Pythonia yläkouluun

SUOMEN KOODI-KOULU

Oppilaan kirja

Jussi Koivisto,
Helena Mäkinen,
Juuso Nieminen

Pythonia yläkouluun

Oppilaan kirja - Versio 1.2

Kustantaja: Suomen Koodikoulu Oy

www.codeschool.fi

ISBN: 978-952-7403-31-0

SUOMEN KOODI-KOULU

CODE SCHOOL FINLAND

Sisällys

Osa I - Ensiaskeleet Python-ohjelmointiin **5**
Aloitus - Johdatus osaan I 6
Luetaan - Muuttujat ja print-funktio 7
Ohjelmoidaan - Muuttujat ja print-funktio 8
Luetaan - Input ja str-funktiot 10
Ohjelmoidaan – Input ja str-funktiot 11
Luetaan- Datatyypit 13
Ohjelmoidaan – Datatyypit 14
Luetaan - Laskutoimitukset 16
Ohjelmoidaan – Laskutoimitukset 17
Yhteenvedon aika! 19

Osa II - Projektityö **20**
Perustetaan ensin ohjelmistoyritys 21
Koodausongelmien kohdatessa 22
Tehtäväsi on kirjoittaa koodia 22
Asiakastyö - Arvontaviesti 23
Asiakastyö - Palkkalaskuri 24
Asiakastyö - Räppinimigeneraattori 25
Asiakastyö - Onnennumero 26
Asiakastyö - Nimigeneraattori 27
Projektin loppuyhteenveto 28

LIITTEET - Python opas **29**
Ehtolause, if...else 30
for-silmukka, for loop 32
Sisäkkäiset silmukat, nested loops 34
Listat, lists 36
Omat funktiot, functions 38
Satunnaisuus, random 39
Päivämäärä, aika ja selain 41
Hyvät ohjelmointikäytännöt 43
Tiivistelmä työkaluista 44

Osa I

Ensiaskeleet Python-ohjelmointiin

Koodia on kaikkialla. Tietenkin tietokoneissa, mutta myös esimerkiksi älypuhelimissa, pesukoneissa, pankkikorteissa ja digitaalisissa termostaateissa. Koodausta tarvitaan nykyään myös monissa eri ammateissa, teitpä työtäsi sitten tieteiden, taiteiden tai talousasioiden parissa.

Tämä opas johdattaa sinut Python-nimiseen ohjelmointikieleen. Se on täydellinen aloitteleville koodareille, koska se on selkeä ja siihen löytyy paljon oppaita. Vaikka se soveltuu aloittelijoille, niin silti myös ammattikoodarit käyttävät sitä. Se on suosittu esimerkiksi tekoälyohjelmoinnissa.

Osa I johdattaa sinut Python-ohjelmoinnin perusteisiin.

Aloitus - Johdatus osaan I

Ennen Python-ohjelmointiin tutustumista asetetaan tavoitteita.

VASTAA ERILLISEEN TIEDOSTOON TAI VIHKOON JA PALAUTA VASTAUKSESI OPETTAJALLE

1. Oletko ohjelmoinut aiemmin? Jos olet, kerro minkälaisia ohjelmointitaitoja sinulla on. Mitä ohjelmointikieliä tai ohjelmointiympäristöjä olet käyttänyt? Jos et ole ohjelmoinut aiemmin, kerro minkälaisia odotuksia sinulla on Python-ohjelmoinnista.

   _____

   _____

   _____

2. Mitä kouluarvosanaa tavoittelisit, jos ohjelmoinnista annettaisiin arvosana? _____

   Miksi valitsit tämän arvosanan? Mitä muita tavoitteita sinulla on ohjelmoinnin suhteen?

   _____

   _____

   _____

3. Ohjelmointia opetellessa on tärkeää käyttää sopivia oppimisstrategioita. Mieti miten koodausta olisi hyvä opetella.

   _____

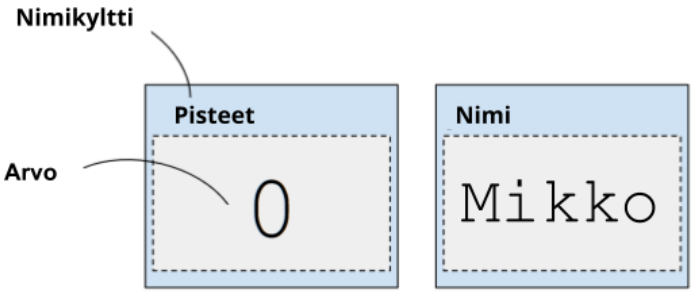
   _____

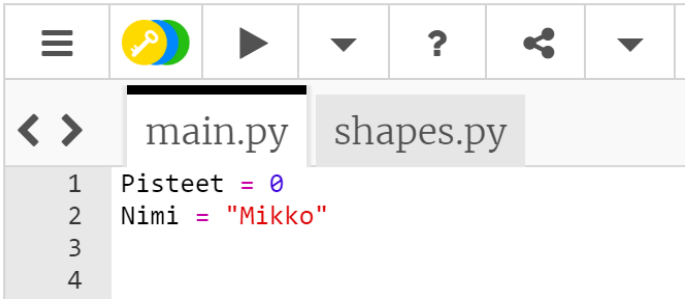
   _____

Luetaan - Muuttujat ja print-funktio

Koodatessa käytetään usein **muuttujia.** Niiden tärkeyden vuoksi tähän muuttujista kertovaan kappaleeseen kannattaa perehtyä huolella.

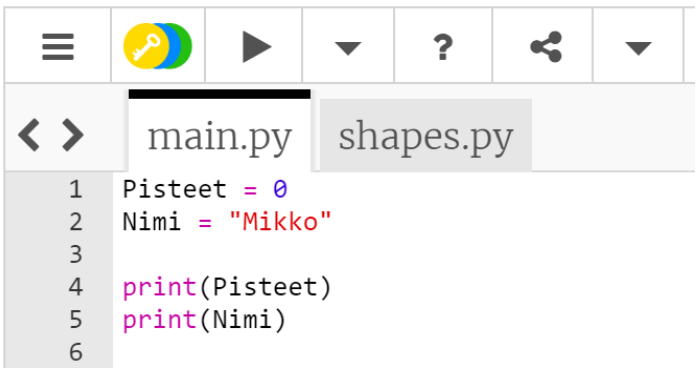
Muuttujia käytetään tiedon tallentamiseen ja nimeämiseen. Pythonissa tätä tietoa kutsutaan dataksi. Muuttujan voi ajatella olevan ikään kuin laatikko, jolla on nimikyltti. Nimikyltin avulla löydät tarvittavan datan, kun tarvitset sitä. Pelatessa saatat esimerkiksi haluta tietää kuinka paljon pisteitä on kertynyt tai kuinka monta elämää on jäljellä.





Numeroiden lisäksi muuttujiin voi tallettaa myös tekstiä. Jos talletat muuttujaan tekstiä, se pitää laittaa lainausmerkkeihin. Numeroihin ei laiteta lainausmerkkejä ympärille. Näin tietokone tietää minkä tyyppistä tietoa muuttujaan on talletettu.

Muuttuja luodaan antamalla sille havainnollinen nimi. Muuttujan nimi kertoo mitä tietoa siihen on talletettu. Talletettua tietoa kutsutaan muuttujan arvoksi. Muuttujan nimen ja sen arvon välillä on yhtäsuuruusmerkki. Näin muuttujalle asetetaan arvo.



Ennen kuin pääset kokeilemaan tätä itse, täytyy oppia käyttämään print-funktiota. Pythonissa print-funktiota käytetään tulostamaan tekstiä tai sulkeissa olevan muuttujan arvo.

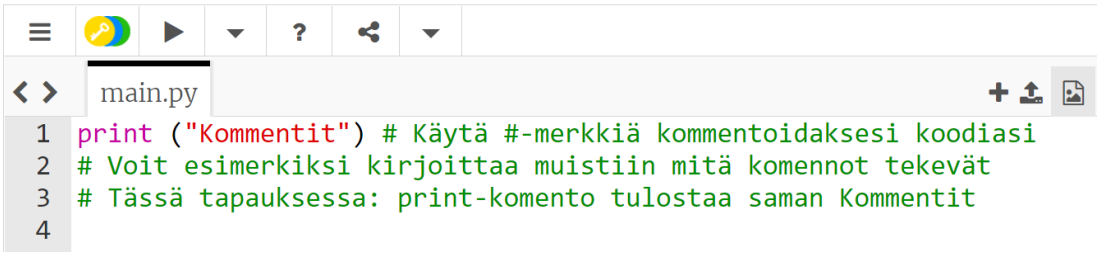
Huomaa: *Muuttujan nimessä ei voi olla välilyöntiä. Sen sijaan käytetään alaviivaa. Älä myöskään käytä muuttujien nimissä ääkkösiä (ä, ö, å).*

Ohjelmoidaan - Muuttujat ja print-funktio

Paras tapa oppia ohjelmointikieliä, tai mitään kieliä ylipäänsä, on käyttää niitä. Nyt on sinun vuorosi kirjoittaa ensimmäiset rivisi Pythonilla.

Ohjeet

Kirjoita alla oleva koodi ja suorita se jokaisen uuden rivin jälkeen nähdäksesi mitä tapahtuu. Lisää kommentti jokaisen rivin loppuun käyttämällä risuaita-merkkiä #. Selitä kommentissasi mitä kyseinen koodirivi tekee. Kommenttien lisääminen on hyödyllistä aina koodatessa. Siitä on apua itsellesi, kun palaat koodisi pariin ja myös muille, jotka lukevat koodiasi.



