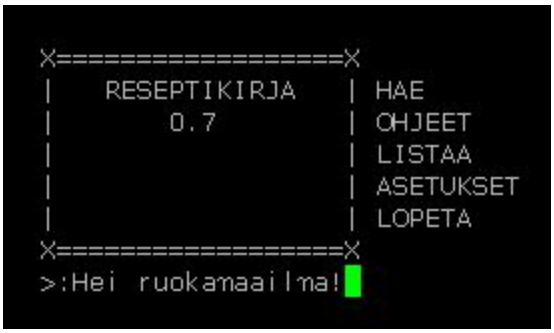
Älykäs reseptikirja

Projekti, CSE-A1121

Pietu Roisko, 298362
EST, 4. vuosikurssi
5.5.2016



Kuvakaappaus ohjelman suorituksesta

Yleiskuvaus

Projektin tavoitteena oli toteuttaa älykäs reseptikirja. Tehtävänannossa vaadittiin perustoimintona lähinnä monipuolista hakuominaisuutta.

Keskityin projektissa käytettävän ohjelman luomiseen. Ohjelma lataa tiedostoista käyttäjän jääkaapin (Analogia ruokakaapille. Tehtävässä ei eroteltu kylmässä pdiettäviä aineksia normaaleista), sekä reseptit tietystä kansiosta. Ohjelma myös lataa allergiatiedot tiedostosta. Tämän jälkeen käyttäjä voi määrittää omat allergiansa ja suorittaa hakuja jääkaapin sisällön perusteella.

Ajattelin toteuttaa työn keskivaikeana, mutta loppujen lopuksi “helppo”-taso vei niin paljon aikaa että käyttöliittymä koodaukseen ei ollu tarpeeksi aikaa. Ohjelman käyttöliittymä onkin projektin heikoin osa.

Tehtävänannossa pyydettiin toteuttamaan kaikki tiedostot selkokielisinä. Näin ollen noudatin hyvää unix-pohjaista mottoa “asetustiedostot on tehty muokattaviksi jos niitä voi lukea”. Myös kurssilla on mainittu “human-readable-editable” tiedostot. Tästä syystä en lähtenyt toteuttamaan ohjelman sisäistä jääkaapin tai reseptie lisäys-tai muokkaus moduulia.

Oma toteutus: Helpot vaatimukset, keskivaikeiden vaatimusten toteuttamista aloitettu.

Käyttöohje

Käynnistys

Ohjelman ajoon tarvitaan python 3.4 tai uudempi.
Aja ohjelma komennolla `python3.4 main.py`

Käyttöliittymä

Ohjelma tervehtii käyttäjää valikolla (HAE, OHJE LISTAA, ASETUKSET, LOPETA).

HAE

Ohjelma siiryy hakutilaan.
Mikäli et ole määrittänyt allergioita, tulee ohjelma kysymään allergiatietosi. Ne tallennetaan asetustiedostoon.

Käyttäjä voi hakea:

(1) Jääkaapin sisällön perusteella
(2) Ainesosan perusteella
(3) Nimen perusteella

Hauissa otetaan allergiatiedot huomioon.

(1) Kohdassa listataan myös mitä reseptin aineista puuttuu ja kuinka paljon.

OHJE

Tulostaa ohjeen

LISTAA

Ohjelma siiryy listaustilaan.

Käyttäjälle voi listata:

(1) jääkaapin sisällön
(2) ladatut resptit
(3) Tietyn reseptin sisällön

(3) Kohdassa listataan myös mitä reseptin aineista puuttuu ja kuinka paljon.

ASETUKSET

Ohjelma tulostaa ladatut asetukset

LOPETA

Ohjelman suoritus loppuu.

Ohjelmasta voi myös poistua ctrl+d yhdistelmällä

Tulosteet

Ohjelma kertoo haettaessa jääkaapin sisällöllä, tai yksittäistä reseptiä haettaessa, mitä jääkaapista uupuu.

