CT10A0013 Ohjelmointi Pythonilla

L13: Käyttöliittymät ja dokumentointi

Uolevi Nikula

Päivän asiat

- Teoria: Dokumentoinnista
- Käytäntö
  - Perusohjelman toiminnot ja arkkitehtuuri
  - Graafinen käyttöliittymä, GUI
  - Komentorivikäyttöliittymä
- Lopuksi

Teoria: Dokumentoinnista

Laajan ohjelman dokumentointi

Miksi tehdä dokumentteja

Laajan ohjelman dokumentointi

- Erityisesti laajoissa ohjelmissa on **useita erilaisia tarpeita**, joten eri toiminnallisuuksia sisältävät osat toteutetaan tyypillisesti erillisinä moduuleina, esimerkiksi
  - Käyttöliittymä – keskittyy käyttäjän ymmärtämään maailmaan ja terminologiaa
  - Toiminnallinen yksikkö – varsinaiset laskentasäännöt yms. kaavat ja toiminnallisuus, *businesslogiikka*
  - Tietovarasto – tiedon varastointiin liittyvät toiminnot
- Ohjelmassa voi olla myös liittymiä muihin laitteisiin tai ohjelmistoihin, jolloin ne toteutetaan yhdessä tai useissa moduuleissa
- Toteuttamalla erilaisia tarpeita – ja esitysmuotoja – tarvitsevat osuudet eri moduuleissa, saadaan isoista ohjelmista paremmin hallittavia, siirrettäviä ja muokattavia muutosten yhteydessä

Miksi tehdä dokumentteja?

- Lähes kaikkiin töihin liittyy nykyään dokumentointi
  - Projektien rahoitus perustuu suunnitelmiin
  - Dokumentit ovat laskutuksen/maksun peruste
- Ohjelmistoprojekteissa on ajoittain laajoja dokumentointivaatimuksia eli saatetaan tuottaa
  - **Projektidokumentaatio** projektin läpiviennistä: tavoite, resurssit, aikataulut, riskit, jne.
  - **Laatudokumentaatio**: tietoja laadunvarmistuksesta, virheistä yms. (esim. ISO-9000:een liittyen)
  - **Tuotedokumentaatio**: varsinainen ohjelmiston kuvaus
- Ohjelmien koon kasvaessa myös suunnittelun, määrittelyn ja koodin- sekä projektihallinnan tärkeys kasvaa
- Tällä kurssilla puhutaan pienen ohjelman tuotedokumentaatiosta

Miksi kuvata ohjelmia?

- Ohjelmien kuvaaminen tarkoittaa ohjelman perusperiaatteiden dokumentoimista
  - Tämä mahdollistaa toteutuksen toimivuuden ja hyvyyden arvioinnin, puutteiden huomaamisen sekä vaihtoehtoisten ratkaisujen keksimisen ja kehittämisen suunnitteluvaiheessa ennen varsinaista toteutusta
- Iso kokonaisuus kannattaa pilkkoa pienempiin osiin
  - Hajota ja hallitse
  - Kuvattaessa eri osia voidaan käyttää sekä graafisia että tekstuaalisia tekniikoita
- Ohjelmien kuvauksia tarvitsevat useat käyttäjäryhmät
  - Ohjelman loppukäyttäjä
  - Ohjelmoija
  - Ohjelman tilannut asiakas (maksaja)

Yleisiä huomioita dokumentoinnista

Dokumentointi

- Dokumentointi tarkoittaa yleisesti ottaen asioiden tallentamista
  - Dokumentti voi olla paperi, tiedosto, muistiinpanot, kuva tms.
  - Dokumentointi mahdollistaa tiedon jakamisen ja keskustelun niistä
- Kokoaa ohjelman tekniset ratkaisut yhteen tiettyihin dokumentteihin
- Käsittää ohjelman käyttö-, asennus-, ylläpito-ohjeet yms.
- Dokumentaatio pitäisi tuottaa ohjelmakehityksen yhteydessä
  - Aloittaa ennen ohjelman tekoa
  - Ylläpitää ohjelman käytöstä poistamiseen asti

