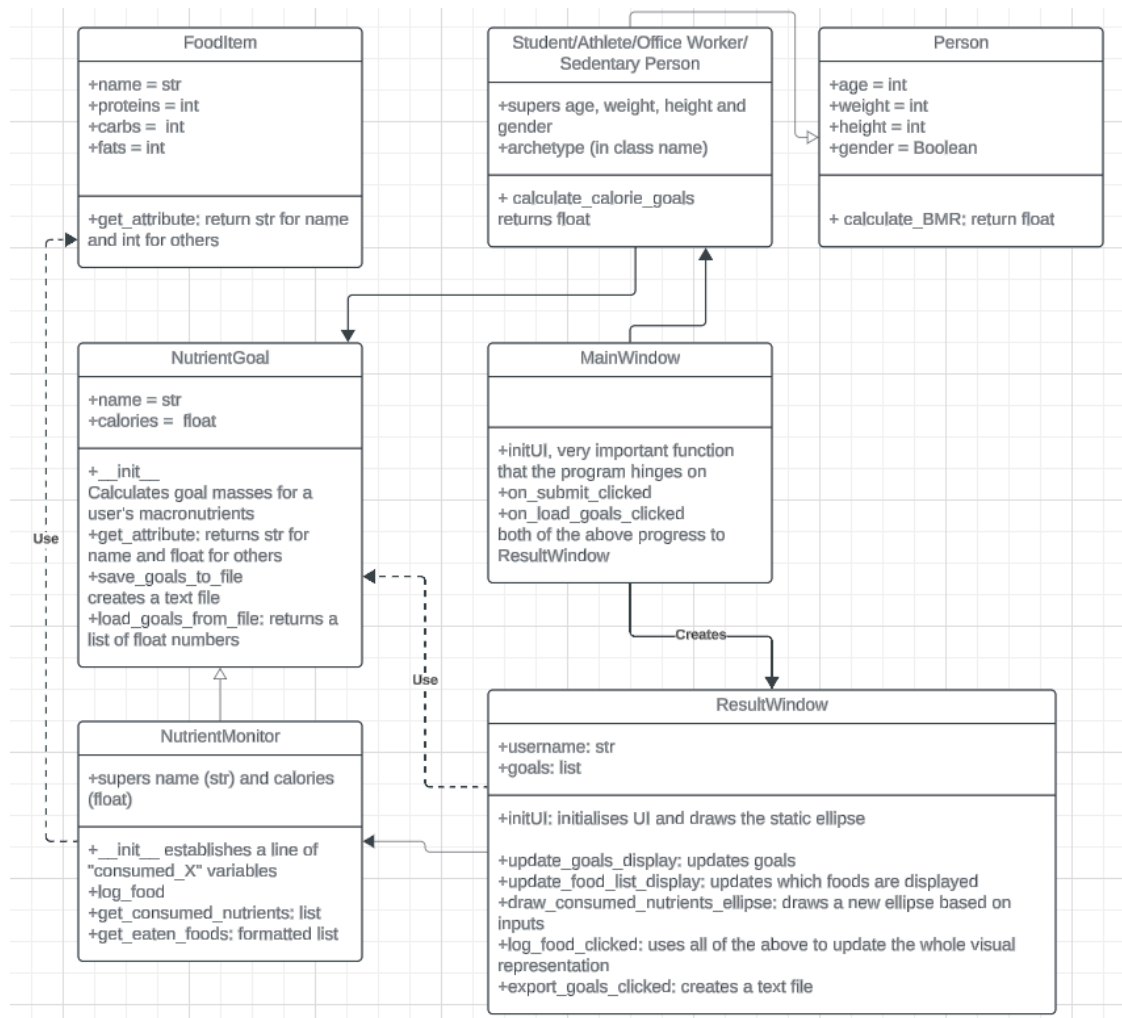
Tässä ohjelmassa käytetään Unittestin lisäksi vain PyQt6:ta ja sen alikirjastoja. Näistä merkittävimmät ovat QtCore, QtWidgets ja QtGui.

