CT10A0013 Ohjelmointi Pythonilla

L02: Perusohjelma

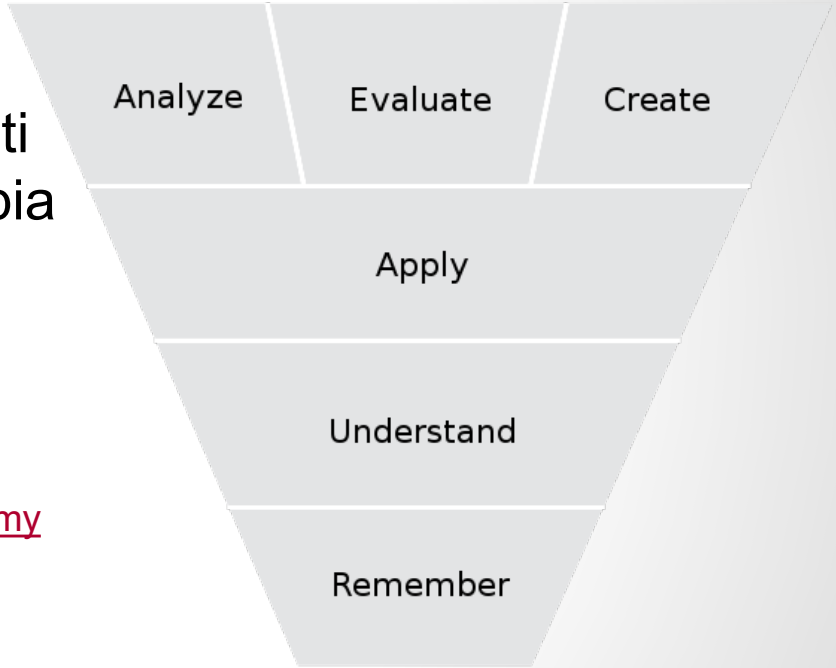
Uolevi Nikula

Päivän asiat

- Osaamistavoitteista yleisesti
- Teoria
  - Historiaa, ohjelmointikieliä, käsitteitä
- Käytäntö
  - Perusohjelman toteutus
  - Perusohjelman rakenne
  - Muuttujista ja niiden rooleista
  - Merkkijonot, yhdistely ja leikkaukset
  - Tulostuksen muotoilu, print ja parametrit
  - Tietotyypeistä
- Lopuksi
- Täydennyksiä oppaan lukuihin 1 ja 2

Osaamistavoitteista yleisesti

- Osaamistavoitteita on erilaisia ja eri tasoisia, mutta lähtökohtaisesti peruskurssien tavoitteena on oppia
  1. muistamaan pääasiat
  2. ymmärtämään pääasiat
  3. soveltamaan osaamista käytäntöön
- Osaamistavoitteista laajemmin
  - http://en.wikipedia.org/wiki/Bloom's_Taxonomy