Tarkastellaan esimerkkiä, jossa haetaan jääkaapin sisällön pohjalta:

```
Jääkaappitiedosto versio 1.0
#Matin jääkaappi
JÄÄKAAPIN SISÄLTÖ
  Maito       1 purkki  a 10dl
  Kinkku      1 paketti a 400g
  Sitruuna    1 kpl
  Sipuli      1 kpl
  Avokado     200 g
  Jauheliha   1 paketti a 350g
```

Haku y.o. Jääkaapin pohjalta voisi olla seuraava.

```
Löytyi 4 kpl tuloksia:
 makaroonilaatikko (Puuttuu 5 ainesosaa)
            Puuttuvat:
                 Makaronia              150 g
                 Kananmunia             2 kpl
                 Juustoraastetta
                 Suolaa
                 Pippuria
```

```
            Ei puutu:
                  sipuli                  150 g ( Kaapissa 1 kpl)
                  jauheliha               200 g ( Kaapissa 350g )
                  maito                   3 dl ( Kaapissa 10dl )
      sima (Puuttuu 5 ainesosaa)
            Puuttuvat:
                  Vettä                   8 l
                  Fariinisokeria          5 dl
                  Sokeria                 6 dl
                  Hiivaa                  teelusikallinen
                  Sitruuna                1 kpl (kaapissa oleva määrä
1 kpl voi riittää pieneen erään)
            Ei puutu:
```

Esimerkki hakutuloksesta

Yllä olevassa esimerkissä näemme, että **makaroonilaatikon** tekemisestä puuttuu 150g makaroonia ja 2kpl kananmunia.

Jäkaapissa on jo valmiiksi sipulia (1kpl) ja jauhelihaa 350g (reseptiin tarvitaan 200g). Lisäksi meillä on maitoa 10dl joka riittää hyvin.

Siman tekemiseen meidän pitäisi ostaa fariinisokeria, sokeria, hiivaa. Reseptissä vaaditaan 2kpl sitruunoita, mutta jääkaapissa on 1kpl. Ohjelma ehdottaa että 1kpl vi riittää pienempään erään.

Kehittynyttä, eikö?!

Ohjelman rakenne

Ohjelma on jaettu moduuleihin toiminnallisuuden perusteella. Moduulit ovat

jääkaappiMod.py

Määrittelee luokan Jaakaappi.
Jääkaappi huolehtii jääkaapin ja allergioiden lataamisesta ja jääkaapin sisällön parsimisesta.

Tärkeitä metodeita ovat **mitaPuuttuu**, joka kertoo mitä kyseisen reseptin toeuttamisesta puuttuu (jääkaapin sisällön perusteella) ja **lapaiseeAllergiat**, jonka avulla tarkistetaan läpäiseekö resepti allergiavaatimuksia.

reseptiMod.py

Määrittelee luokat Resepti ja Reseptikirja.
Reseptit sisältävät reseptitiedoston mukaisen listan ainesosista.

Reseptikirja huolehtii reseptien lataamisesta “resepti”-olioihin ja yksinkertaisista hakufunktioista.

Tärkeitä metodeita ovat **haeAinesosalla** ja reseptin latausfunktio.

ainesMod.py

Määrittelee luokat ruokaAines ja ainesOsa (ruokaAines perillinen).
ruokaAines on pohjaluokka ainesOsalle.

ruokaAines-luokkaa käytetään resepteissä (resepti-olioissa) yksinkertaisena luokkana ainesten tallentamiseen.

AinesOsa on mutkikkaampi luokka jota käytetään jääkaapin sisällön määrittämiseen. Se sisältää jääkaappitiedoston mukaiset tietueet.

Tärkeä metodi on ainesOsa-olion init-funktio, sekä allergioiden parsintafunktio **parsiAllergiat**, joka tarkistaa allergialippujen oikeellisuuden.

Tiedostossa määritellään myös oma virheluokka AinesParseError.

hakuMod.py

Tämä tiedosto toteuttaa monimutkaisemmat hakufunktiot.

haeValmistettavat funktio hakee resepteistä ne reseptit, jotka voidaan toteuttaa N:llä puuttuvalla aineella. Funktio myös huomioi **alireseptien** tunnistamisesta ja läpikäymisestä jos aines-osaa ei ole satavilla. Alireseptit toimivat hienosti. Ohjelma myös kertoo että tarvitsee käyttää alireseptiä.