Kommentit eivät näy suorittaessasi koodin.

Testaa ja tutki!

```
print("Hei!") # Print() funktio tulostaa kirjoitetut merkit.
print("Hei. " + "Minun nimeni on Mikko!")

print("Yksi")
print("Kaksi")
print("Kolme") # Jokainen print-komento tulostuu omalle rivilleen.

Nimi = "Mikko"
print("Hei! Minun nimeni on " + Nimi + ".")

print(Nimi[1])
print(Nimi[0])
print(Nimi[0:2])
print(Nimi[1:3])
print(Nimi[3:])
```

Nyt on sinun vuorosi kirjoittaa koodia itse. Jatka yllä olevan koodin perään vastaukset seuraaviin haasteisiin.

Haaste!

Haaste1: Tee muuttuja nimeltä ***Sukunimi*** ja anna sille arvoksi jokin nimi. Tulosta muuttuja.

Haaste2: Kirjoita koodi, joka tulostaa sukunimen ilman sen ensimmäistä kirjainta. Jos esimerkiksi muuttujan arvo on “Järvinen”, ohjelma tulostaa “ärvinen”

Haaste3: Kirjoita koodi, joka tulostaa muuttujan ***Nimi*** arvon kolmesti samalla riville. Osaisitko myös lisätä huutomerkin perään tällä tapaa:

```
MikkoMikkoMikko!
```

Keksitkö useita erilaisia tapoja saada sama lopputulos? Kokeile erilaisia vaihtoehtoja.

Lähetä koodisi!

Onnittelut! Olet nyt kirjoittanut ensimmäiset omat koodirivisi Pythonilla! Lähetä sekä Testaa ja tutki-tehtävään että haastetehtäviin kirjoittamasi koodi opettajalla häneltä saamasi ohjeiden mukaan.

Halutessasi vertaa koodiasi ystäviesi koodin kanssa. Ratkaisivatko he haasteet eri tavalla? Koodaus on luovaa työtä! Usein samaan lopputulokseen voidaan päästä monella eri tavalla. Ehkä voit oppia jotain ystäviesi koodista?

Luetaan - Input ja str-funktiot

Tietokoneohjelmissa on usein tarpeellista pyytää ohjelman käyttäjää syöttämään siihen tietoja. Ohjelma saattaa esimerkiksi tarvita tietää käyttäjän nimen. Tällöin ohjelmassa on **muuttuja, jotka tallentaa tämän tiedon** ja lisäksi **input-funktio, jolla ohjelma pyytää käyttäjää kirjoittamaan** nimen. Kun käyttäjä on kirjoittanut nimensä, niin input-funktio liittää kyseisen arvon muuttujan sisällöksi.

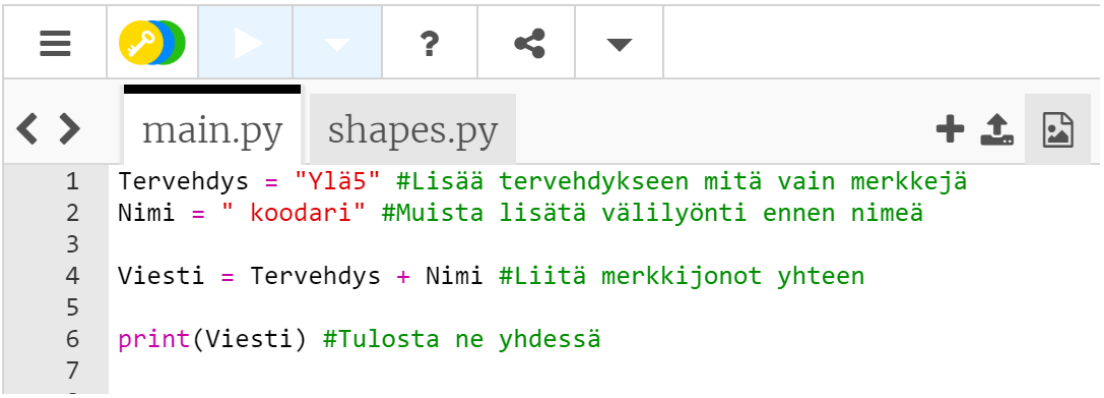
Tässä on esimerkki siitä miten ohjelma pyytää käyttäjää kirjoittamaan nimensä ja tallentaa sen sitten muuttujaan:

```
Nimi = input("Mikä sinun nimesi on?")    # Tässä "Nimi" on muuttuja
print(Nimi)     # "input" on liittänyt käyttäjän kirjoittaman nimen
# Nimi-muuttujaan ja tämä komento tulostaa sen.
```

Ohjelma suorittaa yllä olevan koodin ensimmäisen rivin siten, että se tulostaa suluissa olevan tekstin (Mikä sinun nimesi on?) ja **odottaa, kunnes käyttäjä kirjoittaa** jotain vastaukseksi ja **painaa ENTER-näppäintä.** Koodin toinen rivi tulostaa sen mitä käyttäjä on kirjoittanut.

Python tallentaa tekstiä, sanoja ja lauseita, ja lisäksi mitä vain kirjainten ja merkkien yhdistelmiä **merkkijonoina** (englanniksi **strings**). Tämä on hyvä pitää mielessä sillä Python kohtelee merkkijonoja eri tavalla kuin muuta sisältöä. Pääset toteamaan tämän pian itse harjoitusten myötä.

Kerrataan ensin mitä jo tiedämme merkkijonoista. Toistaiseksi olet oppinut, että liittäessäsi tekstiä muuttujan sisällöksi tarvitset tekstin ympärillä lainausmerkit, jotta Python tietää kyseessä olevan merkkijono. Lisäksi muistat, että voit liittää merkkijonoja toisiinsa **+ merkillä**. Katso tästä esimerkkinä alla oleva koodi.



Tämä koodi tulostaa "ylä5 koodari".

Ohjelmoidaan – Input ja str-funktiot

Nyt pääsemme tutustumaan merkkijonoihin liittyviin funktioihin. Huomaathan että näitä, eikä muitakaan komentoja tarvitse opetella ulkoa. Kirjan liitteenä on **Tiivistelmä työkaluista ja käytännöistä.** Voit aina tarvittaessa etsiä sieltä tarvittavia komentoja.

Ohjeet

Kokeile itsenäisesti input-funktiota ja kirjoita alla olevat koodirivit yksi rivi kerrallaan. Jokaisen rivin päätteeksi lisää itsellesi sopivat kommentit siitä mitä kyseiset komennot tekevät. Käytä #-merkkiä ennen kommenttejasi. Muista, että kommentteja ei suoriteta, mutta ne auttavat sinua ja muita ymmärtämään koodiasi.

Testaa ja tutki!

```
Nimi = "Iiris"
print(Nimi)
input("Kirjoita nimesi: ")   # Tämä komento ei vielä tallenna nimeä!
Nimi2 = input("Kirjoita nimesi: ")   # Tämä tallentaa!
print(Nimi2)
```

> ***Huomio:*** *Voit poistaa kolmannen rivin, koska se ei tee mitään. Kirjoitimme sen vain muistutuksena siitä, että jos käytämme input-funktiota ilman, että teemme lisäksi muuttujan niin käyttäjän kirjoittama teksti ei tallennu mihinkään.*

Jatka yllä olevan koodin perään vastaukset alla oleviin haasteisiin.

Haaste!

Haaste4: Tee ohjelma, joka kysyy käyttäjän suosikkieläintä ja sen jälkeen tulostaa sen.

Haaste5: Tee ohjelma, joka kysyy käyttäjän nimeä ja sen jälkeen tervehtii käyttäjää nimeltä. Laita tietokone myös esittäytymään omalla nimellään. Alla esimerkki tulostuksesta, kun käyttäjän nimi on Mikko ja tietokoneen nimi Iiris.

```
Hei Mikko. Minä olen Iiris.
```

Haaste6: Tee ohjelma, joka kysyy käyttäjän nimeä ja kertoo sitten käyttäjälle mikä nimen ensimmäisen kirjain on. Alla esimerkki, missä käyttäjän nimi on edelleen Mikko.

```
Nimesi alkaa kirjaimella M.
```

Ohjeet

Jatka merkkijonoihin liittyvien funktioiden testaamista. Kirjoita alla oleva koodi, suorita se rivi kerrallaan ja lisää kommentteja siitä mitä komennot tekevät. Huomioi tarvittava määrä sulkuja!

Testaa ja tutki!

```
print(str(Nimi))
print(str.upper(Nimi))
print(str.lower(Nimi))
print(len(Nimi))     # Arvaatko mitä tämä rivi tekee?
```

Onneksi olkoon! Nyt tiedät miten saat käyttäjän syöttämään tietoja ohjelmaan ja miten merkkijonoja muokataan.

Lähetä koodisi!

Tallenna ja lähetä kaikki kirjoittamasi koodi opettajan ohjeiden mukaisesti.

Jaa koodisi ystäviesi kanssa, jos haluat. Vertailkaa ratkaisujanne. Keksittekö erilaisia tapoja ratkaista haasteet?

Luetaan- Datatyypit

Olet toistaiseksi tutustunut yhteen yleiseen datatyyppiin, nimittäin merkkijonoihin. Olet tallentanut sanoja ja tekstiä muuttujaan merkkijonona lisäämällä sanan tai tekstin ympärille lainausmerkit. Lisäksi tiedät, että merkkijonon lyhenne **str** tulee englanninkielisestä sanasta **string**.

