CT10A0013 Ohjelmointi Pythonilla

L12: Tiedon esitysmuodoista

Uolevi Nikula

Päivän asiat

- Teoria: Tiedon esitysmuodoista
- Käytäntö
- Lopuksi

Teoria

Tiedon esitysmuodoista

Aiemmilla luennoilla auki jääneitä asioita

- Bitti – mikä se on?
- Merkkijonojen lajittelu – mihin se perustuu?
- Binaaritiedosto – mikä se on?
- Tiedosto = open(Nimi, “r”, encoding=“utf-8”)
  - Mikä on encoding?
- Katso myös Ohjelmointiopas ja seuraavien asioiden kuvaukset:
  - Liukuluku – miksi liukuluvut eivät ole aina tarkkoja?
  - Pyöristäminen – miksi 0.5 ei olekaan aina 1?

Tieto tietokoneohjelmassa

- Ohjelmoija hallitsee tietoa ohjelmassa muuttujien avulla
- Muuttujat voivat saada erilaisia arvoja
  - Lukuja
  - Merkkijonoja
  - Totuusarvoja
  - Erityisiä tietorakenteita kuten tiedostokahva, lista ja olio

Merkit, merkistö ja kooditaulukot

Sama lähtökohta näyttää erilaiselta eri kooditaulukoilla eli merkistöissä

ASCII-merkistö

- Tietokoneiden kirjainmerkit perustuvat ASCII-merkistöön (ts. kirjaimet ja numerot)
  - Yksi merkki on koodattuna 8 bitillä, alkuaikona 7 bitillä
- ASCII-kooditaulukko löytyy esim. osoitteesta
  - http://fi.wikipedia.org/wiki/ASCII
  - Sivu sisältää myös tarkempaa tietoa ASCII-koodista yms.
- Merkkijonot järjestetään merkkien ASCII-arvojen perusteella
  - Merkkien lajittelun lähtökohta on niiden paikka ASCII-taulukossa. Toisin sanoen esim. A:n ja B:n “aakkosjärjestys” tulee teknisesti niiden paikoista ASCII-taulukossa eli indeksit 65 ja 66. 65 < 66 joten A on “aakkosjärjestyksessä” ennen B:tä samoin kuin A on ennen a:ta eli 65 < 97
  - Erikoismerkkien (esim. skandinaaviset kirjaimet) aakkosjärjestys saattaa olla outo (”väärä”) riippuen niiden sijainnista kooditaulukossa
  - Katso myös MS-Word/PowerPoint ja *Lisää symboli* –käsky

Kooditaulukoita

- ASCII-taulukko ei ole ainoa käytössä oleva taulukko numeroiden muuttamiseen merkeiksi, ks. esim. alla olevia sivuja
  - http://www.kostis.net/charsets/
  - The Python Standard Library | 7. Binary Data Services | 7.2.4. Python Specific Encodings
  - Python HOWTOs: Unicode HOWTO
- Jotta ohjelman käyttäjä näkisi merkit sellaisina kuin tarkoitus on, pitää jokaisessa kooditaulukon valintavaiheessa tehdä sama valinta
  - Vrt. esim. sähköpostiviestit ja tämän kurssin ohjelmat Windows ympäristössä ja toisaalta komentorivillä

Huomioita kooditaulukoista

- Tiedostoa lukiessa pitää tietää, miten merkit ovat koodattu
- Kooditaulukoilla saadaan ohjelma yleensä toimimaan oikein yhdessä ympäristössä
  - Saman ohjelman toimiminen useissa eri ympäristöissä ei ole välttämätöntä peruskurssilla
  - Perehtymällä kooditaulukoihin ja niiden käsittelyyn tämäkin onnistuu
- Mikäli saat sähköpostissa viestin, jonka merkeissä ei ole järkeä, voi viestistä puuttua käytetyn merkistön tiedot. Tällöin voi olla helpompaa pyytää viesti uudestaan merkkitaulukon selvittämisen sijasta
- Mikäli Python-ohjelmasi ei osaa lukea skandinaavisia merkkejä tekstitiedostosta, voi avaamisen yhteydessä määritellä käytetyn koodauksen
  - `Tiedosto = open(Tiedostonimi, "r", encoding = "utf-8")`