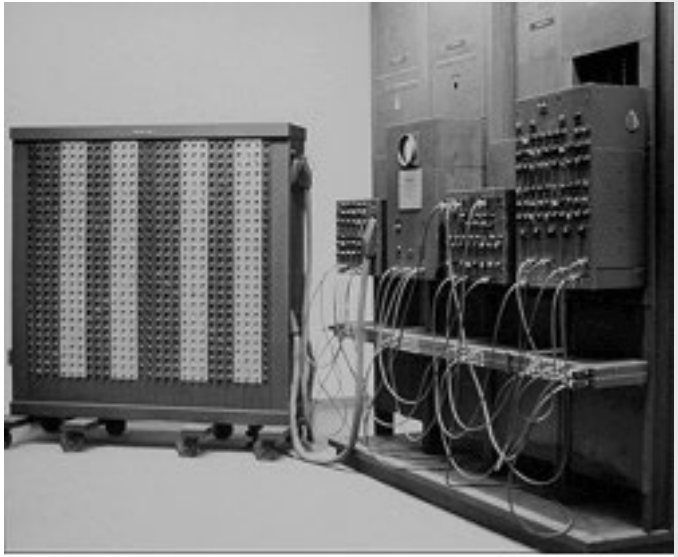


Teoria

Historiaa, ohjelmointikieliä, käsitteitä

Ohjelmoinnin alku

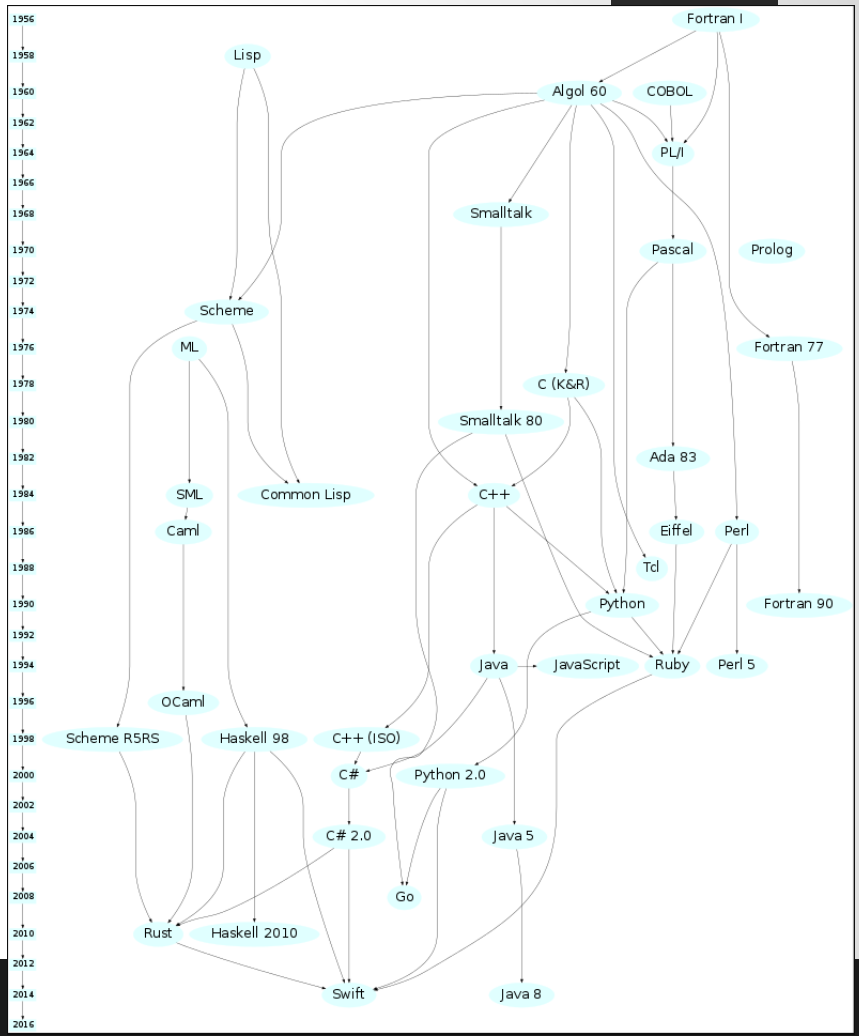
- ENIAC-tietokone, julkaistiin 14.2.1946
- Tietokone oli huoneen kokoinen
- Ohjelmointi tapahtui konekielellä (0 ja 1)
- The ENIAC Museum Online http://www.seas.upenn.edu/about-seas/eniac/index.php



The ENIAC Today

Ohjelmointikielten historia

- Kuva on webistä
  - http://merd.sourceforge.net/pixel/language-study/diagram.html
- Tarkempi kuva löytyy osoitteesta
  - http://www.levenez.com/lang/
- Online Historical Encyclopaedia of Programming Languages
  - 8945 programming languages



Ohjelmointikielten sukupolvet 1/3

- **1GL (1st generation language):** käskyt ja data tietokoneen prosessorin ymmärtämässä muodossa eli konekielellä
- **2GL:** assembler tai assembly –kielet, käskyt ovat tyypillisesti muotoa