Kuvaamisesta

- Kuvaamisen tavoitteena on helpottaa asian ymmärtämistä
  - havainnollistamalla käsiteltävää asiaa
  - keskittymällä oleellisiin asioihin ja välttäen yksityiskohtia eli *abstraktiotason nostamisella*
    - Mitä aliohjelmat **lueTiedosto()** ja **kirjoitaTiedosto()** tekevät?
  - mahdollistaa tiedon jakamisen, keskustelun, yhteisen ymmärryksen luomisen ja virheiden löytämisen

Huomioita kuvaustekniikoista

- Kuvaustekniikoita on paljon erilaisia
- Eri tekniikat sopivat eri vaiheisiin ja erilaisiin tehtäviin
- Mitään ehdottomia sääntöjä ei ole siitä, mitä tekniikkaa käyttää missäkin tilanteessa
- Yleensä kannattaa käyttää samanlaisia tekniikoita kuin muut osapuolet kuten työkaverit ja asiakas
  - Tekee kuvausten lukemisen ja ymmärtämisen helpommaksi muille
  - Parempia tekniikoita kannattaa ottaa käyttöön, muttei liian äkillisesti tarvittavan opiskelun takia
- Kuvaaminen on tärkeää kaikissa eri ohjelmistokehityksen vaiheissa: määrittely, suunnittelu, toteutus, testaus ja ylläpito

Eri vaiheiden dokumentointi

Määrittely, suunnittelu, toteutus, testaus, ylläpito

Ohjelman määrittely

- Määrittelyvaiheen tavoitteena **ongelman selvittäminen**
  - ongelman tarkastelu, tunnistaminen ja ymmärtäminen
  - odotettavien toimintojen selvittäminen, tiedon kerääminen ja dokumentointi, jäsentely (esim. vaatimukset, rajoitteet, olettamukset) ja analyysi
  - ongelman jakaminen pienempiin osiin
- Määrittelyvaiheessa tuotetaan asiakkaan käsitteillä ja näkökulmasta tehty vaatimusmäärittelydokumentti
- Määrittelydokumentti on keskeinen asiakkaan läpikäymä ja hyväksymä dokumentti – ainakin pitäisi olla
- Yleensä määrittelyvaiheessa käytetään tavallisia tekstinkäsittelyohjelmia ja taulukoita, ehkä esitysgrafiikkaa havainnollistamaan yksinkertaisia rakenteita
  - Määrittelyvaiheeseen erikoistuneita ohjelmistoja on olemassa
- Usein asiakkaan/käyttäjän määrittelyt ovat erittäin yleisiä ja riittämättömiä, ts. ne ovatkin käytännössä lähtökohta toteuttajan määrittelyille

Yksinkertainen määrittely

- Yksinkertaisen mutta kattavan ohjelman määrittelyn lähtökohdat ovat seuraavat:
  1. tehtävän sanallinen kuvaus
  2. lähtötietojen kuvaus
  3. tulostietojen kuvaus
  4. keskeisimmät käsittelysäännöt
  5. ongelmasta tehdyt oletukset
  6. esimerkkitapaus

Esimerkki tehtävän kuvauksesta 1

- Tehtävänä on laskea keskiarvo oppilaiden kokeesta saamista pistemääristä
  - Lyhyt sanallinen kuvaus: lasketaan luokan oppilaiden kokeessa saamien pistemäärien keskiarvo
  - Lähtötiedot
    - pistemäärät p_1, ... , p_N
    - $0 < p_i <$ maxarvo
    - N ja p_i kokonaislukuja, $N > 0$
    - maxarvo positiivinen kokonaisluku
  - Tulostustiedot
    - osallistujien lukumäärä
    - keskiarvo
    - ilmoitukset epäkelvoista pistemääristä