Numerot ja niiden eri esitysmuodot

Sama luku näyttää erilaiselta eri kantaluvuilla

Numeroiden esitysmuotoja

- Saman luvun voi tulostaa eri muodoissa, esim. desimaali, oktaali ja heksadesimaali muodoissa
  - 1-1-1; 8-10-8; 10-12-A; 15-17-F; 16-20-10
- Huomaa
  - Numeron 2 potenssit ovat kiinnostavia tietokoneohjelmien kannalta, koska ne kuvaavat muistin tiloja – 0 ja 1 (onko johdossa tai muistipaikassa jännite vai ei)
  - Tietokoneen sananleveys on normaalisti kahden potensseja: 2, 4, 8, 16, 32, … eli 2^2, 2^3, 2^4, 2^5, …
  - Tietokoneisiin liittyvät ratkaisut ovat tyypillisesti X-bittisiä, esim. salausalgoritmit (32, 64, 128, 256, 512, 1024, …)

Kantalukumuunnokset 1

- Kantalukumuunnoksia tehdään yleensä binaari-, desimaali-, oktaali- ja heksadesimaalilukujen välillä
- Binaari-, oktaali- ja heksadesimaalimuunnokset ovat suoraviivaisia, koska ne kaikki perustuvat kahden potensseihin (2^n)
- Huomaa, että kantaluku tarkoittaa aina sitä, kuinka monta erilaista arvoa luku voi saada, esim.
  - 2 (=2^1) : luvut 0 ja 1 mahdollisia arvoja
  - 8 (=2^3) : luvut 0, 1, 2, 3, 4, 5, 6 ja 7 mahdollisia
  - 16 (=2^4) : luvut 0, 1, …, 9, a, b, c, d, e ja f mahdollisia
  - 10: luvut 0, 1, …, 7, 8, 9 mahdollisia – ei perustu 2-potensseihin!

Kantalukumuunnokset 2

- Kun kaikki erilaiset arvot on käyty läpi, siirrytään seuraavaan numeroon eli esim. kymmenjärjestelmässä numeron 9 jälkeen tulee 10 eli (lue oikealta vasemmalle, ks. 10 potensseja!)
  - $1 * 10^1 + 0 * 10^0 = 10$
  - $1 * 10^2 + 0 * 10^1 + 1 * 10^0 = 101$
- Vastaavasti binaariluvuissa tarvitaan kolme numeroa, kun kaksi ei riitä kuvamaan arvoa, esim.
  - $4_{10} = 1 * 2^2 + 0 * 2^1 + 0 * 2^0 = 100_2$

Lisätietoa Internetistä

- Kantaluvuista ja niiden muunnoksista löytyy lisäinfoa esim. seuraavilta sivuilta
- Binaariluvut
  - http://fi.wikipedia.org/wiki/Bin%C3%A4%C3%A4rij%C3%A4rjestelm%C3%A4
- Oktaaliluvut
  - http://fi.wikipedia.org/wiki/Oktaalij%C3%A4rjestelm%C3%A4
- Kymmenjärjestelmän luvut
  - http://fi.wikipedia.org/wiki/Kymmenj%C3%A4rjestelm%C3%A4
- Heksadesimaaliluvut
  - http://fi.wikipedia.org/wiki/Heksadesimaalij%C3%A4rjestelm%C3%A4