  ADD 12, 8
- Ensimmäisen ja toisen sukupolven kieliä kutsutaan ***matalan tason kieliksi***, koska
  - niitä käyttäessä pitää ymmärtää tietokoneen rakennetta
  - ohjelmaa ei voi siirtää suoraan tietokoneesta toiseen

Ohjelmointikielten sukupolvet 2/3

- **3GL:** korkean tason kielet, esim. PL/I, C, Java ja Python. Kääntäjä muuntaa korkean tason ohjelmointikielen lauseet konekielelle; Javan yhteydessä tulosta kutsutaan bytecode:ksi, josta kohdeympäristön Java-virtuaalikone tekee 1GL:ää. Esimerkiksi Java voi näyttää seuraavalta:

```
public boolean handleEvent (Event evt) {
   switch (evt.id) {
            case Event.ACTION_EVENT: {
                     if ("Try me" .equald(evt.arg)) {
      …
} } }
```

Ohjelmointikielten sukupolvet 3/3

- **4GL:** suunniteltu olemaan lähempänä luonnollista kieltä kuin 3GL. Tietokantojen yhteydessä käytettäviä kieliä sanotaan usein 4GL kieliksi, esim.

  EXTRACT ALL CUSTOMERS WHERE "PREVIOUS PURCHASES" TOTAL MORE THAN $1000
- **5GL:** visuaalisen tai graafisen käyttöliittymän avulla tehtävää ohjelmointia, joista yleensä tehdään lähdekoodia 3GL tai 4GL kielille. Esimerkiksi Microsoft, Borland ja IBM tarjoavat visuaalisia 5GL tuotteita Java-kehitykseen.
- 3, 4 ja 5 sukupolven kielet ovat ***korkean tason kieliä***
  - Kielet eivät ole tiukasti sidoksissa koneen sisäiseen toteutukseen
  - Yleensä niitä voi siirtää tietokoneesta toiseen ja kääntää siellä
  - Lähde: http://whatis.techtarget.com/definition/0,,sid9_gci211502,00.html

Ohjelma, ohjelmisto ja tietojärjestelmä

- Ohjelmoinnin ja ohjelmistotuotannon käsitteistö ei ole täysin vakiintunutta, mutta yleisesti ottaen pätevät seuraavat määritelmät
  - *Ohjelma*: suoritettava tiedosto (esim. nimi.py)
  - *Ohjelmisto*: ohjelma + dokumentaatio + apuohjelmat yms. (esim. asennusohjelma)
  - *Tietojärjestelmä*: ohjelmisto + organisaation toimintatavat yms.
  - *Järjestelmä*: tietojärjestelmä + laitteisto
- Tällä kurssilla keskitytään ohjelmiin ellei toisin mainita – siis *ohjelma*-osaan

Käytäntö

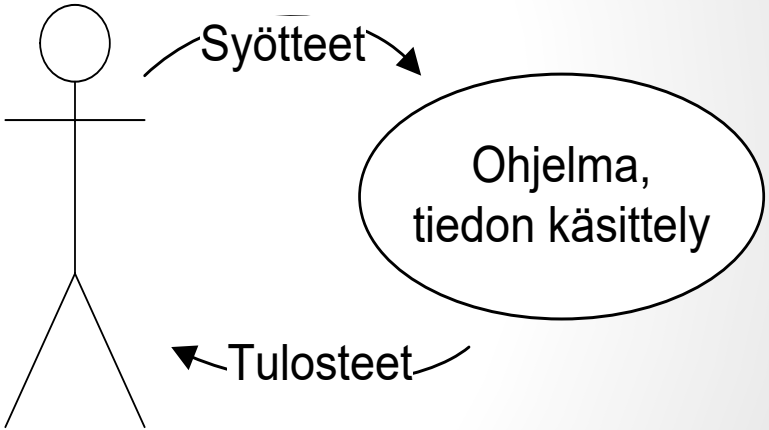
Perusohjelman toteutus
Perusohjelman rakenne
Muuttujista ja niiden rooleista
Sijoituslause
Tietotyypeistä