Esimerkki tehtävän kuvauksesta 2

- Keskeiset käsittelysäännöt
  - lasketaan pistemääriä yhteen kunnes syötettävät pistemäärät loppuvat
  - jaetaan pistemäärien summa niiden lukumäärällä
  - epäkelvoista lähtötiedoista annetaan virheilmoitus eikä niitä huomioida keskiarvoa laskettaessa
- Oletukset
  - koe on arvosteltu kokonaislukupistemäärillä, jotka eivät voi olla negatiivisia. Etukäteen ei välttämättä tiedetä, kuinka monta osallistujaa kokeessa on
- Esimerkki
  - syöttötiedot
    - maxarvo on 30
    - pistemäärät 16, 15, -12, 17, 17
  - tulostustiedot
    - virheellinen 3. pistemäärä -12
    - osallistujia 4
    - keskiarvo 16.25

Hyvän määrittelyn ominaisuuksia

- **Täydellisyys**: ongelman määrittely sisältää kaikki ongelman alku- ja lopputilan tärkeät seikat ja rajoitteet
- **Yksikäsitteisyys**: eri henkilöt eivät voi tulkita ongelman määrittelyä eri tavoin
- **Yhdenmukaisuus todellisuuden kanssa**: ongelman ratkaisu noudattaa todellisuudessa esiintyviä matemaattisia malleja, lakeja jne.
- **Eheys**: alku- ja lopputilan välillä vallitsee johdonmukaisuus, ratkaisumenetelmä sisältää ratkaisun kaikki vaiheet tarkasti kuvattuina
- **Yleisyys**: ongelman ratkaisun tulisi olla mahdollisimman yleinen, jotta se kattaisi mahdollisimman suuren kohdeongelmien esiintymien joukon

Ohjelman suunnittelu

- Suunnitteluvaihe eli ***ongelman ratkaisun*** **määrittely**
  - ratkaistaan ongelma periaatteellisella tasolla
  - toteutettavien toimintojen ja niiden edellyttämien algoritmien valinta ja kehittäminen (esim. lajittelu ja suorituskykyvaatimukset)
  - sopivien tietorakenteiden valinta (esim. muuttuja, luokka/olio, lista, tms.)
  - ratkaisun kuvaaminen siten, että sen toimivuutta voidaan arvioida
- Suunnitteludokumentteja ovat esim. arkkitehtuurisuunnitelma (moduulit ja niiden liittymät toisiinsa), rajapintamäärittelyt (parametrit ja paluuarvot), valitut algoritmit ja tietorakenteet
- Yleensä asiakkaalla ei ole kiinnostusta tai osaamista arvioida suunnitelmien hyvyyttä tms. vaan se on ohjelmoijille tarkoitettu dokumentti
- Suunnitteludokumentteja voi tehdä samoilla työkaluilla kuin määrittelyä, mutta suunnittelussa siirrytään usein edistyneiden työkalujen käyttöön

Ohjelman toteutus

- Ohjelmointivaiheessa eli koodattaessa valitut tietorakenteet ja algoritmit kirjoitetaan ohjelmointikielellä
- Ohjelmoinnin tukena on yleensä **tyyliopas**
  - Tavoitteena on, että kaikki ohjelmoijat kirjoittaisivat koodia samalla tyylillä
  - Yhteinen tyyli helpottaa muiden kirjoittamien koodien lukemista ja ohjelmoijien siirtymistä eri projektien välillä
  - Poikkeaminen tyylistä herättää epäilyksen virheestä
- Koodia kirjoitettaessa dokumentointi tarkoittaa lähinnä muuttujien ja aliohjelmien nimeämistä sekä kommentointia, tyyliopas antaa näihin yhteisesti sovittuja suuntaviivoja
  - Myös suunnitelmia ja määrittelydokumentteja saatetaan joutua tarkentamaan tai korjaamaan toteutusvaiheessa