Funktio kutsuu jääkaappiMod.py:n allergiatarkistuksia hakujen jälkeen. Funktio palauttaa listan, joka on järjestetty puuttuvien aines-osien lukumäärän mukaan kasvavassa järjestyksessä.

functionsMod.py

Tämä tiedosto määrittelee erilaisia apufunktioita.

Tärkeitä funkioita:
wordSimilarity funktion avulla voidaan vertailla eri sanojen (aineksien) nimiä. Näin ohjelma ymmärtää että reseptissä esiintyvä *jauhoja* on sama kuin jääkaapissa oleva *jauhot* (ks. algoritmit).

erotus funktio huolehtii kahden ainemäärän vertailusta. Funktio muuntaa ainesmäärät samaan yksikköön ja laskee erotuksen (ks. algoritmit). Erotus esitetään “preferoidussa” yksikössä, jos sellainen on määritetty.

Tiedostossa määritellään myös omat virhefunktiot NotFoundError ja Loadingerror.

main.py

Nimensä mukaan main.py sitoo yhteen muiden moduulien funktiot käyttöliittymään. Tiedostossa toteutetaan yksinkertainen tekstipohjainen käyttöliittymä.

On syytä huomioida että tämän tehtävän pääfokus ei ole ollut käyttöliittymässä. Algoritmit ja ohjelman sisäiset funktiot ovat vieneet ~90% ohjelmointityöstä.

mainx.py

Tarkoituksena oli toteuttaa myös graafinen käyttöliittymä. Tämä kuitenkin jäi kesken.

Algoritmit

onkoTuotetta

Tiedostossa jaakaappiMod.py on funktio *onkotTuotetta*, joka tarkistaa löytyykö kaapista jotakin tiettyä *ruokaAine*en mukaista elintarviketta. Jos löytyy, palauttaa funktio elintarvikkeen määrän.

Funktio osaa myös laskea tuotteiden summan, eli jos kaapissa on esimerkiksi kaksi 1L maitopurkkia, palauttaa funktio tuloksena 2L.

onkoTuotetta on käytössä usein juuri ennen erotusfunktion käyttöä.

Erotusfunktio

Tiedostossa functionMod.py määritellään erotusfunktio. Se mukailee tehtävänannon mukaista

“aine ... mitataan erilaisilla mitoilla. Kaapissa oleva jauho myös ostetaan kiloittain, mutta mitataan resepteissä desilitroissa. Tee ohjelmaasi luokka joka hoitaa kaikki muunnokset mittojen välillä.”

Toteutus ei ole luokka, mutta sen voinemme antaa anteeksi. Erotusfunktiolle annetaan parametrinä reseptin mukainen *ruokaAine* (ja tiedot sen määrästä ym), sekä kaapista löytynyt vastaava Selitetään lyhyesti erotusfunktion toiminta:

- Selvitetään molempien aineiden määrä ja yksiköt
- Jos yksiköt täsmäävät, voidaan erotus laskea suoraan
- Muutoin tarkistetaan löytyykö molempien aineiden yksiköt *tilavuudet*-listasta, tai *massat*- listasta. Jos näin on voidaan olettaa että jonkinlainen muunnos on mahdollista.
    - Seuraavaksi tarkistetaan onko aineen tiheys määritelty *esimääreet*-listassa. Jos on voidaan muunnos suorittaa painosta tilavuuteen ja päinvastoin
      TAI
    - Jos molempien yksiköt ovat tilavuus- tai massasuureita voidaan muunnos tehdä ilman tiheystietoja (tätä ei ole tällähetkellä toteutettu)

Kyseisen funktion toteuttaminen hankaloituu myös mikäli reseptin aineet on taivutettu. Sanojen vertailun takia ohjelma kuitenkin pärjää (ks. alla).

Säännölliset lausekkeet

Ohjelman tiedostojen parsinnassa käytetään säännöllisiä lausekkeita (regular expressions).

Yksinkertaisimmiillaan regexiä käytetään rivien pilkkomiseen whitespacen pohjalta, esimerkiksi allergiatiedostossa. Tämä tehdään komennolla
`re.split('\s+', rivi.rstrip().lower()).`
Komento “\s” koskee whitespacea ja “+” merkitsee yksi-tai-enemmän.