### 4. Ohjelman rakenne

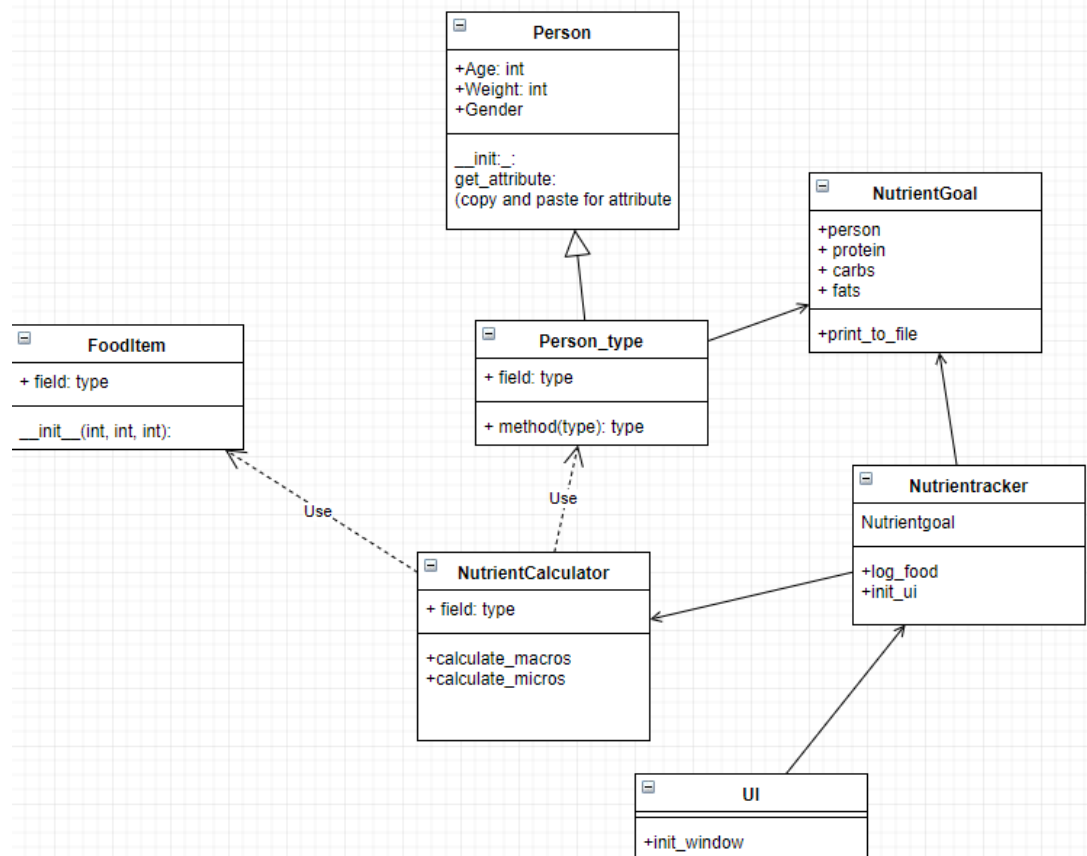
Tässä näennäisen yksinkertaisessa ohjelmassa on monia olennaisia osia, ja ne voidaan jakaa karkeasti Front- ja Back-tiedostoihin. Front-end- ja back-end-tiedostot koostuvat käyttäjälle esitettävistä elementeistä sekä tietojen käsittelystä ja laskennasta. Ensimmäisenä ongelmana on kerätä käyttäjältä tietoja, joita ohjelman muut osat voivat sitten hyödyntää laskelmissaan. Tämä olisi MainWindow-luokka: se kerää käyttäjältä olennaisia tietoja, jotka se välittää ohjelman laskentaosalle. Tässä vaiheessa kuvaan astuvat luokat NutrientGoal ja Person (jonka alaluokat edustavat eri henkilöäarkityyppejä). Ne laskevat käyttäjälle sekä energia- että makroravintoainetavoitteet. Tärkeitä metodeja näille toiminnoille ovat Personin `calculate_bmr` ja NutrientGoalin `__init__`. Jälkimmäinen sekä alustaa että laskee käyttäjälle makroravintotavoitteet BMR-laskelmien perusteella. Nämä tiedot tallennetaan NutrientGoal-luokkaan ja siirretään ohjelman seuraavaan osaan.