Perusohjelman toteutus

- Ohjelmointi on *uutta luovaa toimintaa* kuten esim. rakentaminen, ompeleminen tai maalaaminen
  - Ensin selvitetään ja **määritellään**, mitä pitäisi tehdä
  - **Suunnitellaan**, miten työ kannattaa tehdä
  - Tehdään työ valmiiksi (ehkä vaiheittain), **toteutetaan**
  - Kokeillaan toimiiko lopputulos, **testataan**
- ***Usein kannattaa aloittaa tekemällä ensin tutut asiat***
  - Kun homma etenee, ohjaa työ usein tekijää eteenpäin
  - Muutoksiin kannattaa varautua, etenkin jos aiempi kokemus on vähäistä
  - **Kokeile, tee virheitä, yritä uudestaan, opi virheistä**

Ohjelman tekemisen vaiheet

1. **Määrittele** mitä tehdään
   - Ohjelman tehtävä
   - Syötteet, mitä ohjelma kysyy käyttäjältä
   - Tulosteet, mitä ohjelma tulostaa käyttäjälle
   - Tiedon käsittely, miten saat syötteistä tulosteet
2. **Suunnittele** ohjelman rakenne, miten tehdään
3. **Toteuta** eli ohjelmoi/koodaa ohjelma
4. **Testaa** ohjelma eli varmista sen toimivuus ja oikeellisuus



Perusohjelman rakenne

- Hyvä nyrkkisääntö yksinkertaisten ohjelmien rakentamiseen on pyrkiä ryhmittelemään yhteen
  - Kiintoarvojen/muuttujien alustus
  - Tiedon lukeminen
  - Tiedon käsitteleminen
  - Tiedon tulostus

Esimerkki perusohjelman rakenteesta

```
# Alustukset
AaniNopeus = 340
####################################
# Tiedon lukeminen ja tyyppimuunnos
Syote = input("Anna kulunut aika:")
Aika = int(Syote)
####################################
# Tiedon käsitteleminen
Etaisyys = Aika * AaniNopeus
Etaisyys = Etaisyys / 1000
####################################
# Tiedon tulostus
print("Salama löi ", Etaisyys, "km:n päässä.")
print("(C) Seppo 2010")
```

Muuttujat

- Tietokoneohjelma käsittelee tietoa ohjelman suorituksen aikana
  - Käyttäjä voi antaa tietoa eli syötteen input-käskyllä
  - Ohjelma voi laskea arvoja
- Tyypillisin tapa tiedon säilyttämiseen on tiedon sijoittaminen muuttujan arvoksi, esim.
  - `Nimi = input("Anna nimi: ")`
  - `Keskiarvo = (1+3+2)/3`
- Muuttujan keskeisin tehtävä on tiedon säilyttäminen ohjelman suorituksen aikana
- Muuttujan nimi yksilöi muuttujan eli yksi muuttuja voi sisältää vain yhden arvon kerrallaan

Muuttujien nimeämissääntöjä

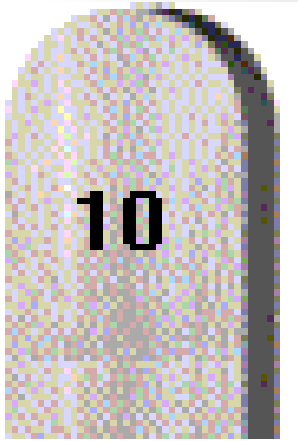
- Muuttujien nimet ovat **tunnuksia**, joille on sääntöjä
  - Alettava kirjaimella tai alaviivalla '_'
  - Nimessä voi olla kirjaimia (isoja ja pieniä) sekä numeroita (0-9)
  - Ei ääkkösiä eli skandinaavisia merkkejä eikä erikoismerkkejä
    - (Python hyväksyy, muut kielet ei hyväksy eli älä käytä)
  - Isot ja pienet kirjaimet eri asia
    - Pythonissa tunnukset "Talo" ja "talo" ovat kaksi eri asiaa kuten puhekielessä "talo" ja "valo”