Jääkaappitiedostojen parsinnassa käytetään kuitenkin monimutkaisempaa
`re.findall(r"\s?(\w+)\s*(\d*\.?\d*)?(\w+)?(?:\n|$)", rivi)[0]`
Lauseke löytää whitespacella erotellut ryhmät ja erotteleetodellisen ainemäärän muodon “9dl” osiin “9” ja “dl”.

Findall palauttaa listan, joka koostuu joukoista (set). Kullakin rivillä on kuitenkin vain yksi joukko, joten käytetään [0].

Sanojen vertailu

Olen aikaisemmin tutustunut Simon Whiten uudenlaiseen sanojenvertailu- algoritmiin. Algoritmi vertailee sanoja kirjainparien avulla.

Yksinkertainen leksikografinen sanojen vertailu on myös toteutettu useilla ohjelmointikursseilla (esim C:ssä sana1 < sana2). White kuvailee artikkelissaan, miksi algoritmi on hyvä. Lyhykäisyydessään se vertailee sanojen samanlaisuutta, mutta tulkitsee myös ekvivalensseja. Se on myös kieliriippumaton ja sopii hyvin suomen kielen taivutettujen sanojen vertailuun, koska suomen kielessä käytetään liitteitä.

Algoritmissä sanat jaetaan kirjainpareihin. Tämän jälkeen tarkistetaan montako yhteistä kirjainparia sanoilla on. Pairs(s) kertoo kuinka montaa kirjainparia sanassa on. Sanojen s1 ja s2 samankaltaisuus lasketaan tämän jälkeen kaavalla:

$$similarity(s1,s2) = \frac{2 \times |pairs(s1) \cap pairs(s2)|}{|pairs(s1)| + |pairs(s2)|}$$

Algoritmillä ei ole nimeä, eikä sitä ole tietääkseni julkaistu tieteellisessä julkaisussa. Tämän vuoksi algoritmi voi olla monille tuntematon.

Käytin ohjelmassa “saman sanan” kynnysarvona lukuja 0.7-0.8 (1 = sama sana algoritmin mukaan). Tämä oli ns. hihavakio.

Mielestäni oli tärkeää että ohjelma toimii “tavallisessa käytössä”, jossa muiden kirjoittamien reseptien taivutusmuotoja ei voida tarkistaa tai vaatia olemaan perusmuodossa. **Sanojen samankaltaisuuden vertailu lisää tiedostojen joustavuutta ja mahdollistaa “oikean” käytön.**

Hakualgoritmit

Ohjelman haut on suoritettu yksinkertaisilla silmukoilla, jossa reseptit iteroidaan läpi. Näin ohjelman aikakompleksisuus on monesti riippuvainen reseptin ainesosista tai jääkaapin sisällön määrästä kompleksisuudella O(n^2).

Mikäli ruoka-aineita ja reseptejä olisi *paljjon* tulisi hakualgoritmejä kehittää. Nykyiseen käyttöön tavalliset silmukat kuitenkin riittävät.

Haluan myös huomauttaa että huomasin pari päivää ennen projektin palautuspäivää, että **projektin olisi voinut varmasti toteuttaa pythonin tietokantakirjastoilla**, jolloin homma olisi ollut ns. erittäin helppo. Tietokannat eivät kuitenkaan olleet vaatimus ja nykyinen toteutus kokemusta matalan-tason hakufunktioiden kehittämisessä. Tietokantojen osalta myös samankaltaisten sanojen vertailu olisi ollut hankalampaa.

Tietorakenteet

Tieto on tallennettu omiin luokkiin (Jaakaappi, Resepti, ruokaAines, ainesOsa, Resepti, ReseptiKirja). Suuressa osassa aineiden tallennusta ja vertailua ainekset tallennettiin listoihin, jotka koostuivat monikoista (tuple).

Pythonissa listat ovat mutable ja monikot (tuple) ovat immutable. Käytössä siis molempia. Tekovaiheessa myös joukkoja (set) käytettiin ahkerasti, mutta allekirjoittanut huomasi huolestuttavia ominaisuuksia kun joukko muutetaan listaksi - python voi järjestää listen sattumanvaraisesti. Tämä johti eri tuloksiijn eri ajokerroilla, joten todettiin, että joukot eivät ole tähän tarkoitukseen.