Kun nämä toiminnot on suoritettu, ohjelma siirtyy käyttämään ResultWindow-luokkaa, jossa on monia tärkeitä toimintoja. Ensinnäkin `initUI` luo ikkunan, johon kaikki muut elementit sijoitetaan. Piirakkadiagrammit piirretään GraphicsEllipseItemilla. Vasemmanpuoleinen piirakkakaavio on staattinen ja piirretään käyttäjän tavoitetiedoilla, kun taas oikeanpuoleinen piirakkakaavio on dynaaminen ja päivittyy käyttäjän syötteiden mukaan. Erittäin tärkeitä metodeja luokan sisällä ovat metodit, jotka päivittävät näyttöä jollakin tavalla, olipa se sitten teksti- tai graafinen elementti. Nämä metodit ovat `update_goals_display`, `update_food_list_display` ja `draw_consumed_nutrients_ellipse`, ja ne täyttävät nimensä mukaiset toiminnot. ResultWindow-luokka on repositoryn ylivertaisesti suurin tiedosto, koska se sisältää niin monia olennaisia toimintoja, jotta graafinen käyttöliittymä toimisi.

Alla on luokkakaavio ohjelmalle:



Vertauksen vuoksi, alla on suunnittelutasolla esitetty luokkakaavio:



Luokkarakennetta on muutettu jonkin verran suunnitelmasta. Syynä tähän oli käytännöllisyys. Ehdotetut NutrientCalculator- ja NutrientTracker-luokat on yhdistetty NutrientMonitor-luokaksi ja NutrientGoal-luokka on ottanut osan suunnitelluista toiminnoista. Samaan aikaan epämääräinen "UI"-luokka on muodostettu MainWindow- ja ResultWindow-luokiksi. Tällä kertaa tarvitsin GUI:n sisältävän kaksi erillistä luokkaa, joilla on eri toiminnallisuudet, joten kirjoitettiin kaksi luokkaa.

## 5. Algoritmit

Käyttäjän energiantarve lasketaan hänen ohjelmaan syöttämiensä tietojen perusteella käyttäen Institute of Medicinen kaavaa seuraavasti:

Miehille:

$$E = 662 - (9,53 * A) + PA * ((15,91 * W) + (539,6 * H))$$

Naisille:

$$E = 354 - (6,91 * A) + PA * ((9,36 * W) + (726 * H))$$

Nämä arvot lasketaan ensin valitun arkkityypiluokan (opiskelija, urheilija jne.) mukaan, ja NutrientGoal-ohjelmassa niitä käytetään sitten kunkin makroravintoaineen tavoitemassan saannin laskemiseen. Proteiinien energia saadaan kertomalla edellä mainitun kaavan ratkaisu 0,3:lla, hiilihydraattien energia käyttämällä 0,45:tä ja rasvojen energia 0,25:tä. Tämän jälkeen löydetään kunkin makroravintoaineen massat. Koska proteiinit ja hiilihydraatit sisältävät 4 kcal/g ja rasvat 9 kcal/g, massat saadaan seuraavasti:

$$\begin{aligned} m(\text{prot})(g) &= E \text{ kcal/g} * 0.3/4 \text{ kcal/g} \\ m(\text{carb})(g) &= E \text{ kcal/g} * 0.45/4 \text{ kcal/g} \\ m(\text{fats})(g) &= E \text{ kcal/g} * 0.25/9 \text{ kcal/g} \end{aligned}$$

Molempien ellipsien piirtämiseen liittyy myös laskutoimituksia. Ensimmäinen ellipsi on staattinen ja se piirretään annettujen parametrien avulla. Ensimmäinen makroravintoaineet lasketaan yhteen, jotta saadaan muuttuja, joka edustaa ravintoaineiden ”kokonaismäärää”. Sitten ellipsin osat lasketaan jakamalla kaikki makroravintoaineet kokonaissummalla (yksittäisten makroravintoaineiden tarpeet lasketaan edellä). Kulma alkaa 0:sta ja jokaisessa osakulmassa pätee seuraava kaava.