Muuttujien roolit

- Muuttujilla on erilaisia rooleja ohjelmoinnissa, esim.
  - Kiintoarvo
  - Tuoreimman säilyttäjä
  - Tilapäissäilö
- Rooleja on 11 ja niistä muutamiin yleisimpiin palataan myöhemmin kurssilla. Tarkempaa tietoa rooleista löytyy alla olevalta sivulta http://saja.kapsi.fi/var_roles/role_list.html

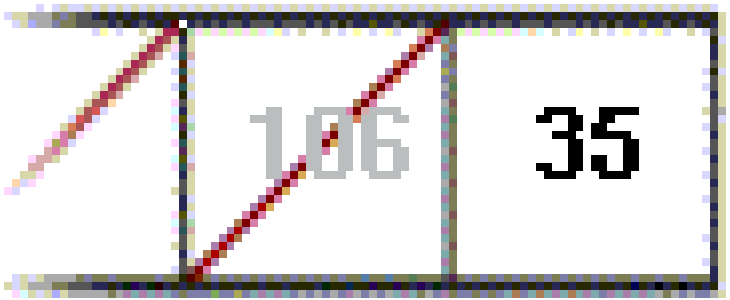
Roolit: Kiintoarvo

- Muuttuja, jonka arvoa ei muuteta sen asettamisen jälkeen, on *kiintoarvo*
- Keskeisimmät syyt kiintoarvojen käyttöön ovat
  - Ohjelman selventäminen eli ymmärrettävyys
  - Muutosten tekemisen helpottaminen
- Käyttö ei yleensä ole pakollista
- Tyypillisiä esimerkkejä ovat luonnonvakiot (pii) ja muuntokertoimet (tuuma <-> sentti)



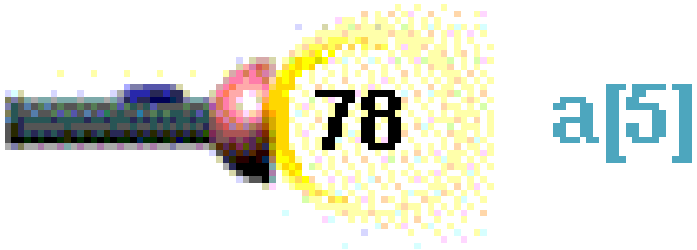
Roolit: Tuoreimman säilyttäjä

- *Tuoreimman säilyttäjä* –muuttuja sisältää viimeisimmäksi käsitellyn arvon. Tämä voi olla esim.
  - yksi alkio joukosta
  - viimeisimmäksi saatu arvo (esim. käyttäjältä tietoja kysyttäessä)

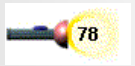


Roolit: Tilapäissäilö

- Muuttuja on tilapäissäilö, jos sen arvoa tarvitaan aina vain hyvin lyhyen aikaa. Tyypillisiä käyttötarkoituksia ovat
  - Ohjelman tehostaminen: usein tarvittavan laskutoimituksen tulos talletetaan muuttujaan turhan laskennan välttämiseksi
  - Ohjelman selventäminen: lasketaan tulos omaan muuttujaansa vaikkei tämä ole välttämättä tarpeen



Perusohjelman muuttujien roolit




```
    # Alustukset - kiintoarvo
    AaniNopeus = 340
####################################
    # Tiedon lukeminen - tuoreimman säilyttäjä
    Syote = input("Anna kulunut aika:")
    Aika = int(Syote)
####################################
    # Tiedon käsitteleminen - tilapäissäilö
    Etaisyys = Aika * AaniNopeus
    Etaisyys = Etaisyys / 1000
####################################
    # Tiedon tulostus
    print("Salama löi ", Etaisyys, "km:n päässä.")
    print("(C) Seppo 2010")
```

Sijoituslause

- Laskutoimituksia tehdään usein esim. *sijoituslauseessa*
- Sijoituslauseella vaihdetaan muuttujan arvoa
- Sijoituslause koostuu
  - *muuttujasta*
  - *operaattorista* (yhtäkuin-merkki)
  - *lausekkeesta* (esim. laskutoimitus)