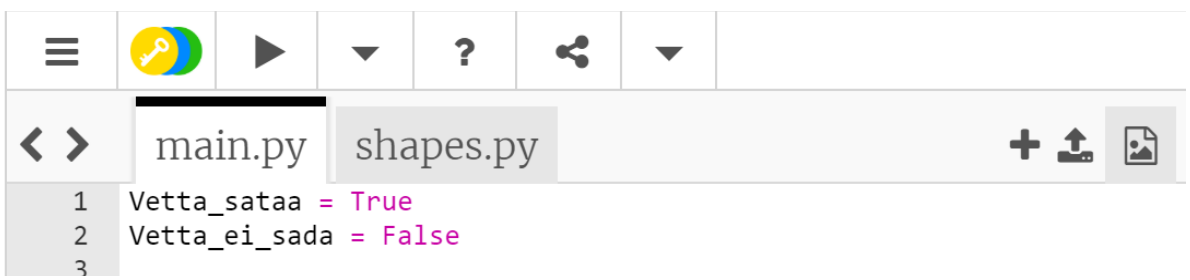
Seuraavaksi tutustutaan kokonaislukuihin, liukulukuihin, listoihin ja totuusarvoon. Nämä kaikki ovat yleisiä datatyyppejä.

Aloitetaan siitä, miten Pythonissa tallennetaan ja käytetään lukuja. Kuten saatat matematiikan tunnilta muistaa on olemassa erityyppisiä lukuja. **Kokonaisluvuilla** ilmoitetaan kohteiden lukumäärää. Pythonissa kokonaislukujen lyhenne on **int** ja se tulee englannin sanasta **integer**. Kokonaislukuja käytetään yleensä esimerkiksi ikää ja pisteitä tallentavissa muuttujissa. Pythonissa desimaalilukuja kutsutaan **liukuluvuiksi**. Englanniksi liukuluku on **float** ja koodissa sitä ei lyhennetä vaan käytetään samaa sanaa. Esimerkiksi lämpötilaa ja rahan arvoa tallentavat muuttujat voivat sisältää liukulukuja.

Tallensitpa sitten merkkijonoja tai lukuja, saatat joskus haluta tallentaa niitä enemmän kuin yhden. Tämä on mahdollista käyttämällä **listaa**, johon viitataan koodissa sanalla **list**. Lista voi sisältää useita eri arvoja ja listassa nämä arvot ovat myös järjestyksessä.

Kuvitellaanpa, että haluat luoda korttipelin. Korttipakassa on 52 korttia, joten joutuisit luomaan 52 muuttujaa, jos tekisit jokaiselle kortille oman muuttujan. Siinä olisi paljon tekemistä! Onneksi Pythonissa muuttujaan voi tallentaa myös listan. Lista luodaan lisäämällä arvot pilkulla erotettuna hakasulkeiden sisään. Alle näet esimerkin Kortit-nimiseen muuttujaan tallennetusta listasta.

Viimeinen yleisistä datatyypeistämme oli **totuusarvo**, joka on englanniksi **boolean** ja se lyhennetään koodissa muotoon **bool**. Totuusarvo voi olla joko tosi eli **True** tai epätosi eli **False.**



Huomaa: *True ja False kirjoitetaan isolla alkukirjaimella.*

Ohjelmoidaan – Datatyypit

Datatyyppeihin tutustumisen jälkeen on sinun vuorosi kirjoittaa koodia ja harjoitella käyttämään niitä.

Ohjeet

Alla olevissa harjoituksissa käydään läpi kaikki yleisimmät datatyypit. Suorita komennot jokaisen rivin jälkeen nähdäksesi mitä koodi tekee. Muista edelleen lisätä kommentteja! Se tapa on hyvä omaksua heti alusta alkaen.

Testaa ja tutki!

```
# Yleisimmät datatyypit

Nimi = "Mikko"                  # merkkijono, engl. string (str)
Ika = 16                        # kokonaisluku, engl. integer (int)
Lampotila = 11.5                # float (float), käytä pistettä!
Ostoslista = ["juusto", "leipä"]    # list (list)
Vastaus = True                      # boolean (bool)

# Tulostetaan print()-funktiolla jokainen muuttuja yksi kerrallaan.
print(Nimi)
print(Ika)
print(Lampotila)
print(Ika + Lampotila)

# Kokeile yhdistää datatyyppejä
print(Ika + Nimi) # Poista tämä rivi sen jälkeen kun olet testannut
#sen

# Testaa mitkä datatyyppien yhdistämiset toimivat ja mitkä eivät
# toimi. Poista ne jotka eivät toimi.
print("Ulkolämpötila on " + Lampotila + ".")
print("Ulkolämpötila on " + str(Lampotila) + " Celsiusastetta.")
```

```
# Testaa seuraavat komennot ja kirjoita kommetit:
print(int(Lampotila))
print(Lampotila)
Lampotila = int(Lampotila)
print(Lampotila)
```

Haaste!

Haaste7: Tee ohjelma, joka pyytää käyttäjää kirjoittamaan nimensä ja ikänsä. Koodaa ohjelma tervehtimään käyttäjää nimeltä ja ilmoittamaan hänen ikänsä. Esimerkki alla:

```
Hei! Nimesi on Mikko. Olet 16 vuotta vanha.
```

Haaste8: Tee ohjelma, joka

1. pyytää käyttäjää kirjoittamaan nimensä ja syntymävuotensa
2. laskee käyttäjän iän syntymävuoden perusteella
3. Tervehtii käyttäjää nimeltä ja ilmoittaa hänen ikänsä

Katso esimerkki alla (Käyttäjän nimi on Mikko ja hän on syntynyt vuonna 2006).

```
Hei! Nimesi on Mikko. Täytät tänä vuonna 16 vuotta.
```

Vinkkejä haasteisiin:

- Muista käyttää input()-funktiota käyttäjän tietojen kysymiseen. Muista myös tallentaa nämä tiedot muuttujiin. Jos et muista miten tämä tehdään niin palaa Input- ja str-funktiot kappaleeseen tai katso apua liitteestä *Lunttilappu Python-ohjelmointiin*
- Haaste8: Mieti miten itse laskisit iän syntymävuoden ja vallitsevan vuoden avulla?

Lähetä koodisi!

Tallenna ja lähetä kaikki kirjoittamasi koodi opettajan ohjeiden mukaisesti.

Jaa koodisi ystäviesi kanssa, jos haluat. Vertailkaa ratkaisujanne. Keksittekö erilaisia tapoja ratkaista haasteet?

Luetaan - Laskutoimitukset

Sekä kokonaislukuja että liukulukuja voidaan käyttää laskutoimituksissa aivan samaan tapaan kuin matematiikassa. Voit kertoa tietokoneelle minkä laskutoimituksen haluat suorittaa eri symbolien avulla. Nämä symbolit edustavat matemaattisia operaatioita.

Matemaattiset operaattorit:

+	***Yhteenlasku***
-	***Vähennyslasku***
*****	***Kertolasku***
/	***Jakolasku***

Huomaa: *Jakolaskun symboli on oikealle kallistuva viiva (/), ei vasemmalle kallistuva.*

Voit käyttää myös vertailuoperaattoreita. Niitä tarvitaan esimerkiksi ehtolauseissa (*jos a < b, niin..*) Näihin tutustutaan vielä lisää myöhemmin.

Vertailuoperaattorit:

a == b	***Yhtäsuuri kuin***
a != b	***Erisuuri kuin***
a < b	***Pienempi kuin***
a > b	***Isompi kuin***
a <= b	***Pienempi tai yhtä pieni kuin***
a >= b	***Suurempi tai yhtä suuri kuin***

Huomaa: *Jos haluat verrata onko a yhtäsuuri kuin b, sinun on käytettävä* ***kahta yhtäsuuruusmerkkiä****. Muuten määrittäisit a:n b:ksi samaan tapaan kuin määrität muuttujalle arvoja.*

Lisäksi on myös **loogisia operaattoreita.** Niitä käytetään yhdistämään vertailuja esimerkiksi ehtolauseissa (***jos*** *a < b* ***ja*** *b < c).* Loogisia operaattoreita ei juurikaan käytetä tässä oppilaan kirjassa, mutta on hyvä tutustua niiden käyttöön

Loogiset operaattorit :

and	and-lauseen arvo on tosi, jos molemmat ehdot ovat tosia
or	or-lauseen arvo on tosi jos jomman kumman ehdon arvo on tosi
not	Muuntaa totuusarvon päinvastaiseksi

Ohjelmoidaan – Laskutoimitukset

Nyt on sinun vuorosi kokeilla mitä kaikkea voit tehdä luvuilla.

Ohjeet

Kirjoita jokainen alla oleva koodirivi ja suorita se. Kirjoita omat kommenttisi komentojen perään.

Testaa ja tutki!

```
54+2

print(54+2)
print(54-2)
print(54*2)
print(54/2)
print("54+2")

print(2+6+"blaablaablaa") # Mikä tässä on vikana?

Numero = 1
print(Numero)
Numero = Numero + 1
print(Numero)

Sana = "Hei"
print(Sana)
Sana = Sana + "iiiii!"
print(Sana)
```

Huomaa: *Kun olen testannut komennon "`print(2+6+"blaablaablaa")`" poista rivi. Muuten ohjelmasi ilmoittaa virheestä joka kerta kun suoritat sen.*

Haaste!

Haaste9: Tee ohjelma, joka kysyy käyttäjältä kaksi luku, kertoo ne yhteen ja tulostaa vastauksen.