Ohjelmakoodin dokumentointi 1

- Kuvaamisen keskeinen tavoite on helpottaa ohjelman ymmärtämistä
  - Tätä voidaan tehdä kooditasolla antamalla **kuvaavia nimiä** muuttujille ja aliohjelmille
  - Myös **tiedostojen systemaattisella nimeämisellä** voidaan vähentää muistamisen tarvetta ja helpottaa oikean tiedoston löytämistä
- Ohjelmakoodi on joskus vaikeasti ymmärrettävää
  - Tällöin koodiin voidaan laittaa kommentteja kertomaan, mitä koodilohkoissa tapahtuu
  - Aliohjelmien docstring on yksi tapa helpottaa koodaajan ja ylläpitäjän elämää
  - Muuttujien roolien tunnistaminen ja dokumentointi helpottaa koodin ymmärtämistä
  - Jokaisen rivin kommentointi ei yleensä ole hyvä idea

Ohjelmakoodin dokumentointi 2

- Ohjelmakoodiin tulisi dokumentoida myös ohjelman tekijä ja siihen vaikuttaneet henkilöt
  - Mikäli koodista ei saa selvää esim. ylläpitovaiheessa, olisi hyvä pystyä selvittämään, kuka ohjelman on tehnyt ja milloin – ja kuka on muuttanut sitä sen jälkeen
  - Tiedoston alussa on myös hyvä dokumentoida koodin omistaja (yritys) sekä muut tiedot, joista voi olla apua immateriaalioikeuksia selvitettäessä
- Yksi hyvä tapa dokumentoida koodia on lisätä siihen lisenssi-tiedot, ks. esim. avoimen lähdekoodin lisenssi – GNU General Public License: http://www.gnu.org/licenses/

Ohjelman testaus

- Testausvaiheen tavoitteena on varmistua siitä, että ohjelma toimii oikein
  - ohjelmassa on oikeat toiminnot
  - toiminnot on toteutettu oikein
  - rajoitteiden ja vaatimusten täyttyminen, esim. suorituskyky
  - virhetilanteiden käsittely
- Testausdokumentaation eli testaussuunnitelman ja -raportin perusteella voi arvioida, millä tavoin ohjelma on testattu ja miten hyvä sen laatu on
- Testaussuunnitelmat ja –raportit tehdään usein tekstinkäsittely- ja taulukkolaskentaohjelmilla
  - Myös asiaan erikoistuneita ohjelmistoja on olemassa

Ohjelman ylläpito

- Ylläpito eli ohjelman muokkaaminen käyttöönoton jälkeen
  - ylläpito tarkoittaa korjausten tekemistä ja uusien ominaisuuksien lisäämistä
  - ylläpito voi muodostaa noin 2/3 ohjelmiston elinkaarikustannuksista
  - ajan tasalla oleva dokumentaatio on ylläpidon lähtökohta – niin syötteenä kuin lopputuloksena

Kurssin tehtävien dokumentointi

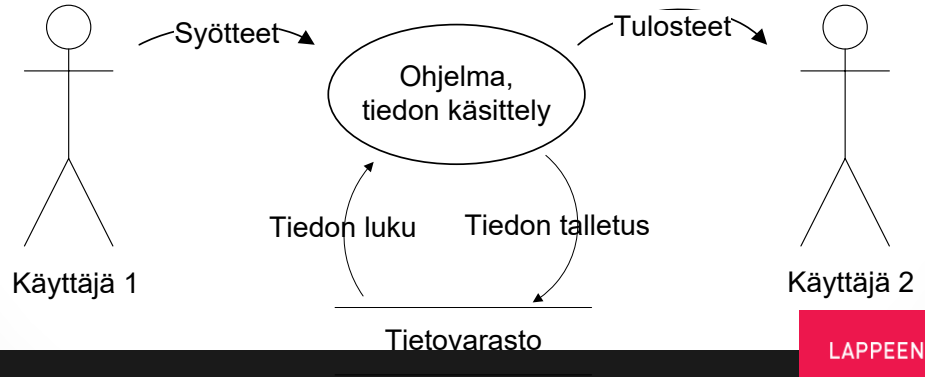
- Kaikki tämän kurssin tehtävänannot eivät noudata edellä esitettyjä käytäntöjä
  - Ohjelmointi on helpompaa, kun on valmiit määrittelyt. Ohjelmoinnin opetteluvaiheessa on kuitenkin hyvä miettiä, mitä tietoja tarvitsee ohjelman tekemiseen eli miten ohjelma tulisi määritellä, jotta sen voisi ohjelmoida. Siksi ohjelmoinnin opettelu sisältää myös tarvittavien lähtötietojen määrittelytaidon.
  - Ohjelmoijan ensimmäinen tehtävä on määritellä tehtävä riittävällä tarkkuudella
  - "Riittävä" määrittely on henkilökohtainen asia, mutta riippuu myös tilanteesta