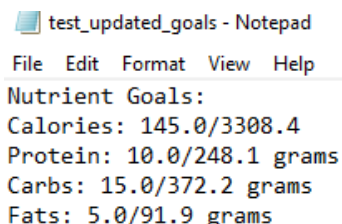
$\beta_1 = \alpha * 16 + \beta_0 * 16$ , jossa  $\alpha$  on nykyinen aloituskulma ja  $\beta_0$  on nykyisen makroravintoaineen leikkauskulma. Molemmat muuttujat kerrotaan 16:lla, koska PyQt6 toimii kuudestoista-asteina. Kun tämä operaatio on suoritettu, uusi aloituskulma lasketaan kaavalla  $\alpha_1 = \alpha_0 + \beta_1$ , jossa  $\alpha_0$  on edellinen aloituskulma ja  $\beta_1$  piirakkakaavion laskettu leikkauskulma. Kun nämä on tehty, ellipsi piirretään. Dynaaminen piirakkakaavio piirretään täsmälleen samalla tavalla, paitsi että kulmat saadaan kulutetuista ravintoaineista eikä tavoiteravintoaineista. Edellinen ellipsi poistetaan ensin graphic scenestä, jos se (ellipsi) on olemassa. Tämä tapa piirtää piirakkakaavioita ilman kiellettyjen kirjastojen käyttöä on luultavasti järkevin, sillä projektiassistenttini vinkkasi sen minulle ystävällisesti. Tutkin lyhyesti myös muita tapoja piirakkakaavioiden piirtämiseen (muuta GraphicsItem-alaluokkia), mutta ne eivät vaikuttaneet missään määrin yhtä vakuuttavilta kuin suositeltu polku. Matplotlib olisi tietysti ollut ylivoimaisesti tehokkain ratkaisu, jossa olisi säästetty kymmeniä rivejä koodia, mutta ymmärrän täysin, että se on kielletty oppimisen vuoksi.

## 6. Tietorakenteet

Projektissa on tietenkin monia erilaisia tietorakenteita, jotka ovat tärkeitä ohjelman toimivuuden kannalta. Merkittävimpiä ovat luettelot. NutrientGoal-luokka palauttaa luettelon tavoitteista, jota käytetään myöhemmin ohjelmassa hyvin monta kertaa, erityisesti ellipsejä piirrettäessä. Kyseisessä osassa käytetään yhtä liukulukua kerrallaan ellipsin osien laskemiseen, mikä sopii täydellisesti kyseiseen tilanteeseen. Sitten on tietenkin monia primitiivisiä tietorakenteita, kuten kokonaislukuja, liukulukuja tai merkkijonoja. Ohjelman myöhemmissä osissa on myös sanakirja, kun osoitetaan oikeat makroravintoaineet niiden ominaisiin arvoihin. Tämä on erityisen hyödyllistä, kun muistetaan tietoja kyseisestä makroravintoaineesta. Ohjelmassa on tietysti myös merkkijonoja, joita on kaikkialla ohjelmassa. Niitä käytetään pääasiassa kohteiden merkitsemiseen, kuten sen selvittämiseen, mikä väri edustaa mitä makroravintoaineita. Olen käyttänyt projektissani sekä muuttumattomia (esim. merkkijonoja) että muuttuvia (esim. listoja) tietorakenteita.

## 7. Tiedostot ja formaatit


Ohjelmani käsittelee yksinomaan tekstitiedostoja eikä mitään muuta. Se voi sekä lukea että luoda tekstitiedostoja ohjelman asianmukaisessa osassa. Ohjelma lukee tavoitteet rivi riviltä sanakirjamaisesta tiedostosta:



```
test_updated_goals - Notepad
File Edit Format View Help
Nutrient Goals:
Calories: 145.0/3308.4
Protein: 10.0/248.1 grams
Carbs: 15.0/372.2 grams
Fats: 5.0/91.9 grams
```

Se ohittaa ensimmäisen rivin ja alkaa määrittää kunkin rivin toista elementtiä vastaavalle makroravintoaineelle. Nämä tallennetaan sitten kokonaislukuina, jotka siirretään eteenpäin ohjelmassa.