Alla kaksi kiinnostavaa esimerkkiä:
Värikoodauksena on luokat ja luokan instanssit.

Jääkaappi

```
class ruokaAines
      self.nimi
class ainesOsa (ruokaAines)
      self.maara
      sel.yksikko
class Jaakaappi
      self.ruokaAineet [ ainesOsa, ainesOsa ]
      Self.allergiat [ ‘G’, ‘P’ ]
```

Y.o. pseudokoodi kuvaa sitä, minkälaisessa tietorakenteessa jääkaappin ruoka-aineet säilytetään. Jääkaapin **ruokaAineet** on lista, joka koostuu luokan **ainesOsa** instansseista.

Instanssit omaavat mm. Attribuutit **nimi, maara, yksikko ja todMaara**. TodMaara määrittelee “todellisen määrän” (1 purkki ei ole todellinen määrä, mutta 30g on).

Allergiat tallennetaan tällä hetkellä vain lippuina. Tulevaisuudessa allergiat voitaisiin muuttaa monikoksi (tuple), jolloin myös ruokaAines voisi olla allergian kohde.

Reseptikirja

```
class ruokaAines
      self.nimi

class resepti
      self.ainekset [ (ruokaAines, 1, kpl), (ruokaAines, 400, g) ]

class ReseptiKirja
      self.reseptit [ resepti, resepti]
```

Tämä pseudokoodi taas kuvaa kuinka respetikirjan ja reseptien tietorakennetta. Huomaa, että ohjelmassa käytetään useaan otteeseen samoja tietorakenteita. Näin ollen olemassa olevien luokkien on oltava ainakin aika hyvin jäsennelty.

Tiedostot

Koska ruoka-aineilla on monia ominaisuuksia on tiedoston hallintaan olisi mahdollista käyttää ohjelman luomaa rajapintaa. Näin ollen jääkaapin sisältö voitaisiin tallentaa binääritiedostossa (lohkomuodossa). Tämä helpottaa tiedoston parsimista, koska virhealttius on pienempi. Valitettavasti tehtävänanto määrittää silti, että kaikki tiedostot tulee tallentaa tekstimuodossa. Tämä asetti haasteita tiedostoje parsimiselle.

Tiedostot

- Asetukset.txt
- Allergiat.txt - allergiatietoja aineista
- Jaakaappi.txt - jääkaapin sisältö
- /reseptit/ kansio

Asetustiedosto

Asetustiedosto on aina ohjelman kansiossa sijaitseva ‘asetusket.txt’.
Asetustiedoston avulla määritellään muiden tiedostojen ja kansioiden sijainnit. Se myös tallentaa käyttäjän allergiatiedot seuraavaa kertaa varten.

Tiedosto alkaa **otsikkorivillä**, jonka tulee alkaa sanoilla “Asetustiedosto”. Tämän jälkeen seuraa **asetuksia** muodossa “asetus arvo” rivi kerrallaan. **Kommentit** aloitetaan risuaidalla (#) ja ne jatkuvat rivin loppuun.

allergiat asetus kertoo käyttäjän omat “default” liput joilla haetaan. Rivin voi jättää pois tai asettaa arvoksi None, mikäli käyttäjä ei ole allerginen millekään. Ks. mahdolliset liput alla kohdassa “allergiatiedosto”.

Esimerkki asetustiedostosta:

```
Asetustiedosto
Allergiatiedosto   allergiat.txt
Jaakaappitiedosto  jaakaappi.txt
reseptikansio            ./reseptit/
allergiat                P,L
```

Jääkaappitiedosto

Vaikka nimen mukaan puhutaan jääkaapista, ei ohjelma erottele kuivia, märkiä tai kylmiä aineita säilytyspaikan perusteella. Jääkaappi on analogia kaapille, ja se tulee mieltää enemmän “varastona”.

Tiedosto alkaa **otsikkorivillä**, jonka tulee alkaa sanoilla “Jääkaappitiedosto”. Versionumeroa ei käytetä, mutta se ei aiheuta virhettä ja voi olla hyödyksi tulevissa versioissa.