Käytäntö

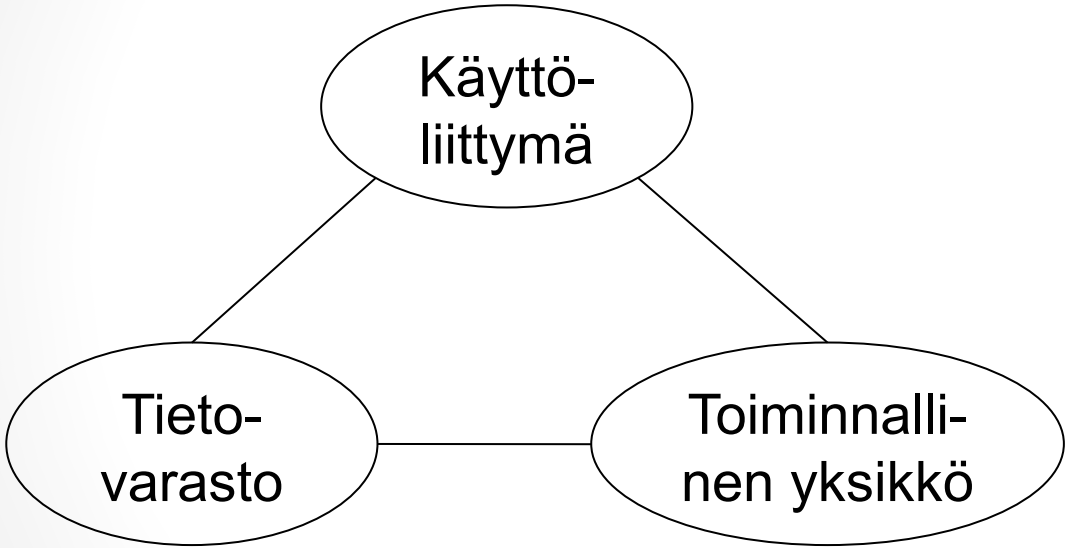
Perusohjelma
Ohjelman arkkitehtuuri
Ohjelmointivideot eli demot

Perusohjelman toiminnot (L07)

- Perusohjelma sisältää kolme perustoimintoa
  - **Käyttäjäinteraktion** eli tiedon vaihdon käyttäjän kanssa (syöttö & tulostus; 1 tai useampi käyttäjää)
  - **Tiedon käsittely** itse ohjelmassa
  - **Tiedostonkäsittely** eli tallennus tietovarastoon ja sen luku sieltä
- Nämä kaikki on käyty läpi



Ohjelman perusarkkitehtuuri



Demo 1: Graafinen käyttöliittymä

Esimerkkejä graafisesta käyttöliittymästä

- Tutustu Windowsin Laskin –ohjelmaan
  - Ohjelmassa on graafinen käyttöliittymä
  - Ohjelmalle voidaan välittää tietoa leikepöydän kautta ja samoin tulos voidaan ottaa talteen leikepöydän kautta
- Tekstinkäsittely, taulukkolaskenta, …