Kun ohjelma luo tekstitiedoston, se näyttää jokseenkin samankaltaiselta kuin lukuvaiheessa:

 Test\_export - Notepad

File Edit Format View Help

Nutrient Goals:  
Calories: 597.0/3308.4  
Protein: 47.0/248.1 grams  
Carbs: 55.0/372.2 grams  
Fats: 21.0/91.9 grams

## 8. Testaus

Ohjelman testaaminen tapahtuu suurin piirtein alkuvaiheessa esitetyn suunnitelman mukaisesti, vaikka se jätettiin tarkoituksella epämääräiseksi. Ohjelmassa on testit NutrientGoal- ja NutrientMonitor-ohjelmien kelvollisten syötteiden testaamiseksi, jotta tavoite- ja ruokapäiväkirjatiedot olisivat kelvollisia. Se testaa myös MainWindow'n otsikot ja avaa ikkunan (vaikka sitä ei olekaan testattu), ja se tarkistaa, että virheellisiä käyttäjän syötteitä ei hyväksytä eteenpäin. ResultWindow-ikkunan osalta on muutama yksinkertainen testi, joilla varmistetaan, että init-toiminto ajaa oikein.

Ohjelmaa testattiin sekä manuaalisesti syöttämällä tietoja käynnissä olevaan ohjelmaan, kuten käyttäjä normaalisti tekisi. Sitten on tietenkin olemassa kolme erillistä tiedostoa ohjelman eri osien testaamista varten. Tällä hetkellä ohjelma läpäisee kaikki testit, jotka voidaan tehdä testaustiedostoissa paitsi yhden, jonka se tarkoituksellisesti epäonnistuu.

## 9. Tunnetut ongelmat ja puutteet

Olen yrittänyt varmistaa, että ohjelman normaalin käytön aikana vääriä (negatiivisia tai virheellisiä) syötteitä ei hyväksytä kerätessä tietoja käyttäjältä tai ruokien kirjaamisprosessissa. Tiedän kuitenkin, että paino-, pituus- ja ikäluokkien suuret syötteet hyväksytään. Tämä tarkoittaisi sitä, että jos käyttäjä syöttää ikänsä arvoksi 1000, ohjelma antaa hänelle negatiivisen kalori- ja makroravintotavoitteen. Lisäksi ohjelma unohtaa käyttäjän syöttämät tiedot, jos se suljetaan. Siksi ohjelma pystyy lukemaan kulutetut ravintoaineet myös tavoitetiedostoista. Minun on myönnettävä, että nykyisillä tiedoillani en pysty luomaan ohjelmalle sellaista käyttäjäkantaa, että se muistaisi käyttäjät myös poistumisen jälkeen.



Arvelen myös, että tekstitiedostojen toiminnoissa on jotain, joka aiheuttaisi niiden hajoamisen. Minulla ei ole ollut riittävästi aikaa testata niitä perusteellisesti, joten saattaa olla jotain mitä en ole huomannut. Tällä hetkellä ne toimivat tarkoitetulla tavalla, mutta tiedän etteivät ne ole niin vankkoja kuin haluaisin niiden olevan.

Ohjelmassa on muutamia osia, joita haluaisin parantaa edelleen. Jos tämä olisi työtä varten, yrittäisin tehdä käyttöliittymästä vielä helpommin ymmärrettävän yhdellä tai kahdella lisätoiminnolla. Toteuttaisin ohjelmaan myös salasanan ja muistin ulkopuolisen lopetuksen, jos minulla olisi enemmän aikaa projektille. Toteuttaisin myös parempaa tiedostojen käsittelyä ohjelmaan, jotta niihin mahtuisi enemmän tietoa ja jotta ne pystyisivät myös omaksumaan enemmän tietoa käyttäjältä.

## 10. Kolme parasta ja kolme heikointa

Aloitan ohjelman heikoimmista kohdista. Kuten mainitsin, tekstitiedostotoiminnoissa on luultavasti jotain pinnan alla olevaa vikaa. Ne vain tuntuvat minusta ”heikoilta” ja ne voidaan luultavasti hakkeroida hyvin helposti.