Vinkki haasteeseen:

- Tarvitset input()-funktiota lukujen kysymiseen. Tallenna luvut omiin muuttujiin.

Haaste10: Tee ohjelma, joka laskee ympyrän pinta-alan.

Vinkki haasteeseen:

- Ympyrän pinta-ala on luvun π ja säteen neliön tulo
- Tee muuttuja nimeltä Pii ja anna sille arvoksi sopiva likiarvo. **Huomaa**: Pythonissa käytetään desimaalipilkun sijaan pistettä.
- Kysy säteen pituus käyttäjältä.

Lähetä koodisi!

Tallenna ja lähetä kaikki kirjoittamasi koodi opettajan ohjeiden mukaisesti.

Jaa koodisi ystäviesi kanssa, jos haluat. Vertailkaa ratkaisujanne. Keksittekö erilaisia tapoja ratkaista haasteet?

Yhteenvedon aika!

Vastaa seuraaviin kysymyksiin ennen projektin aloittamista.

VASTAA ERILLISEEN TIEDOSTOON TAI VIHKOON JA PALAUTA VASTAUKSESI OPETTAJALLE

1. Miltä ohjelmointitehtävät tuntuivat? Olivatko ne sinulle helppoja vai vaikeita?

2. Miten paljon koet oppineesi uutta?

3. Kuvaile jotain toimivaa opiskelutapaa mitä käytit tehdessäsi tehtäviä. Miksi se oli hyvä? Kerro myös jostain vähemmän toimivasta opiskelutavasta.

Anna itsellesi kouluarvosana tämän hetkisen osaamisesi mukaan:

Arvosana 5-6	Arvosana 7-8	Arvosana 9-10
☐	☐	☐

Miksi valitsit kyseisen arvosanan?

Osa II

Projektityö

Nyt on aika ryhtyä töihin ja testata ohjelmointitaitojasi työelämään liittyvässä projektissa!

Tietokoneohjelmia suunnitellaan auttamaan ihmisiä ja tekemään ihmisten elämästä helpompaa- ja joskus tietysti myös hauskempaa! Ohjelmointi on siis usein ongelmanratkaisua, sillä se tarjoaa ratkaisuja ihmiselämän ongelmiin.

Tässä osassa pääset ratkaisemaan ongelmia, joko itsenäisesti tai sitten parisi kanssa. Oppiminen tapahtuu yrityksen ja erehdyksen kautta, joten älä pelkää tehdä virheitä. Koodauksessa virheiden tekeminen kuuluu asiaan!

Tässä osassa perustat ohjelmistoyrityksen, joka saa toimeksiantoja mielenkiintoiselta asiakkaalta. Toimeksiannot ovat ohjelmointitehtäviä, joiden ratkaiseminen onnistuu osassa 1 opittujen tietojen, sekä liitteenä olevien materiaalien (Python-opas ja Tiivistelmä työkaluista ja käytännöistä) avulla.

Perustetaan ensin ohjelmistoyritys

Ryhmätyö onnistuu parhaiten, kun aluksi sovitaan tietyistä asioista yhdessä. Esimerkiksi työskentelyyn ja viestintään liittyvät säännöt on syytä laatia yhdessä. Varmistakaa, että kaikki ovat tietoisia omista tehtävistään ja yhteisistä tavoitteista ennen töiden aloittamista.

Yrityksellä on hyvä olla mieleenpainuva nimi. Miettikää se ensin ja halutessanne voitte suunnitella myös yrityksen iskulauseen ja logon.

Jokaisen yrityksen olisi hyvä myös miettiä minkälaisia arvoja se noudattaa toiminnassaan. Tämä tarkoittaa sitä minkälaiset asiat yritykselle ovat tärkeitä. Jos yritykselle ovat arvokkaita esimerkiksi ympäristöasiat niin tämä ohjaa sitä minkälaisia tehtäviä yritys tekee ja miten se ne tekee. Keskustele ryhmäsi kanssa yrityksenne arvoista. Tässä kohtaa saa olla myös vitsikäs!

Ohjeet

Täyttäkää alla olevat tiedot yrityksestänne joko erilliselle paperilla tai allaolevaan kuvaan.

Yrityksen nimi: _____

Työntekijät: _____

Iskulause: _____

Yrityksen arvot tai säännöt:

_____

_____

_____

_____

Koodausongelmien kohdatessa

Parhaimmatkin koodarit jäävät joskus jumiin koodinsa kanssa. Tässä on muutamia vinkkejä, joita voit kokeilla ongelmatilanteissa:

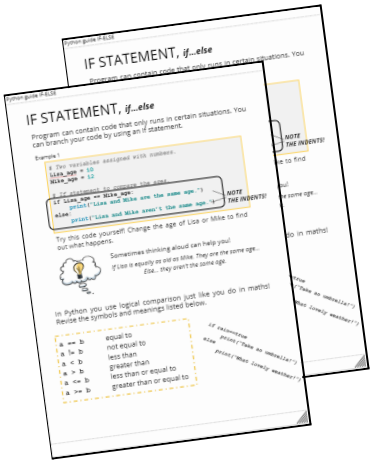
- Ensinnäkin jaa ongelmasi ryhmäsi kanssa ja miettikää sitä yhdessä. Useampi ihminen tarkoittaa myös useampia ideoita!
- Voit kokeilla piirtää ja kirjoittaa ongelmasi paperille.
- jos edelliset neuvot eivät auta, poistu työpöydän ääreltä ja siirry lattialle jumppaamaan hetkeksi. Muutama punnerrus parantaa ajatteluakin!
- Jos jumppaaminenkaan ei auta, voit kysyä neuvoa muilta ryhmiltä tai etsiä sitä internetistä.
- Viimeisenä oljenkortena voit kysyä neuvoa opettajalta. Hän ei todennäköisesti anna sinulle oikeita komentoja, mutta hän voi neuvoa mistä vastausta voisi lähteä etsimään.

Tehtäväsi on kirjoittaa koodia

Yrityksesi vastaanottaa sähköpostia Jouni Virtuaaliselta, joka on Moblies-nimisen yrityksen omistaja. Moblies valmistaa pelaamiseen tarkoitettuja älypuhelimia. He tarvitsevat yritykseltäsi apua esimerkiksi markkinointiin ja asiakaspalveluun liittyvien viestien automatisoinnissa. Jouni lähettää sähköpostitse toimeksiantoja. Ne ja vinkit tehtävien suorittamiseen löytyvät seuraavilta sivuilta.

Moblies

Huomaa: ***Keltaisella yliviivatut*** *kohdat asiakastöissä ovat lisähaasteita! Voit jättää ne halutessasi tekemättä.*

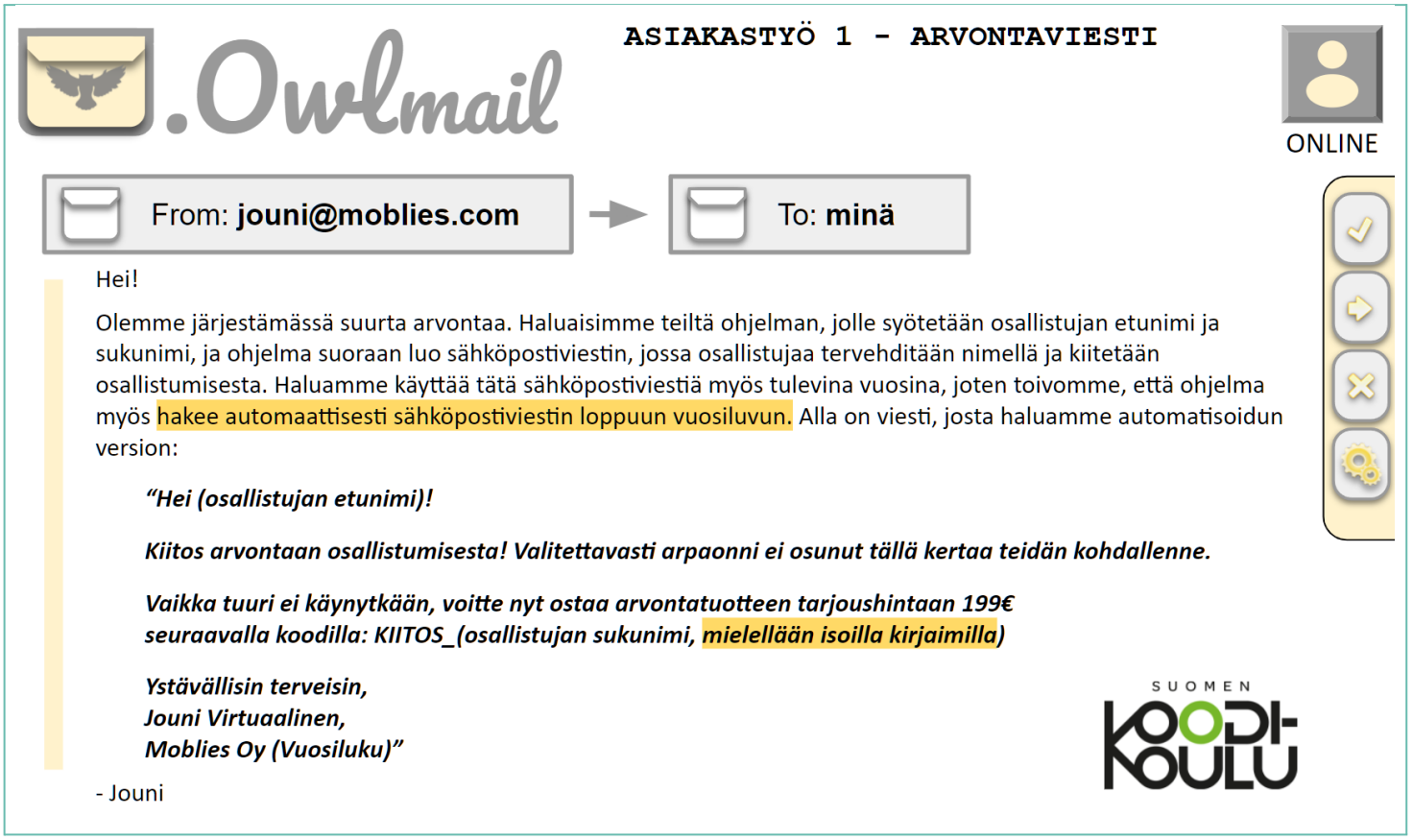


Muista käyttää hyväksesi osan II **Python-opasta!** Voit tarvita myös **Lunttilappu Python-ohjelmointiin** liitettä.