Taustaa graafisille käyttöliittymille

- Graafisia käyttöliittymiä ruvettiin tekemään teknologian ja osaamisen kehittymisen myötä
- Idea on, että asiat (ohjelmat, valikkokäskyt jne.) on helpompi ***tunnistaa*** kuin ***muistaa*** ulkoa
  - Huomaa, että merkkipohjaiset järjestelmät ovat usein nopeampia käyttää erityisesti asiantuntijoille – jotka muistavat käskyt ulkoa
- Nykyään tavoitteena on, että samat toiminnot näyttäisivät samanlaisilta eri ohjelmistoissa, esim. **Avaa** ja **Talleta** –dialogit jne.
  - Tämä helpottaa sekä ohjelmien käyttöä että niiden tekemistä
  - Tästä syystä käyttöliittymien suunnittelu edellyttää usein kohdeympäristön tyylioppaiden opiskelua ja noudattamista

Uudelleenkäyttö

- Huomaa, että **Avaa** ja **Talleta** –dialogit ovat itse asiassa sama dialogi eri teksteillä
  - Tämä helpottaa osaltaan ohjelmoijan työtä, koska tarvitsee osata vain yhden – joskin monipuolisemman – dialogin käyttöä kahden sijasta
  - Tämä on saatu aikaan riittävällä suunnittelutyöllä
- Tehokas ohjelmointityö edellyttää valmiiden komponenttien hyvää tuntemusta
  - Näin vältetään tarve keksiä pyörä uudestaan
  - Komponenttien uudelleenkäyttö nostaa niiden vaatiman suunnittelun ja testaamisen tasoa

Ohjelman toteutusmalli

- Graafiset käyttöliittymät edellyttävät tietyn ohjelmointimallin noudattamista
- MS-Windowsissa ohjelmointimalli on seuraava:
  1. Toteuta haluttu toiminnallisuus aliohjelmina (esim. kirjoitaTiedosto, lueTiedosto)
  2. Lisää valikoihin toteutettujen toiminnallisuuksien nimet valintoina ja/tai ikkunan nappeina
  3. Yhdistä nappi/valikon valinta ja siihen liittyvä toiminnallisuus eli aliohjelma sopivalla käskyllä
  4. Käynnistä valmis ohjelma, joka seuraa näytön tapahtumia ja kutsuu tehtyjä aliohjelmia tapahtumien perusteella (esim. *talleta-napin-painallus* –tapahtuma). Toisin sanoen, kun Windows havaitsee uuden tapahtuman, jokainen sovellus tarkistaa vuorollaan oman ikisilmukan avulla, pitäisikö sen tehdä jotain ko. tapahtuman suhteen

Oma graafinen Windows ohjelma

- Tee ohjelma, joka mahdollistaa puhelinluettelotietojen syöttämisen ja poistamisen sekä tiedostoon tallettamisen ja tiedostosta lukemisen graafisen käyttöliittymän avulla
- Käytä Windowsin standardidialogeja tiedoston lukemisessa ja tallettamisessa
- Lopuksi huomaa, että demo on viimeistelemätön
  - Graafinen käyttöliittymä tuo paljon joustavuutta ohjelman käyttöön ja samalla paljon uusia virhemahdollisuuksia
- Tämä ohjelma on esitelty viikon ohjelmointivideolla

Oppaita

- Mikäli Windows-ohjelmointi kiinnostaa, löytyy lisäinfoa esim. seuraavista osoitteista
  - Tutoriaaleja eli opastavat tekemisen aloittamisessa
    - http://wiki.python.org/moin/TkInter
    - http://thinkingtkinter.sourceforge.net/
    - (http://effbot.org/tkinterbook/ - ei saatavilla tällä hetkellä)
  - Referenssi-manuaali
    https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/index.html
- Suomenkielisiä Python-oppaita löytyy osoitteesta
  - https://wiki.python.org/moin/FinnishLanguage

Demo 2: Komentorivikäyttöliittymä

Tiedon välittäminen ohjelmalle

- Tähän mennessä ohjelmamme ovat saaneet tietoa käyttäjältä kahdella tavalla
  - Kysymällä käyttäjältä ohjelman suorituksen aikana
  - Lukemalla tietovarastosta (tekstitiedostosta)
- Kolmas yleinen tapa tiedonvälitykseen on komentoriviparametrien käyttö
  - Ohjelman käynnistämisen yhteydessä sille annetaan tietoa argumentteina komentorivillä
  - Tämä on normaali toimintamalli merkkipohjaisilla ohjelmilla erityisesti unix-puolella, ns. putki