Seuraava on hyvin selvästi testaus. Tiedän, että testit ovat hyvin yksinkertaisia ja testaavat melko pinnallisia asioita, kuten otsikon pätevyyttä. Minun on myönnettävä, että yksikkötestit ovat heikoimpia alueitani

Python-ohjelmoinnissa; pystyn hyvin helposti seuraamaan omaa ohjelmaani virheenkorjauksen näkökulmasta, mutta kun testauksessa oleellinen ”toinen kerros” lisätään mukaan, tipun hieman kärryiltä.

Ohjelman heikoimpiin osiin kuuluu visuaalinen esitys, se sisältää kyllä kaiken tarvittavan tiedon, mutta tekee sen hyvin alkeellisella tavalla. Jos minulla olisi enemmän aikaa, haluaisin säätää fontteja, tekstin sijoittelua, kuvia ja niin edelleen.

Vaikka sanoin, että visuaalinen esitys on yksi ohjelman heikoimmista osa-alueista, se on samalla vahvin. Se on hyvin yksinkertainen ymmärtää. On otsikot, jotka kertovat käyttäjälle, mitä hänen pitäisi odottaa joltakin ikkunan alueelta tai widgetiltä.

Seuraava hyvä puoli on helppokäyttöisyys. Voin väittää, että jokainen käyttäjä ymmärtää, mitä tapahtuu sen jälkeen kun hän ajaa ohjelman alustalla, ja siitä olen hyvin ylpeä.

Kolmanneksi paras osa on päivittyvä graafinen käyttöliittymä. Olen ylpeä siitä, että näyttö muuttuu käyttäjän syötteiden mukaan, ja tuntuu kuin käyttäjä osallistuisi aktiivisesti ohjelman suorittamiseen.

## 11. Poikkeamat suunnitelmasta

Ainoa asia, joka mielestäni poikkeaa suunnitelmasta, on se, etten sisällyttänyt ohjelmaan mikroravintoaineita. Suoraan sanottuna olin jo niin pitkällä ohjelmointiprosessissa, kun muistin, että minun piti ottaa ne käyttöön, etten vain halunnut täysin uudistaa alustavaa prosessia. Pelkäsin, että se johtaisi ongelmiin, joita en pystyisi kovin helposti tunnistamaan.

Projektin aikataulu on suunnilleen kuten kuvittelin, vaikka olenkin tainnut käyttää tässä vaiheessa yli 100 tuntia. Jos kuudennessa checkpointissa olin käyttänyt noin 60 tuntia, tässä vaiheessa olen varmaan jo yli 100. Olin hyvin yllättynyt siitä, kuinka kauan dokumentin kirjoittaminen kesti. Korjasin monia virheitä koodussa ja lisäsin joitakin asioita sitä kirjoittaessani, kun minun oli pakko tarkastella projektini toiminnallisuutta.

Toteutusjärjestys oli kuitenkin juuri sellainen kuin suunnittelin: ensin tekstipohjainen ohjelma, jossa on GUI:n tukitoiminnot, sitten GUI ja sen toiminnot ja lopuksi testaus.

## 12. Aikataulu

Seuraava on hyvin karkea aikataulu ohjelman rakentamiselle tänä keväänä. En pysty muistamaan päivämääriä, mutta kerron, minkä kanssa työskentelin ja suunnilleen milloin.

Alkukevät: Projektisuunnitelma, ensimmäiset luokat, main-funktio.

Maaliskuu: Valmis tekstipohjainen ohjelma, lähes kaikki olennaiset toiminnot valmiina, ensimmäiset yritykset graafisen käyttöliittymän luomiseksi, ensimmäiset testit, monia virheiden korjauksia.

Huhtikuu: GUI:n laajentaminen, dynaamisten painikkeiden lisääminen, toisen ellipsin lisääminen, tiedostotoimintojen parantaminen huomattavasti, hyvin paljon virheiden korjauksia.

Toukokuu: Muutamien viimeisten toimintojen lisääminen, viimeiset testit, viimeiset virheiden korjaukset, asiakirjan kirjoittaminen.

Sanoisin, että yleinen aikataulu ja toteutusjärjestys eivät poikenneet merkittävästi suunnitellusta järjestyksestä.