Jääkaapin **sisältölistaus** alkaa otsikon “JÄÄKAAPIN SISÄLTÖ” jälkeen. Sisältölistauksen formaatti on:

```
Aines, kpl, kplyksikkö, [ à, todellinen määrä, allergiatiedot, parasta ennen]
```

Huomaa että erottelumerkkinä tiedostossa käuytetään whitespacea (uosituksena sarkain <tab>). Ainoastaan kolme enismmäistä parametria ovat pakollisia. Huomaa myös että “todellinen määrä” tulee kirjoittaa yhteen ilman välilyöntiä yksikön ja määrän välillä.

Ohjelmaan tähän versioon ei ole ehditty toteuttaa allergiatietojen lataamista jääkaapsita (tulevat erillistiedostosta) eikä päivämäärän hyödyntämistä. **Kommentit** aloitetaan risuaidalla (#) ja ne jatkuvat rivin loppuun.

Esimerki jääkaappitiedostosta:

```
Jääkaappitiedosto versio 4.5
# Matin jääkaappi
JÄÄKAAPIN SISÄLTÖ
Maito            1     purkki      à     10dl        VL    3.4.2016
Sipuli           1     kpl
Makaroneja       1     pussi       à     1kg
```

Allergiatiedosto

Allergiatiedostoon tallennetaan tunnettujen ruoka-aineiden allergiatiedot. Alkuperäisessä suunnitelmassa allergiatiedot piti laittaa jääkaappitiedoston määre-osaan, mutta tämä jääköön tulevaan versioon.

Tiedosto alkaa **otsikkorivillä**, jonka tulee alkaa sanoilla “Allergiatiedot”. Tämän jälkeen seuraa **allergiatietoja** muodossa “aine allergialiput” rivi kerrallaan. **Kommentit** aloitetaan risuaidalla (**#**) ja ne jatkuvat rivin loppuun. Huomaa että aineella voi määritellä useta allergiatietoja.

Allergialiput ovat kirjaimia, jotka kuvaavat aineen allergiatietoja:

L	Laktoosi
P	Pähkinä
G	Gluteeni
A	Allergeenit
S	Soija

Ohjelmassa määritellyt allergia liput

Lippujen käyttö selviää alal olevasta esmierkkitiedostosta:

```
Allergiatiedot
# L = laktoosi, P = Pähkinä, G = Gluteeni
Maito             L
Kerma             L
Pähkinä           P
Snickers          P
Makaronit         G
```

Reseptit

Reseptit ladataan asetustiedoston määrittelemästä kansiosta. Kaikki **.txt** tiedostot luetaan.

Tiedosto alkaa **otsikkorivillä**, jonka tulee alkaa sanoilla “Reseptitiedosto”. Versionumeroa ei käytetä, mutta se ei aiheuta virhettä ja voi olla hyödyksi tulevissa versioissa.

Tämän jälkeen seuraa **nimirivi**, joka määrittää reseptistä mudostuvan ruoan nimen.

Nimi jälkeen seuraa **annosrivi**, joka kertoo kuinka ison määrän ruokaa resepti tuottaa. Reseptin **aineslistaus** alkaa otsikon “RAAKA-AINEET” jälkeen. Listauksen formaatti on:

```
Aines, [määrä]
```

Huomaa että erottelumerkkinä tiedostossa käuytetään whitespacea (uosituksena sarkain <tab>). Ainoastaan kolme enismmäistä parametria ovat pakollisia. Huomaa myös että määrä tulee kirjoittaa yhteen ilman välilyöntiä yksikön ja määrän välillä. Aineslistauksen jälkeen seuraa otsikko “OHJEET”, jonka alle voidaan sisällyttää **ruoan teko-ohje**.

Esimerrki reseptitiedostosta:

```
Reseptitiedosto versio 1.0
Makaroonilaatikko
1 Vuoka
RAAKA-AINEET
Sipulia           150g
Jauhelihaa        200g
Makaronia         150g
Maitoa            3dl
Kananmunia        2kpl
Juustoraastetta
Suolaa
Pippuria
OHJEET
Keitä makaronit ja paista jauheliha. Sekoita kanamunat maitoon. Sekoita
kaikki vuokaan ja ripottele juustoa päälle. Paista 15min.
```

Testaus

Ohjelmassa on toteutettu Yksikkötestit pythonin unittest-moduulin avulla. Testit ovat kunkin moduulitiedoston lopussa (esim functionsMod.py) ja ne ajetaan jos tiedosto ajetaan yksin.