Sijoituslauseen toiminta

- Esimerkiksi

```
Tuntipalkka = 12
Tuntipalkka = Tuntipalkka + 3
```

- Ensimmäinen rivi on selkeä, mutta toinen rivi
  - *Ensimmäiseksi* lasketaan oikean puolen arvo
    - Saadaan 15
  - Sitten sijoitetaan saatu arvo **`Tuntipalkka`** -muuttujaan
    - Muuttujan **`Tuntipalkka`** arvoksi tulee 15

Merkkijonojen muodostaminen

- Merkkijonoja käytetään paljon ohjelmoinnissa, esim. Pythonissa syöte näppäimistöltä on aina merkkijono
  - Uusia merkkijonoja muodostetaan *yhdistelemällä* palasia
  - Merkkijonoista erotetaan kiinnostavia osia *leikkauksilla*
- Usein tuloste muodostetaan merkkijonona, jolloin itse print-lause yksinkertaistuu, esim.
  - Nimi = input(”Anna nimi: ”)
  - Tuloste = ”Hei ” + Nimi + ”, oli kiva tavata.”
  - print(Tuloste)
- Näin pystytään määrittämään tulosteen kaikki merkit halutulla tavalla

Merkkijonoleikkaukset

- Merkkijonosta voi erottaa erilaisia osia leikkauksilla
- Leikkaukset merkitään muuttujan perään hakasuluilla ja kaksoispisteillä eli [x:y:z]
  - x: aloituspaikka (oletus 0)
  - y: lopetuspaikka, tässä olevaa merkkiä ei oteta mukaan (oletuspituus)
  - z: siirtymäväli (monenko merkin välein, oletus 1)
  - negatiivinen luku tarkoittaa käsittelyä lopusta alkuun päin
- Merkkijonon pituuden saa selvitettyä tarvittaessa len-käskyllä, len(nimi)
- Leikkauksia käytetään kun tieto on säännöllisesti muodostetuissa merkkijonoissa, esim. henkilötunnus: 010101A123B eli ppkkvv[+-A]nnnX

Tulostuksen muotoilu, print ja parametrit

- print-käskyllä voi tulostaa numeroita, muuttujia ja merkkijonoja:
  - Nimi = ”Kalle”
  - print(123, Nimi, ”Ville”)
- print-käskyä voidaan muotoilla eli räätälöidä
  - end= ’\n’ # oletusarvoisesti print-lause päättyy rivinvaihtoon eli ’\n’ –merkkiin
    - print(”Moi”, end=’.’) # print-lauseen lopussa on piste, ei rivinvaihto
  - sep=’ ’ # oletusarvoisesti kenttäerotin, separator, on välilyönti
    - print(”moi”, ”moi”, sep= ’-’) # nyt kenttien välissä on tavuviiva
- **Tulostettavan merkkijonon voi muodostaa myös halutuista merkeistä!**

Pythonin tietotyypeistä

- Pythonissa käytetään yleensä numero- (kokonais- ja desimaaliluvut) tai merkkijonomuuttujia
- **Tietotyyppiä voidaan muuttaa manuaalisesti**
  - Kokonaisluvuiksi **int**(…)
  - Desimaaliluvuksi **float**(…)
  - Merkkijonoiksi **str**(…)
- Huomaa, että int-katkaisee desimaaliosan pois ja luvun pyöristäminen pitää tehdä **round**()-funktiolla, esim. round(1.2345, 2)
- Alimerkkijonon tai yhden merkin voi erottaa merkkijonosta leikkauksena hakasuluilla, esim. Nimi[0], Nimi[0:2], Nimi[:-1]

Lopuksi

Osaamistavoitteet

Osaamistavoitteet yleisesti

- Teoria
  - Historia, ohjelmointikielien kirjo, määritelmiä
- Perusohjelman toteutus
  1. Ongelman ja ratkaisun **määrittely** – esim. tarve, syötteet, kaavat ja tulosteet
  2. Ohjelman eri vaiheiden eli algoritmin **suunnittelu**
  3. Ohjelman **toteutus**, **testaus** ja parantelu