Työniloa ja tsemppiä ongelmanratkaisuun!

1. Asiakastyö - Arvontaviesti

On se aika vuodesta kun Moblies järjestää arvonnan asiakkailleen. Lue alla olevan Jounin viesti. Osaatko auttaa häntä?



Vinkit:

- Vuosiluvun lisäämiseen löydät neuvot Python-oppaasta.
- Osan I harjoituksista löytyy myös apua.

Tehtävälista (*Ruksi tehdyt kohdat*)*:*

- ❑ Lue sähköpostiviesti.
- ❑ Siirry Python-oppaaseen ja opiskele sieltä *Päivämäärä ja aika* -kappale.
- ❑ Suunnittele tehtävän suorittaminen ensin suullisesti tai paperille.
- ❑ Kirjoita ohjelmaan vaadittava koodi. Muista tehdä tässäkin ryhmätyötä!

2. Asiakastyö - Palkkalaskuri

Palkkapäivä lähestyy! Jouni tarvitsee apuasi palkanlaskennassa.

Vinkit:

- Apua kannattaa aina etsiä osan I harjoituksista. Tässä tehtävässä tarvitset matemaattisia operaatioita!
- Muistathan, että **Keltaisella yliviivatut** kohdat asiakastöissä ovat lisähaasteita! Voit jättää ne halutessasi tekemättä.

Tehtävälista (*Ruksi tehdyt kohdat*):

- ❑ Lue sähköpostiviesti.
- ❑ Suunnittele tehtävän suorittaminen ensin suullisesti tai paperille.
- ❑ Kirjoita ohjelmaan vaadittava koodi. Muista tehdä tässäkin ryhmätyötä!

3. Asiakastyö - Räppinimigeneraattori

Työpaikoilla on hyvä järjestää erilaisia tapahtumia työntekijöiden viihtyvyyden parantamiseksi. Lue lisää Jounin viestistä!

Vinkit:

- Kertaa *listat* ja lue niistä lisää Python-oppaasta.
- Käytä hyväksesi Jounin lähettämää koodia.

Tehtävälista (*Ruksi tehdyt kohdat*):

- ❑ Lue sähköpostiviesti.
- ❑ Opiskele Python-oppaan kappale *Listat*
- ❑ Suunnittele tehtävän suorittaminen ensin suullisesti tai paperille.
- ❑ Kirjoita ohjelmaan vaadittava koodi. Muista tehdä tässäkin ryhmätyötä!

4. Asiakastyö - Onnennumero

Onko sinulla onnennumeroa? Jounikin tarvitsee niitä.

Vinkki:

- Lue Python-oppaasta kohta *Omat funktiot*

Tehtävälista *(Ruksi tehdyt kohdat):*

- ❑ Lue sähköpostiviesti.
- ❑ Opiskele Python-oppaan kappale *Omat funktiot*
- ❑ Suunnittele tehtävän suorittaminen ensin suullisesti tai paperille.
- ❑ Kirjoita ohjelmaan vaadittava koodi. Muista tehdä tässäkin ryhmätyötä!

5. Asiakastyö - Nimigeneraattori

Mitä Jounilla on tällä kertaa mielessään? Paljon erilaisia nimiä! Lue Jounin viesti ja mieti ryhmäsi kanssa miten niitä saataisiin tuotettua koodaamalla.

Vinkki:

- Lue Python-oppaasta kohta *Silmukat* ja *Sisäkkäiset silmukat*

Tehtävälista *(Ruksi tehdyt kohdat):*

- ❑ Lue sähköpostiviesti.
- ❑ Opiskele Python-oppaasta kappaleet *Silmukat* ja *Sisäkkäiset silmukat*
- ❑ Suunnittele tehtävän suorittaminen ensin suullisesti tai paperille.
- ❑ Kirjoita ohjelmaan vaadittava koodi. Muista tehdä tässäkin ryhmätyötä!

Projektin loppuyhteenveto

Vastaa seuraaviin pohdintatehtäviin.

VASTAA ERILLISEEN TIEDOSTOON TAI VIHKOON JA PALAUTA VASTAUKSESI OPETTAJALLE

1. Miltä asiakastyöt tuntuivat? Olivatko ne helppoja vai vaikeita?

   _____

   _____

2. Miten ryhmätyö sujui projektin aikana?

   _____

   _____

3. Voisitko kuvitella työskenteleväsi oikeasti ohjelmistoyrityksessä?

   _____

   _____

Anna itsellesi kouluarvosana tämän hetkisen Python osaamisesi mukaan:

Arvosana 5-6	Arvosana 7-8	Arvosana 9-10
☐	☐	☐

Miksi valitsit kyseisen arvosanan?

_____

_____

LIITTEET

Python opas

Tämä osa sisältää tarvittavia tietoja asiakastöiden toteuttamiseen. Tutustu siihen ennen töiden aloittamista, jotta tiedät mitä se sisältää. Mukana on myös muutamia harjoituksia, joiden tekeminen auttaa ymmärtämään opeteltavia asioita.

Kaikkea tässä oppaassa olevaa tietoa ei tarvitse muistaa ulkoa, vaan voit aina palata tarkistamaan asiat täältä.

Ehtolause, *if...else*

Komentojen suorittamiselle voidaan antaa ehtoja. Tässä esimerkki ehtolauseesta, jossa verrataan kahta lukua:

Esimerkki 1 - Yksinkertainen ehtolause:

```
# Kaksi muuttujaa, jotka kuvaavat kahden henkilön ikää.
Julian_ika = 13
Kallen_ika = 16

# Ehtolause, joka vertaa, ovatko henkilöiden iät samat.
if Julian_ika == Kallen_ika:
    print("Julia ja Kalle ovat samanikäisiä.")
else:
    print("Julia ja Kalle ovat eri ikäisiä.")
```

HUOMAA SISENNYKSET!

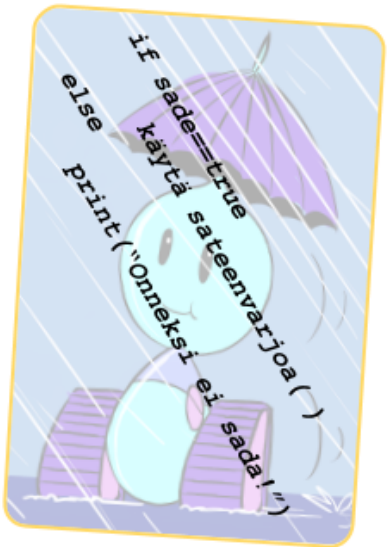
Kokeile yllä olevaa koodia itse. Muuttamalla henkilöiden ikää saatat saada eri tuloksia!

Edellisen ehtolauseen voi **sanallistaa** seuraavasti:

Jos Julian ikä on yhtäsuuri kuin Kallen ikä...
he ovat samanikäisiä.
Muussa tapauksessa...
he ovat eri ikäisiä.

Esimerkissä 1 vertailijana käytettiin yhtäsuuruutta (==). Pythonissa voidaan käyttää myös muita vertailijoita:

`a == b`	yhtäsuuri
`a != b`	erisuuri
`a < b`	pienempi kuin
`a > b`	suurempi kuin
`a <= b`	yhtä suuri tai pienempi kuin
`a >= b`	yhtä suuri tai suurempi kuin



Kun käytetään vertailijana merkkiä >, voidaan verrata kumpi on vanhempi:

Esimerkki 2 - Vertailijana *suurempi kuin*:

```
# Ehtolause, joka vertaa, kumpi on vanhempi.

if Julian_ika > Kallen_ika:
    print("Julia on vanhempi kuin Kalle.")
else:
    print("Kalle on vanhempi kuin Julia.")
```

Ehtoja voidaan lisätä käyttämällä lisäehtoja (***elif*** eli else if). Näin saadaan mukaan myös tapaus, jossa henkilöt ovat saman ikäisiä:

Esimerkki 3 - Lisäehdot:

```
# Ehtolause, joka vertaa, kumpi on vanhempi vai ovatko he
# samanikäiset.

if Julian_ika > Kallen_ika:
    print("Julia on vanhempi kuin Kalle.")
elif Julian_ika < Kallen_ika:
    print("Kalle on vanhempi kuin Julia.")
else:
    print("Julia ja Kalle ovat samanikäisiä.")
```

Tässä vielä harjoitus, jota kannattaa kokeilla. Siinä ilmenee, kuinka voidaan verrata, **onko muuttujan arvo tietyllä välillä**.