Bittioperaatioista

- Koska tietokone tallettaa tietoa binaarikoodina, on monissa ohjelmointikielissä olemassa myös bittioperaatioita
- Yleisimmät bittioperaatiot Pythonissa ovat
  - bitwise AND eli &
  - bitwise OR eli |
  - bittijonon XOR eli poissulkeva or eli ^
  - bittijonon kääntö eli ~
- Lisäksi bittien siirtoon on olemassa operaattorit
  - siirto oikealla >>
  - siirto vasemmalla <<
- Bittioperaatioita käytetään yleensä maskien kanssa selvittämään tietyn bitin arvoa, bittimaski voisi olla esim. 0100 eli 3. bitti oikealta päällä
  - Onko 3. bitti oikealta päällä voitaisiin siis selvittää ottamalle **bitwise and** –operaatio numeron 4 kanssa – (4 & tutkittava_muuttuja)

Tiedon esittäminen tietokoneohjelmissa

Esitysmuodoista

- Tietoa voidaan esittää bitteinä, lukuina ja erilaisina kirjainmerkkeinä tai symboleina
  - Numeroihin liittyy usein kantalukumuunnoksia
  - Merkkien muunnokset tehdään usein erilaisten taulukoiden avulla
- Tiedon eri esitystapoja tarvitaan erilaisten tarpeiden vuoksi
  - Tiedon säilyttäminen, käsittely ja tulkinta

Elektroniikka vs. ihmiset

- Tietokoneen muisti perustuu elektroniikkaan, joka tallentaa tietoa elektroniikan käsittelemässä muodossa – jännite-eroina
  - Jotta ohjelmat voisivat toimia, on tietokoneen hallittava ohjelmoijan ja elektroniikan erilaiset tietojen esitysmuodot ja tehtävä niiden välisiä muutoksia tarpeen mukaan
  - Laitteiston läheinen ohjelmointi tarkoittaa sitä, että ohjelmat joutuvat tulkitsemaan esim. mittalaitteiden ja anturien antamia tietoja käyttäjän ymmärtämään esitysmuotoon
- Tietokoneen käyttäjät ymmärtävät yleensä luonnollista kieltä
  - Jotta käyttäjät voivat käyttää ohjelmia, on ohjelman ohjattava käyttäjää sopivalla tavalla
  - Tietokoneohjelma joutuu usein tekemään erilaisia muunnoksia tiedon esitysmuodoissa, jotta käyttäjät voivat käyttää ohjelmaa ilman ongelmia
  - Myös eri ohjelmat käsittelevät tietoa hyvin erilaisissa muodoissa, esim. tekstinä, taulukoina, esitysgrafiikkana ja valokuvina

Tietoa massamuistilla

- Tieto talletetaan kovalevylle ykkösinä ja nollina – eli bitteinä
  - Kovalevyn pinnalla on magneettista ainetta, jota voidaan käsitellä sähköllä tiedon tallentamisvaiheessa
  - Tietoa luettaessa lukupäähän indusoituu sähköä levyn pinnassa olevien magneettivarausten mukaan
  - Jännitetasoja on kaksi eli *ei-jännitettä* (bitti on nolla) ja *jännite* (bitti on yksi)
- Lisätietoa löytyy esim. alla olevasta osoitteesta

  http://fi.wikipedia.org/wiki/Kiintolevy

Biteistä numeroiksi

- Massamuistilla olevista ykkösistä ja nollista muodostuu bittivirta esim. 0110001011110
- Bittivirta voidaan jakaa ryhmiin ja tulkita numeroina
  - Esimerkiksi kolmella bitillä voidaan esittää numeroita 0..8, neljällä bitillä 0..16, jne.
  - Normaalisti merkit koodataan 8 bitillä, joilla voidaan esittää numerot 0..255 ($2^8 = 256$)
  - Yhtä merkkiä kohden oleva bittien määrä on nk. *sanaleveys*