Komentoriviparametri-ohjelma

- Tee ohjelma, joka tulostaa kaikki komentorivillä annetut parametrit

- Varmista ensin, että Python-hakemistosi on käyttöjärjestelmän polku-muuttujassa
  - Kirjoita komentoriville ”python --version” – jos tulee versionumero, niin homma on kunnossa, jos ei tule, niin Python ei ole asennettu L01 ohjeiden mukaisesti
  - Voit etsiä, esim. Windows Explorerin Search’llä, oman `python.exe` –tiedostosi kansion ja lisätä polkuun – tai asentaa Pythonin uudestaan

Komentorivi-laskin -ohjelma

- Tee komentorivi-ohjelma, joka laskee ohjelman nimen jälkeisen parametrin ilmoittaman laskutoimituksen kahden seuraavan operandin välillä, esim. komentorivi

```
laske + 3 5
```

tulostaisi lukujen 3 ja 5 summan komentorivi-ikkunaan

Komentorivi-laskin, huomioita

- Komentorivi on yksi tapa käynnistää ja välittää tietoa ohjelmalle
  - Jos yksi ohjelma tulostaa näytölle tietoja, voi toinen ohjelma lukea käyttäjän syöttämää tietoa ”näppäimistöltä”/näytöltä – unixin *putki*
- Graafinen käyttöliittymä on toinen yhä yleisempi tapa käynnistää ohjelma ja syöttää siihen tietoa sekä nähdä ohjelman tulokset
- Erilaisille käyttöliittymille on erilaisia käyttötarkoituksia ja ne asettavat erilaisia vaatimuksia sekä käyttöympäristölle että ohjelmien toteutukselle. Yleisesti ottaen graafiset käyttöliittymät ovat komentorivi-ohjelmiin verrattuna
  - työläitä toteuttaa, vaativat paljon osaamista ja aikaa
  - vaateliaita käyttöjärjestelmän suhteen eli vaativat paljon resursseja
  - käyttäjäystävällisempiä oikein toteutettuina

Lopuksi

Osaamistavoitteet

Osaamistavoitteet

- Perusymmärrys seuraavista asioista
  - Ohjelmaan liittyvät **dokumentit** kuten projekti-, laatu-, tuote-; määrittely-, suunnittelu-, toteutus-, testaus-; käyttö-, asennus- ja ylläpitodokumentti
  - **Valikkopohjainen ohjelma**. Tällä kurssilla tyypillinen ohjelma, jonka käyttöliittymä perustuu kirjaimiin eli ASCII-merkkeihin ilman grafiikkaa
  - **Komentoriviohjelma**. Unixissa tyypillinen ohjelma, jossa ohjelman nimen perään kirjoitetaan parametrit komentoriville
  - **Graafinen käyttöliittymä**. Visuaalisiin elementteihin perustuva käyttöliittymä ja Windowsin tapa toteuttaa niitä ikisilmukan avulla
  - **Ohjelman määrittelyn lähtökohdat**. Tehtävän sanallinen kuvaus, lähtötiedot, tulostiedot, keskeisimmät käsittelysäännöt, olettamukset ja esimerkki
- Tentissä riittää osata tällä kurssilla tyypillisen valikkopohjaisen ohjelman toteutus, mikä on tehty esim. harjoitustyössä

Täydennyksiä oppaan lukuun 13

Ei tyyliohjeita pienille Python-ohjelmille

Oppaan esimerkit ja käsitellyt asiat

Käsitellyt asiat oppaan luvussa 13

- Graafinen käyttöliittymä: Esimerkki 13.1, 13.2
- Komentoriviparametrit: Esimerkki 13.3
- Käyttöliittymät ja graafisten käyttöliittymien perusteet
- Komentorivikäyttöliittymä ja komentoriviparametrit