Esimerkki 4 - Tämä ohjelma haukkuu sinua iästä riippumatta

```
# Pyydetään käyttäjältä nimi ja syntymävuosi
Nimi = input("Nimesi: ")
Ika = input("Ikäsi: ")

# Muutetaan "Ika" kokonaisluvuksi, jotta sitä voidaan käyttää vertailussa.
Ika = int(Ika)

# iästä riippuen valitaan "Pahasana"
if Ika <= 11:
    Pahasana = "junnu!"
elif 12 <= Ika <= 19:
    Pahasana = "teini!"
else:
    Pahasana = "vanhus!"

#Esitetään lause
print("Hei "+Nimi+". "+"Olet "+str(Ika)+"-vuotias. "+"Senkin "+Pahasana)
```

for-silmukka, for loop

Hyvä tapa toistaa komentoja useita kertoja nopeasti on käyttämällä ***for***-silmukkaa. Sillä voidaan helposti määrätä, kuinka monta kertaa komentoja toistetaan. Seuraavassa esimerkissä tulostus toistetaan 10 kertaa:

Esimerkki 1 - Yksinkertainen silmukka:

Toistojen määrä (10)

```
for i in range(0, 10):
  print("LOL!")
```

HUOMAA SISENNYS

Kokeile yllä olevaa koodia itse. Testaa, mitä tapahtuu kun **muutat *ylärajaa* eli suluissa olevaa jälkimmäistä lukua (10).**

Tämän ohjelman voi **sanallistaa** seuraavasti:

Toistetaan 10 kertaa seuraavat komennot:
Tulosta "LOL!"

Esimerkki 2 - Useita komentoja silmukassa:

```
for i in range(0, 10):
  print("Iso")
  print("paha")
  print("robo!")
```

Kokeile yllä olevaa koodia itse. Komennot suoritetaan järjestyksessä. Viimeisen komennon jälkeen palataan alkuun.

Esimerkki 3 - Yksinkertainen silmukka:

```
luku = 0  # Tehdään muuttuja 'luku', joka on aluksi 0.

for i in range(0, 100):
  luku = luku + 1 # Joka toistolla lukuun lisätään 1.
  print(luku)
```

Kokeile yllä olevaa koodia itse. Voit kokeilla mitä tapahtuu, kun muutat koodissa olevia lukuja.

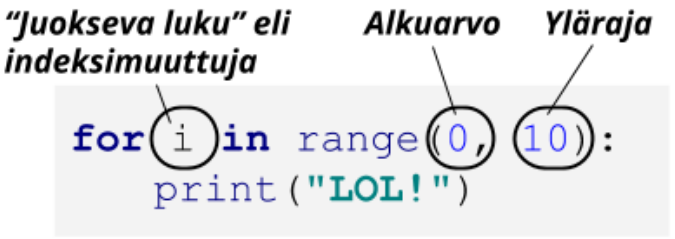
Esimerkki 4 - Kirjaimia lisäävä silmukka:

```
sana = "Moro"
for i in range(0, 100):
  sana = sana + "o"
  print(sana)

sana = sana + "!"
print(sana)
```

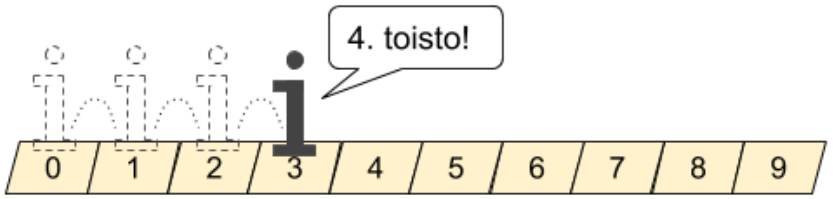
Kokeile yllä olevaa koodia itse. Miksi vain viimeisessä sanassa on perässä huutomerkki? Koska toisto pitää saada päätökseen ennen kuin päästään jatkamaan seuraaviin komentoihin!

Mutta mikä ihmeen i? Ja numerot? Mitä ne tarkoittavat?



Ja sama "suomeksi": Kun alkuarvoksi asetetaan 0 ja ylärajaksi 10, *juokseva luku* i aloittaa nollasta. Kun komennot on kerran suoritettu, *juoksevasta luvusta* tulee 1. Tämä on toinen toistokerta.

Viimeisellä toistokerralla i on 9. Tämä on kymmenes toistokerta.



Esimerkki 4 - 'Juoksevaa lukua' eli indeksimuuttujaa voidaan käyttää toistossa:

```
for i in range(0, 100):
  print(i)
```

Kokeile yllä olevaa koodia itse. *Juokseva luku* i on siis muuttuja, jota voidaan hyödyntää toistorakenteen sisällä. Tässä esimerkissä i on ensimmäisellä toistokerralla 0 ja viimeisellä 99. Toistoja tulee yhteensä 100.

Sisäkkäiset silmukat, nested loops

For-silmukoita voidaan laittaa sisäkkäin. Tehdään kuitenkin ensin ohjelma, joka kokoaa kolmion, joka koostuu #-merkeistä.

Esimerkki 1 - "Kolmion" piirtäminen:

```
for i in range(0, 10): # Toista 10 kertaa
   print("#" * i)      # Tulosta merkki '#' kertaa i.
```

Kokeile yllä olevaa koodia itse. Joka rivillä pitäisi olla aina yksi uusi risuaita (englanniksi *hashtag*). Ohjelma piirsi periaatteessa kolmion!

Mitä jos halutaan piirtää 5 peräkkäistä kolmiota? Voitaisiin kopioida vain yllä olevan koodi 5 kertaa. Mutta vielä parempi on käyttää sisäkkäisiä silmukoita eli laittaa edellinen koodi uuden toistorakenteen sisään.

Esimerkki 2 -Kaksi sisäkkäistä toistorakennetta:

```
for i in range(0, 5):              # Ensimmäinen silmukka
   for u in range(0, 10):          # Toinen silmukka
       print("#" * u)
```

HUOMAA SISENNYKSET

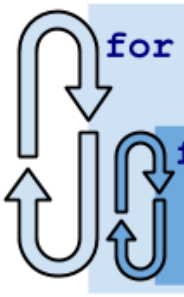
Kokeile yllä olevaa koodia itse. Huomaa, että toisessa silmukassa käytetään indeksin i tilalla u-kirjainta. Näin molemmilla toistorakenteilla on oma indeksi.

Mitä jos laitetaan komento ensimmäisen silmukan sisään ennen toisen silmukan alkua?

Esimerkki 3 -Kaksi sisäkkäistä toistorakennetta, tulostus väleissä:

```
for i in range(0, 5):
   print("Tämä on kolmio numero " + str(i))
   for u in range(0, 10):
       print("#" * u)
```

Kokeile yllä olevaa koodia itse. Seuraavalla sivulla on selitetty tarkemmin, miksi ensimmäiseen silmukkaan laitettu tulostus tapahtuu kolmioiden välissä.



```
for i in range(0, 5):
    print("Tämä on kolmio numero " + str(i))
    for u in range(0, 10):
        print("#" * u)
```

Kun ohjelmassa saavutaan ensimmäiseen silmukkaan...

1. Suoritetaan tulostus *Tämä on kolmio numero 0*
2. Toinen silmukka aloitetaan
3. Ohjelma jää "jumiin" toiseen silmukkaan, kunnes sen kaikki 10 toistoa on suoritettu.
4. Kun toinen silmukka on suoritettu kokonaan, päästään vasta ensimmäistä kertaa ensimmäisen silmukan loppuun. Nyt alkaa ensimmäisen silmukan toinen toistokerta.

Nämä vaiheet toistuvat, kunnes ensimmäisen toistorakenteen kaikki 5 toistoa on suoritettu.

Tämän ohjelman voi **sanallistaa** seuraavasti:

Toistetaan 5 kertaa seuraavat komennot:
> *Tulosta "Tämä on kolmio numero (tähän indeksi)"*
> *Toistetaan 10 kertaa seuraavat komennot:*
> > *Tulosta "#" kerrottuna indeksillä.*

Muuttamalla ensimmäisen silmukan indeksin alkuarvoksi 1, saadaan ohjelma sanomaan ennen ensimmäistä kolmiota *Tämä on kolmio numero 1.* Nyt indeksin maksimin täytyy puolestaan olla 6, jotta toistoja tapahtuu 5 kappaletta.

Kun ohjelma kokoaa risuaita kolmion, ensimmäinen rivi jää tyhjäksi, koska aluksi indeksi on 0. Tämäkin voidaan korjata alkuarvoa ja maksimia muuttamalla:

Esimerkki 4 - Paranneltu ohjelma:

```
for i in range(1, 6):
    print("Tämä on kolmio numero " + str(i))
    for u in range(1, 11):
        print("#" * u)
```

Listat, lists

Listat ovat meille tuttuja: Osallistujalista, ruokalista, biisilista... Myös ohjelmoinnissa voidaan käyttää listoja. Listaan voidaan tallentaa yksittäisten kirjainten ja numeroiden lisäksi muun muassa sanoja ja lauseita. Tässä esimerkki:

Esimerkki 1 - Yksinkertainen lista:

```
# Listanmuuttuja tehdään hakasulkuihin.
# Listan alkiot erotetaan pilkulla.
lista = ["Eka", "Toka", "Kolmas", "Neljäs"]
print(lista)
```

Kokeile yllä olevaa koodia itse. Testaa, mitä tapahtuu kun **muutat *alkioita* eli hakasuluissa olevia listan jäseniä.**

Listan jäseniin eli *alkioihin* päästään käsiksi seuraavalla tavalla:

Esimerkki 2 - Listan alkioiden käsittely:

```
# Listan yksittäisiä alkioita voidaan käsitellä näin.
# Alkio 0 on listan ensimmäinen alkio.
lista = ["Eka", "Toka", "Kolmas", "Neljäs"]

print(lista[0])
print(lista[1])
```

Kokeile yllä olevaa koodia itse.