Testit on luotu funktiota tehdessä, joten niitä on käytetty funktioiden testaamiseen. Tästä johtuen lopullinen käyttöliittymä on jäänyt vähälle kehitykselle.

Miltei kaikille funktioille on kirjoitettu ainakin yksi yksikkötesti. Suurin osa testauksesta on lataamisfunktioissa, sekä hakufunktioissa. Testeissä ladataan ahkerasti dataa, joka on luotu stringIO-moduulin avulla.

Hyvänä esimerrkinä kannattaa katsoa functionsMod.py:n testi **test_erotus**, jossa testataan erotusfunktiota, eli funktiota joka laskee aineiden erotuksia. Funktio muuttaa aineiden yksiköt lennossa (ks. algoritmit).

Yksikkötestejä voi aina hioa lisää, mutta ne tarkistavat perustoiminnot tyydyttävästi.

Puutteet ja viat

Kosmeettiset

- Assistentille olisi voinut laatia enemmän hyviä esimerkkitiedostoja ja reseptejä jota kokeilla.
- Reseptitiedostossa aineen määrä täytyy olla muodossa “1g” tai “1vuoka”, ei muodossa “1 vuoka”. Eli numeron ja kirjainyhdistelmän tulee olla yhdessä. Helppo korjata parantelemalla parsinta. Ajanpuutteen takia kuitenkin jäi.
- Kommentit englanniksi ja suomeksi. Koodi kuitenkin pääpiirteittäin suomeksi (muuttujat ym.)
- Tiheyksien ja “preferoitujen” yksiköiden lataus tiedostosta ei tässä versiossa ole mahdollista. Tiheydet on koodattu tiedostoon “functionsMod.py”, jossa erotus-funktio hoitaa yksiköiden välillä muuntamisen. Olisi siistimpää jos tiheydet ym.ladattaisiin tiedostosta.
- Kommentteja saisi olla lisää ja muuttujien ja luokkien nimiä voisi hieroa

Vakavammat

Reseptien ja muiden tiedostojen parsinta

- Monien latausfunktioiden ja parsimien virhealttiutta ei ole testattu tarpeeksi. Tulisi kirjoittaa enemmän yksikkötestejä, joiden avulla löytäisin bugeja, joita varmasti on.

- Hakualgoritmi järjestää (sort) valmistettavat ruoat puuttuvien ainesosien mukaan. Käytännössä kuitenkin ainesosien “puutteen määrän” tulisi painottaa tätä. Esimerkiksi suolan tai pippurin puuttuminen ei ole niin vakavaa, kuin kilon lihaviipaleen.

- Hakunopeus ei ole ollut optimoinnin kohde
- Ohjelman tulisi heittää enemmän poikkeuksia (exceptions) ja käsitellä niitä vielä taitavammin. Nyt kuitenkin käytetään jonkin verran, mutta koodi voisi olla selvempää.
- Kappalemäärän muunokset puuttuvat. 2kpl sipuli ← → 50g sipulia?

Hyvää ja Huonoa

Hyviä ominaisuuksia

- Muunnokset yksiköiden välillä sujuvat helposti. Makarooneja voi ostaa 1kg tai 400g. Toisaalta reseptissä voi tulla vastaan tilavuusyksiköitä kuten että makarooneja tarvitaan “2dl”. Ohjelma osaa tehdä muutokset näiden välillä (ks. Algoritmit, erotus)

- Ohjelma osaa tulkita ruoka-aineiden nimiä. Näin ollen ohjelma ymmärtää, että *sipuli* ja *sipuleita* ovat sama asia.

Heikkouksia

- Ohjelman käyttöliittymän tekemiseen ei panostettu tarpeeksi, koska resurssit laitettiin sisäisten funktioiden tekemiseen ja testaamiseen. Näin ollen lopullinen käyttöliittymä on valitettavan karu. Graafisen käyttöliittymän teko aloitettiin, mutta se ei valmistunut ja pitäisi tehdä loppuun.