Osaamistavoitteet ohjelmoinnissa

- Perusohjelman rakenne, huom. tulee kehittymään/muuttumaan jatkossa
  - Muuttujien/kiintoarvojen alustus
  - Tiedon lukeminen
  - Tiedon käsitteleminen
  - Tiedon tulostus
- Ohjelmointi
  - Tietotyypit: merkkijono, kokonaisluku, desimaaliluku
  - Muuttujat ja niiden roolit: kiintoarvo, tuoreimman säilyttäjä, tilapäissäilö
  - Tulostuksen muotoilu print-käskyn parametreilla sep ja end, merkkijonot ja niiden yhdistely, leikkaukset, merkkijonon pituus
  - Laskuoperaatiot ja -operaattorit

Täydennyksiä oppaan lukuihin 1 ja 2

Tyyliohjeita pienille Python-ohjelmille

Oppaan esimerkit ja käsitellyt asiat

Pienen Python-ohjelman tyyliohjeet 1

- Muuttujien nimeäminen
  - Anna muuttujalle tietoa kuvaava nimi, esim. Nimi, Ika, Pituus, Paino
  - Voit rakentaa nimen osista, esim. NimiEtu, NimiSuku, PaivamaaraAlku, PaivamaaraLoppu jne.
  - Älä käytä ääkkösiä tai mitään sanomattomia nimiä kuten a, b, c, …
  - Kirjoita kiintoarvot suuraakkosilla, esim. IKARAJA = 18, MAX_LKM = 100
- Ohjelman rakenne tiedostossa
  - Aloita määrittelyillä, esim. kiintoarvot
  - Kysy tiedot käyttäjältä, input + tyyppimuunnos + sijoitus
  - Suorita halutut operaatiot, esim. laskenta
  - Tulosta halutut tiedot käyttäjälle, print + muotoilut

Pienen Python-ohjelman tyyliohjeet 2

- Tulosteiden ulkoasua voi muokata (1) tulostuslauseessa print-käskyn muotoilumerkeillä ja (2) merkkijonon muodostamisvaiheessa
  - print-käskyn tulosta voi muokata sen parametreillä eli tietoalkioiden erotinmerkillä, sep=’ ’, ja tulosteen loppumerkillä, end=’\n’
  - Loppumerkki tulee synkronoida tulostettavan merkkijonon loppumerkin käytön kanssa
  - Merkkijonoa voi muokata muodostusvaiheessa mistä kohdasta tahansa, joten se tarjoaa tyypillisesti laajat ja joustavat muotoilumahdollisuudet, ts. merkkien ja merkkijonojen yhdistely + -merkillä

Pienen Python-ohjelman tyyliohjeet 3

- Merkkijonojen tulee olla lainausmerkkien sisällä eli " "
  - Lainausmerkkien sisällä voi käyttää heittomerkkejä eli ' '
- Yksittäiset merkit/kirjaimet, tulee laittaa heittomerkkien sisään eli ' '
- Tyyppimuunnokset
  - input-käsky palauttaa aina merkkijonon
  - Laskenta edellyttää muutosta kokonais- tai desimaaliluvuksi
  - Nämä operaatiot kannattaa tehdä kahdessa vaiheessa

```
Syote = input("Anna ikä: ")
Ika = int(Syote)
```

Käsitellyt asiat oppaan luvussa 1

- Tiedon tulostaminen: Esimerkki 1.1, 1.2, 1.7
- Tiedon kysyminen: Esimerkki 1.5, 1.6, 1.7
- Muuttujat: Esimerkki 1.3, 1.4, 1.7
- Tietotyypin muunnos: Esimerkki 1.6, 1.7
- Laskentaoperaatiot: Taulukko 1.1
- Katso oppaasta myös
  - Tyypillisiä virheilmoituksia
  - Muuttujat
  - Laskutoimitukset

Käsitellyt asiat oppaan luvussa 2

- Merkkijono ja sen muodostaminen
- Merkkijonon leikkaukset
- Lukujen tyyppimuunnokset ja pyöristys
- Tyypillisiä virheilmoituksia
- Muuttujien roolit
- Tulosteen muotoilu: Esimerkki 2.1, 2.2