Useita alkioita voidaan käsitellä kerralla näin::

Esimerkki 3 - Yksinkertainen silmukka:

```
# Useita alkioita käsitellään näin. Alla oleva komento
tulostaa alkiot 1 ja 2. Alkiot 0 ja 3 jäävät pois
print(lista[1:3])
# Tämä tulostaa vain yhden alkion! Sama kuin käyttäisi [0].
print(lista[0:1])
```

Kokeile yllä olevaa koodia itse. Huomaa, että jälkimmäinen numero merkitsee ensimmäistä pois jätettävää alkiota!

Yksittäisiä alkiota voidaan muuttaa:

Esimerkki 4 - Alkioiden muuttaminen:

```
# Yksittäisiä alkiota voidaan muuttaa näin.
lista_kirjaimista = ["A", "B", "C", "D"]
lista[0] = "H"
print(lista)
```

Kokeile yllä olevaa koodia itse. Muuttuiko lista? Osaatko muuttaa muita listan alkioita?

Listaan voidaan lisätä kokonaan uusia alkioita **.append()**-funktiolla:

Esimerkki 5 - Alkioiden lisääminen:

```
# Listaan saadaan lisättyä alkiota .append() funktiolla.
lista_kirjaimista = ["A", "B", "C", "D"]
lista.append("!")
print(lista)
```

Kokeile yllä olevaa koodia itse. Saatko muutettua listaa komennoilla niin, että listan alkioista tulee sana “HAHA!” ?

Listan alkioille voidaan suorittaa laskutoimituksia:

Esimerkki 6 - Yksinkertainen silmukka:

```
# Tehdään lista numeroista. Listan alkioita voidaan muuttaa
# laskutoimituksilla.
numerot = [1, 2, 3]
print(numerot)
print(numerot[1]+10)
print(numerot[1]*100)

# Kun tulostetaan lista, huomataan, ettei lista muuttunut.
print(numerot)

# Listan alkion arvon muuttamisessa pätee samat säännöt kuin
# muidenkin muuttujien arvon muuttamisessa.
numerot[2] = numerot[2] + 7
# Kun tulostetaan lista, huomataan, että nyt lista muuttui.
print(numerot)
```

Kokeile yllä olevaa koodia itse. Saatko muutettua listaa komennoilla niin, että sen alkiot ovat lopulta 10, 20 ja 30?

Omat funktiot, functions

Kun ohjelmasta tulee pitkä ja monimutkainen, omien funktioiden tekemisestä tulee elintärkeää. Tässä yksinkertainen funktio:

Esimerkki 1 - Funktion määrittely:

```
def tervehdi():
    print("Funktio sanoo moi!")
    return
```

HUOMAA SISENNYKSET!

Kokeile yllä olevaa koodia itse. Tuskin mitään tapahtuu. **Tämä rakenne vain määrittelee funktion, mutta sitä ei suoriteta.**

Esimerkki 2 - Funktion suorittaminen eli *kutsuminen*:

```
def tervehdi():
    print("Funktio sanoo moi!")
    return

tervehdi()
```

Funktion määrittely loppuu siihen, mihin sisennykset loppuvat

Muokkaa koodia yllä olevan kaltaiseksi. **Funktio siis täytyy *kutsua*, jotta se oikeasti suoritetaan ohjelmassa.**

Mutta miksi funktion lopussa on **return**? Ja mihin sulut liittyvät?

Funktion erinomaisuus tulee esille vasta kun sille syötetään tietoa ja se ohjelmoidaan käsittelemään tuota tietoa sekä antamaan käsitelty tieto ulos. Tässä esimerkki kolmion pinta-ana laskevastafunktiosta, jolle annetaan argumentteina ***kanta*** ja ***korkeus*** ja se antaa paluuarvona ***alan***.

Esimerkki 3 - Funktio argumenteilla ja paluuarvolla:

```
# Määritellään funktio, joka laskee kolmion pinta-alan
def Kolmion_pinta_ala(kanta, korkeus):
    ala = kanta*korkeus/2
    return ala

Kolmion_pinta_ala(6, 4)
# Funktio suoritettiin, mutta paluuarvona tulevaa alaa ei
# käsketty tulostaa.
print(Kolmion_pinta_ala(6, 4))   # Näin paluuarvo tulostuu!
```

Satunnaisuus, random

Lukujen arpominen on tärkeä osa esimerkiksi peleihin ja tiedon salaamiseen liittyen. Tässä esimerkki noppaohjelmasta:

Esimerkki 1 - Satunnaisen kokonaisluvun arpominen:

```
import random # Otetaan 'random' käyttöön tässä ohjelmassa.

# Näin saadaan arvottua kokonaisluku tietyltä väliltä.
Silmaluku = random.randint(1,6)
print(Silmaluku)
```

Kokeile yllä olevaa koodia itse. Testaa, mitä tapahtuu kun **muutat *randint* funktion argumentteja eli suluissa olevia lukuja.**

Tämän ohjelman voi **sanallistaa** seuraavasti:

Arvotaan luku väliltä 1 ja 6 ja
tallennetaan se nimellä 'Silmaluku'.
Tulostamalla nähdään, minkä luvun tietokone arpoi.

Kokonaislukujen lisäksi voidaan arpoa alkioita listasta* (eli kokonaisia merkkijonoja tai numerosarjoja). Tässä esimerkissä arvotaan, onko ruokana hampurilainen vai munakasta:

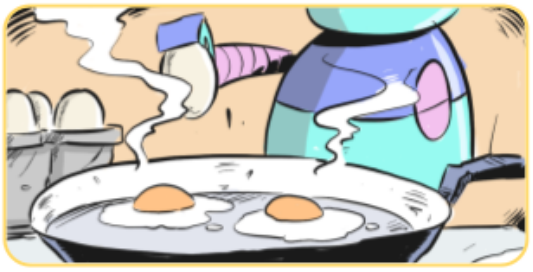
Esimerkki 2 - Satunnaisen sanan arpominen listasta:

```
import random # Otetaan 'random' käyttöön tässä ohjelmassa.

# Esimerkki, jossa arvotaan yksi listan alkio.
Ruokana = random.choice(["hampurilainen","munakas"])
print(Ruokana)
```

Listassa käytetään hakasulkeet

Lisätietoa listoista saat oppaasta "Lista, **list"*



Esimerkin 1 ohjelma toimii kuusi sivuisen nopan tavoin. Siinä arvotaan kokonaisluku väliltä 1 ja 6, kuten noppaa käytettäessä.

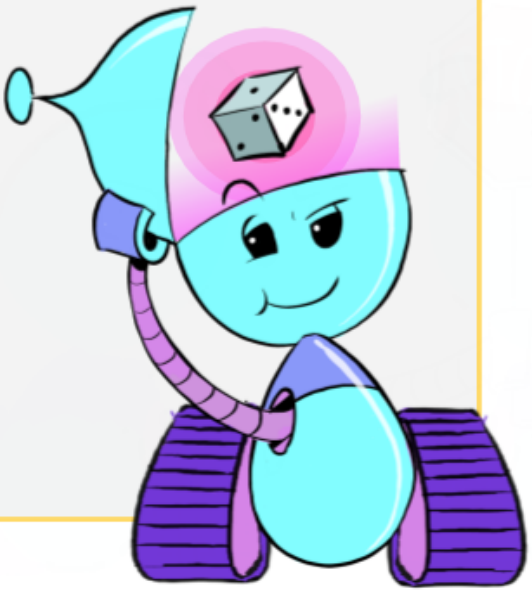
Tähän voidaan yhdistää myös silmäluvun näyttäminen. Tässä esimerkissä silmäluvut esitetään tulostamalla tähtiä ja käyttämällä sopivia välejä:

Esimerkki 3 - Edistynyt noppa ehtolauseella*

```
import random # Otetaan 'random' käyttöön.

# Arvotaan kokonaisluku väliltä 1-6.
Silmaluku = random.randint(1,6)
print(Silmaluku)

# Ehtolauseen avulla näytetään sopiva silmäluku.
if Silmaluku == 1:
    print("     ")
    print("  *  ")
    print("     ")
elif Silmaluku == 2:
    print("    *")
    print("     ")
    print("*    ")
elif Silmaluku == 3:
    print("    *")
    print("  *  ")
    print("*    ")
elif Silmaluku == 4:
    print("*   *")
    print("     ")
    print("*   *")
elif Silmaluku == 5:
    print("*   *")
    print("  *  ")
    print("*   *")
else:
    print("*   *")
    print("*   *")
    print("*   *")
```



*Lisätietoa ehtolauseista saat oppaasta "Ehtolause, ***if...else***".

Päivämäärä, aika ja selain

Pythoniin voidaan tuoda lisää toimintoja tuomalla ohjelmaan moduuleja. Tämä tapahtuu käyttämällä ***import*** komentoa ohjelman alussa. Tässä oppaassa esitellään kolme moduulia hyödyllistä moduulia.