  Väitän että käyttöliittymän rajoittuneisuus johtuu siitä, että allkirjoittanut ei ole koskaan käyttnyt tosissaan mitään tekstipohjaista järjestelmää, eikä siksi osaa suunnitella hyvää sellaista.

- Ohjelman hakufunktiot eivät kerro tarpeeksi, vaikka osaisivat. Käytännössä ainoa “tarvittava” funktio on jääkaapin sisällön perusteella haku. Tulosteissa on kuitenkin parantelemista (esimerkiksi että ohjelma ilmoittaisi aineen kuuluvan alireseptiin).

  Funktiot myös ilmoittavat kun käytetään alireseptiä, mutta tämän ja muiden ominaisuuksien demonstroimiseen olisi pitänyt panostaa enemmän. Nyt assistentilla voi olla vaikea nähdä mitä kaikkea ohjelma tekee pellin alla.

- Yksiköiden muuntamisfunktiota voi aina kehittää. Miten verrataan 2kpl sipuli ← → 50g sipulia? Miten ulkomaiset mitat kuten cup ja oz?

Lisäksi lisää määreitä pitäisi koodata ohjelmaan tai ne pitäisi ladata tiedostosta (ks. Algoritmit, erotus)

-

Ongelmat ja poikkeamat suunnitelmasta

Allergiatiedot piti suunnitelman mukaan pystyä määrittelemään erillisessä tiedostossa, sekä jääkaappitiedostossa. Ongelmaksi muodostui kuitenkin se että allergiatiedostossa aineen nimi voi olla erilailla taivutettu kuin reseptissä. Implementointiin ei jäänyt aikaa

Myös jääkaapin ja reseptien hallinta oli suunnitteilla. Tästä kuitenkin luovuin aloituskappaleessa mainituista syistä - on turhaa että asioita voi muokata kahdesta paikkaa.

Graafinen käyttöliittymä myös jäi puolitiehen ajan takia.

Arvio lopputuloksesta

Lopputuloksena on hieno ohjelma, jota käytetään tekstikäyttöliittymän kautta. Ohjelma osaa kaikki keskeiset kikat (kuten vaadittu tehtävänannossa) ja soveltuu jo tällaisenaan käytettäväksi. Tuotteiden hallinnointi pitää kuitenkin suorittaa teksititedostojen kautta.

Ohjelmoinnin osalta koodia olisi voinut parannella ja esimerkkitiedostoja olla lisää. Myös testejä tulee kehittää, vaikka niitä on jo nyt miltei kaikille funktioille.

Olen tyytyväinen algoritmien toimintaan ja yksiköiden käsittelyyn. Sanojen samankaltaisuuden kautta “oikeiden” reseptien käyttö ohjelman kanssa onnistuu miltei suoraan paketista. Loppujen lopuksi “helpoiksi” määriteltyjen toimintojen toteuttamiseen meni paljon aikaa, eivätkä ne olleet triviaaleja.

Tulevaisuudessa ohjelmaan voisi lisätä graafisen käyttöliittymän. Uskallan kuitenkin väittää, että nykyinen projekti on keskivertoa huolellisemmin tehty (ks. git logit viitteissä). Ohjelmoitaessa ratkottiin monia ongelmia ja käytettiin kurssilta opittua.Kehitys jaottui pitkälle aikavälille, eikä hommia jätetty viimetippaan (vaikka aika loppuikin). Koodin seasta löytyy helmiäkin ja loppujen lopuksi ohjelma tekee juuri sen mitä tarvitsee.

Sanoisin että ainakin 4/5 suoritus.

Viitteet

White, Simon. How to Strike a Match. **Artikkeli sanojen vertailusta.**

http://www.catalysoft.com/articles/strikeamatch.html

Keittotaito.com. Mittaa oikein. **Artikkeli tavallisten ruoka-aineiden ominaisuuksista.**
http://www.keittotaito.com/mittaa_oikein.html
Git logit.
https://git.niksula.hut.fi/roiskop1/cheap-ass-recipes/commits/master

Liitteet

Ei liitteitä