## 13. Lopputuloksen arviointi

Tämä on siis ollut ylivoimaisesti suurin ohjelmointirupeama, jonka olen suorittanut. Se on ollut ajoittain vaikea, varsinkin kun on kyse ongelmien diagnosoinnista ja virheiden havaitsemisesta kirjoittamassani noin 600 koodirivissä. Se on kuitenkin sytyttänyt mielikuvitukseni ja osoittanut minulle, että jonkin luominen tyhjästä on mahdollista ja jopa hauskaa!

Lyhyesti sanottuna ohjelmani laskee kalorit ja ravintoaineet käyttäjän taustatietojen perusteella. Se voi myös tuoda tavoitteita tiedostoista. Se näyttää, kuinka paljon käyttäjä tarvitsee ja kuinka paljon hän on kuluttanut ohjelman ollessa käynnissä, ja voi sitten viedä nämä tiedot tekstitiedostoon.

Ohjelma on mielestäni sekä melko kestävä että helppokäyttöinen ilman ennakkotietoja. Siinä on kuitenkin puutteita testauksessa, tiedostotoiminnoissa ja käyttäjien muistamisessa. Tulevaisuudessa ohjelmasta voitaisiin tehdä näyttävämpi graafisilla elementeillä, se voisi hyväksyä ja toimittaa enemmän tietoa tiedostoista ja tiedostojen kanssa. Siihen voitaisiin myös asentaa käyttäjäkanta, jotta se voisi muistaa käyttäjät ilman että niitä tarvitsee tallentaa tiedostoon.

Yleisesti ottaen pidän ohjelman rakennetta hyvänä, ja olen käyttänyt perintöä hyvin silloin, kun sitä on tarvittu. Ainoa merkittävä asia, jonka haluaisin muuttaa tällä osastolla, on ResultWindow-luokka. Se on hyvin suuri, ja jos olisin tiennyt että siitä tulisi niin suuri, olisin erottanut joitakin funktioita eri tiedostoon. Kaiken kaikkiaan olen sitä mieltä että nykyinen projektia voisi laajentaa ilman suuria muutoksia rakenteeseen. Tämä johtuu rakenteen modulaarisuudesta ja alenevasta luonteesta, jossa monet eri haarat yhdistyvät lopulliseen esitettyyn näkymään. Luokan tai kahden lisäämisellä edellä mainitut muutokset olisivat mielestäni mahdollisia.

Tämän ohjelman algoritmit ovat melko suoraviivaisia, joten epäilen, ettei niitä olisi voitu tehdä paljon tehokkaammin. Mielestäni jotkut tietorakenteet olisivat voineet tarvita standardointia: jossain ne ovat kokonaislukuja ja jossain liukulukuja.

Olen erittäin tyytyväinen sekä oppimaani että tähän projektiin näkemääni vaivaan.

## 14. Viittaukset ja lähteet

1. Ramalho, Luciano. *Fluent Python*. Sebastopol, CA: O'Reilly, 2015. Print.
2. Python Data Structures <https://docs.python.org/3/tutorial/datastructures.html>
3. *Dietary Reference Intakes for Energy, Carbohydrate, Fiber, Fat, Fatty Acids, Cholesterol, Protein, and Amino Acids*. Washington, DC: National Academies Press, 2005. Print.
4. Unit tests <https://docs.python.org/3/library/unittest.html>
5. PyQt6 <https://www.riverbankcomputing.com/software/pyqt/>
6. QtWidgets <https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/index.html>
7. Course home page <https://plus.cs.aalto.fi/y2/2024/>
8. QtCore documentation: <https://doc.qt.io/qt-6/qtcore-index.html>
9. Sys documentation: <https://docs.python.org/3/library/sys.html>
10. YouTube: <https://www.youtube.com/?app=desktop&gl=FI&hl=fi>
11. GraphicsEllipseItem: <https://doc.qt.io/qt-6/qgraphicsellipseitem.html>
12. Stackoverflow (myös joitain yleisiä vinkkejä): <https://stackoverflow.com>

## 15. Liitteet

Kaikki oleellinen on zip-kansiossa MyCoursesin palautuskansiossa. Kuvat ohjelman toiminnasta ovat tässä dokumentissa.