Päivämäärän saat ohjelmaan käyttämällä ***datetime***-moduulia. Näin ohjelmassa voidaan käyttää muun muassa päivämäärää. Päivämäärän saat käyttämällä funktiota **datetime.date.today()** :

Esimerkki 1 - datetime.date.today() funktiolla saadaan päivämäärä:

```
import datetime # Tuodaan datetime-moduuli ohjelmaan.

print(datetime.date.today()) # datetime.date.today() funktio
hakee päivämäärän muodossa VVVV-KK-PP.

# Päivämäärä ei suoraan tule merkkijonona vaikka se siltä
# näyttääkin. Päivämäärä kannattaakin tallentaa muuttujaan
# merkkijonona, jotta siitä voidaan ottaa tarpeellisia
# tietoja ohjelman käyttöön:
Paivamaara = str(datetime.date.today())

# Nyt voidaan ottaa esimerkiksi vuosi talteen uuteen
muuttujaan 'Paivamaara'-muuttujasta
Vuosi = Paivamaara[0:4]
print(Vuosi)  # Tulostetaan se, niin nähdään sen toimivuus.

Kuukausi = Paivamaara[5:7]
print(Kuukausi)

Paiva = Paivamaara[8:10]
print(Paiva)
```

Huomaa, että datetime-funktiot eivät toimi jos *import datetime* ei -komento puuttuu. Hyviin tapoihin kuuluu, että se *import* -komennot ovat ohjelman alussa, mutta todellisuudessa riittää, kun ne ovat vain ohjelmassa ennen niihin liittyvien funktioiden käyttöä.

Aikaan ja ajoittamiseen liittyvät funktiot saat käyttöösi käyttämällä ***time***-moduulia. Pythonissa ei ole odotukseen liittyviä funktioita, mutta *time*-moduuli tuo odotuksen ohjelmaan funktion **time.sleep()** muodossa :

Esimerkki 2 - time.sleep() funktiolla saadaan aikaan viive:

```
import time                    # Tuodaan time-moduuli ohjelmaan

print("Moi!")
time.sleep(1)                  # Sekunnin odotus
print("Kuuleeko kukaan?")
time.sleep(2)                  # Kahden sekunnin odotus
print("HALOO?")
time.sleep(0.5)                # puolen sekunnin odotus
print("HALOO?")
time.sleep(2)
print("Huoh...")
```

Kokeile yllä olevaa koodia itse. Saatko luotua satunnaisen pituisen viiveen käyttämällä *time*-moduulia yhdessä *random*-moduulin* kanssa?

Lisätietoa random-moduulista saat oppaasta "Satunnaisuus, **random"*

Webbrowser-moduuli tuo selaimen käytön ohjelmaan.

Esimerkki 3 - webbrowser.open() funktiolla saadaan avattua selain:

```
import webbrowser
import time

webbrowser.open("https://www.google.com")
time.sleep(1)
webbrowser.open("https://www.codeschoolfinland.com")
time.sleep(1)
```

Kokeile yllä olevaa koodia itse. Tässä on yhdistetty kahden eri moduulin funktioita.

HUOM! Webbrowser-moduuli ei toimi selaimella käytettävissä ohjelmointiympäristöissä (esim. trinket).

Hyvät ohjelmointikäytännöt

1. **MUUTTUJILLE KUVAAVAT NIMET: Nimeä muuttujat kuvaavilla nimillä** → Selkeyttää koodia ja auttaa muita ymmärtämään koodiasi. Jopa itseäsi kun palaat koodin pariin tauon jälkeen.

   Esimerkki:
   ```
   Etunimi = input("Etunimi: ")
   ```

2. **UNIVERSAALIT KIRJAIMET:** Vältetään ääö-kirjaimia muuttujien nimeämisessä → Melkein mikään kieli tai IDE ei tue näitä kirjaimia, ei edes kaikki Python IDE:t.

   Esimerkki:
   ```
   Paivamaara = str(datetime.date.today())
   ```

3. **RIVINVAIHDOT JA SISENNYKSET: Vältä koodin kasaamista samalle riville. Käytä sopivia sisennyksiä** → Selkeyttää koodia ja auttaa muita ymmärtämään koodiasi. Jopa itseäsi kun palaat koodin pariin tauon jälkeen. Pythonissa sisennysten käyttäminen rakenteissa (silmukka, funktio, ehtolause) on pakollista. Yleensä ohjelmointikielissä tätä ei vaadita, mutta sitä käytetään silti.

   Esimerkki: *(Katso alla olevat esimerkit 4 ja 6)*

4. **KOMMENTIT:** Käytä kommentteja aina kun siitä on apua koodin selkeyttämiseksi. → Selkeyttää koodia ja auttaa muita ymmärtämään koodiasi. Jopa itseäsi kun palaat koodin pariin tauon jälkeen.
   Tee sisennys käyttämällä SARKAIN-näppäintä tai kolmea peräkkäistä välilyöntiä.

   Esimerkki:
   ```
   #Jos tietokone arpoi 1, sen ase on kivi.
   if Tietokoneen_ase == 1:
      print("Tietokone valitsi kiven")

      if Pelaajan_ase == "kivi":
          print("Tasapeli.")
   ```

5. **FUNKTIOT: Käytä funktioita jos mahdollista** → Funktioita voidaan käyttää uudelleen ja ne lyhentävät ja selkeyttävät koodia kun ohjelmat paisuvat pitkiksi.

6. **PAIKALLINEN JA GLOBAALIT MUUTTUJAT:** Kun muuttuja mainitaan ensimmäisen kerran, eikä se ole minkään funktion sisällä, on kyseessä globaali muuttuja. Aloita globaalin muuttujan nimet isoilla alkukirjaimilla. Jos muuttuja tehdään funktion sisällä, kyseessä on paikallinen muuttuja (*local*), jota ei voida käyttää funktion ulkopuolella. Silloin alkukirjain on pieni. Näin muuttujien tyyppi näkyy helposti sen nimestä.

   Esimerkki:
   ```
   def Luvusta_negatiivinen(luku):
       luku = luku-luku-luku
       return luku
   Syotetty_luku = input("Syötä luku")
   Luvusta_negatiivinen(Syotetty_luku):
   ```

   'luku' on paikallinen muuttuja

   'Syotetty_luku' on globaali muuttuja

Tiivistelmä työkaluista

Funktio	Toiminto	Esimerkit
`print(" ")`	Tulostaa tekstiä nähtäväksi konsoliin.	`print("Moikka moi!")` `print(Muuttuja1)`
`input(" ")`	Kirjoitussyötteen pyytäminen käyttäjältä.	`input("Syötä nimi:")` `input("Salasana:")`
`str( )`	Muuttaa sulkujen sisällön merkkijonoksi.	`str(Syotetty_ika)`
`str.lower( )`	Muuttaa sulkujen sisällön pieniksi kirjaimiksi.	`str.lower("PIENI")` → Tulostettuna: pieni
`str.upper( )`	Muuttaa sulkujen sisällön isoiksi kirjaimiksi.	`str.lower("suuri!")` → Tulostettuna: SUURI!
`int( )`	Muuttaa sulkujen sisällön kokonaisluvuksi.	`int(1.6)` → Tulostettuna: 1
`float( )`	Muuttaa sulkujen sisällön liukuluvuksi (desimaali).	`float("1.6")` → Tulostettuna: 1.6
`round( )`	Pyöristää liukuluvun lähimpään kokonaislukuun.	`round(1.6)` → Tulostettuna: 1.6
`len( )`	Antaa merkkijonon tai listan pituuden lukuna.	`len("Mada mada")` → Tulostettuna: 9

Tärkeitä moduuleja:

Moduuli	Selitys	Olennaiset funktiot
`import random`	Tuo *satunnaisuus* funktiot ohjelmaan.	`random.randint(1, 6)` `random.choice("Juu","Ei")`
`import datetime`	Tuo *päivämäärä* funktiot ohjelmaan.	`datetime.date.today()`
`import time`	Tuo *aika* funktiot ohjelmaan.	`time.sleep(1)` `time.sleep(0.1)`
`import webbrowser`	Tuo *selain* funktiot ohjelmaan.	`webbrowser.open("https://www.google.com")`

Datatyypit

Esiintyminen koodissa	Datatyyppi	Esimerkit
`a = 0`	Kokonaisluku (int)	`Ikä = 16` `Kinostus = -1`
`a = "Tekstiä"`	Merkkijono (str)	`Nimi = "Sami"` `Supervoima = "Karisma"`
`a = 0.35`	Liukuluku (float)	`Lämpötila = 21.3` `Lämpötila = 21.3`
`a = ["A", "B", "C"]`	Lista (list)	`Vastaus =["Joo", "Ei"]`
`a = True`	Totuusarvo (bool)	`Hauskaa = True` `Helppoa = False`

Silmukka, ehtolause ja oma funktio:

Rakenteen nimi	Esimerkki

Toistorakenne eli silmukka(for)	`for i in range(10):`
	`  # Komennot`
	`  # sisennettynä`

Ehtolause (if-else)	`if a == 1:`
	`  print("Yksi")`
	`elif a == 2:`
	`  print("Kaksi!")`
	`else:`
	`  print("Joku muu")`

Funktion määrittely	`def nimifunk(etunimi):`
	`  koko_nimi = etunimi + " Toivonen"`
	`  return koko_nimi`

Funktion kutsuminen (suorittaminen)	`nimifunk("Kalle")`

SUOMEN KOODIKOULU

ISBN: 978-952-7403-31-0

9 789527 403310

www.suomenkoodikoulu.fi