Numeroista merkeiksi

- Numerot muutetaan käyttäjän tai ohjelman ymmärtämiksi merkeiksi kooditaulukon avulla
  - ASCII-taulukon kehittämisellä pyrittiin ratkaisemaan numeroiden ja merkkien välinen yhteys
  - Taulukossa oli alunperin 128 merkkiä (7 bittiä), joten erikoismerkit, esim. skandinaaviset kirjaimet (å, ä, ö) ja muut kansalliset merkit (é, €, kiina ja kyrilliset kirjaimet yms.) eivät mahtuneet siihen mukaan
  - Erilaisten merkistöjen käsittelemiseen on kehitetty muita erilaisia kooditaulukoita, joiden avulla numeroita voidaan muuttaa halutuiksi merkeiksi
  - Esim. laajennettu ASCII –taulukko (256 merkkiä) voi sisältää skandinaaviset kirjainmerkit, tai kyrilliset merkit
  - UTF-8 merkistöön mahtuvat sekä skandinaaviset että kyrilliset merkit + muita merkkejä

Binaaritiedosto

- Kurssilla on tähän asti puhuttu ja käsitelty tekstitiedostoja
  - Tiedostoja voi katsella ja muokata editorilla
- Useimmat ohjelmat käyttävät tekstitiedostojen sijaan binaaritiedostoja
  - Tiedot on tällöin koodatussa muodossa, joten tiedoston katselu onnistuu yleensä vain ko. ohjelmalla
  - Binaaritiedostoja ei yleensä voi/saa muokata suoraan
  - Binaaritiedostojen katseluun tarvitaan yleensä ”oikeaa” editoria eli binaarieditoria
- Esimerkkejä binaaritiedostoista ovat mm. Word (.doc), Excel (.xls) ja PDF (.pdf) tiedostot
  - Yleensä kaupalliset ohjelmat käyttävät binaaritiedostoja, koska ne ovat tekstitiedostoja nopeampia käsitellä, sisältävät enemmän ja tarkempaa tietoa, estävät tiedostojen rikkomisen ”vahingossa” jne.

Yhteenveto

- Kovalevyllä on **jännite-eroja**, jotka voidaan tulkita **bitteinä** 0 tai 1
- Biteistä voidaan muodostaa **numeroita**, esim. 8 bitillä voidaan esittää numerot 0-255
- Numeroita voidaan tulkita eri tavoin kooditaulukoiden avulla. Tyypillisesti käytetään **ASCII-taulukkoa**, jolloin numeroista saadaan **ASCII-merkkejä**. Taulukon kirjainmerkeistä muodostuu **sanoja** ja lauseita
- Ilman kooditaulukkoa bittivirran, esim. sähköpostin, tulkinta vastaa salakielisen viestin tulkintaa ilman koodiavainta
  - Oikea kooditaulukko helpottaa elämää
- Ohjelmoitaessa joudutaan ajoittain paneutumaan merkkitaulukoihin käyttäjien ongelmien välttämiseksi
  - Ongelma on merkittävä maailmanlaajuisesti jaeltavissa ohjelmissa
  - Yksi yritys ongelman ratkaisemiseksi on unicode standardi, esim. UTF-8 (www.unicode.org)

Lopuksi

Osaamistavoitteet

Osaamistavoitteet – ks. ohjelmointivideot

- Perusymmärrys siitä, mitä seuraavat ovat ja mihin ne liittyvät, muttei tarvitse pystyä laskemaan tai toteuttamaan näitä asioita esim. tentissä
  - bitti, kooditaulukot kuten ASCII-taulukko, binaaritiedosto
  - binaari-, oktaali-, desimaali- ja heksadesimaaliluvut sekä niiden muunnokset
  - käyttöliittymien eri kielivaihtoehdot ja miten ne toteutetaan ohjelmoijan näkökulmasta

- Huom. Oppaan luvussa 12 käsitellään myös lukujen pyöristämistä Pythonissa ja ajan esitystapaa tietokoneessa.

Täydennyksiä oppaan lukuun 12

Ei tyyliohjeita pienille Python-ohjelmille

Oppaan esimerkit ja käsitellyt asiat

Käsitellyt asiat oppaan luvussa 12

- Binaariluvuilla laskeminen: Esimerkki 12.1, 12.2
- Merkkitaulukot: Taulukko 12.1/ASCII
- Binaaritiedostot ja pickle: Esimerkki 12.3, 12.4
- Pyöristäminen Pythonissa
- Ajan esitystapa tietokoneessa