Jussi Koivisto,
Helena Mäkinen,
Juuso Nieminen

Pythonia yläkouluun

Opettajan opas - Versio 1.2

Kustantaja: Suomen Koodikoulu Oy
www.codeschool.fi

ISBN: 978-952-7403-32-7

Pythonia yläkouluun - Oppimiskokonaisuuden esittely

Tämän oppimiskokonaisuuden tavoitteena on ohjata oppilasta edistämään algoritmista ajatteluaan, hyvien ohjelmointikäytäntöjen käyttöä ja kykyä ohjelmoida yksinkertaisia ohjelmia tekstuaalisessa ohjelmointiympäristössä, Python-ohjelmointikielellä. Oppimiskokonaisuus on tarkoin suunniteltu polku, joka mukailee pelillistä oppimista: Oppilaalle esitellään uusia työkaluja, niiden käyttöä harjoitellaan, sovelletaan ja yhdistetään aikaisemmin opittujen työkalujen kanssa. Harjoituksia ja projekteja suositellaan tekemään pari- ja tiimityönä.

Tämä oppimiskokonaisuus on suunniteltu *POPS 2014* ja *Uudet lukutaidot* -hankkeen osaamistavoitteiden pohjalta. Oppimiskokonaisuudessa korostuvat erityisesti laaja-alaisen osaamisen alueista L1, L4, L5 ja L6.

Oppimiskokonaisuus on kestoltaan 12-14 tuntia. Se toimii erityisen hyvin monialaisena oppimiskokonaisuutena tai valinnaisena kurssina. Kohderyhmä on ohjelmointia jo aikaisemmin harjoitelleet vuosiluokkien 7-9 oppilaat.

Pythonia yläkouluun rakentuu kahdesta osiosta. **Osassa 1** tutustutaan tekstuaalisen ohjelmoinnin perusteisiin. **Osassa 2** sovelletaan juuri opittua ja harjoitellaan erilaisten ohjelmoinnillisten työkalujen käyttöä ongelmanratkaisun ja pienten projektien kautta, työelämän kontekstissa.

***** *Uudet lukutaidot, Ohjelmointiosaaminen: https://uudetlukutaidot.fi/ohjelmointiosaaminen/*

Oppitunnit

Oppimiskokonaisuus koostuu 12-14 oppitunnista. Alla on ehdotus tuntien käytöstä.

Lista oppitunneista		
Oppitunti #	**Sisältö**	**Oppilaan kirjan sivut**
1	Johdatus: Muuttujat ja print-funktio	5-9
2	Input ja str-funktiot	10-12
3	Datatyypit	13-15
4	Laskutoimitukset	16-18
5	Viimeistellään haasteet; Yhteenveto	19
6	Perustetaan yritykset; Säännöt	20-22
7	Projektityö	34-35
8	Projektityö	36-40
9	Projektityö	41-45
10	Projektityö	46-50
11	Projektityö	46-50
12	Projektityö	-
13	Testaa tietosi; Itsearviointi	51 + välikoe

Tämä ajankäyttösuunnitelma pohjautuu 9lk oppilaiden kanssa tehtyihin testauksiin.

Sisällysluettelo

Opettajan oppaan rakenteesta

Opettajan opas sisältää kaikki *Oppilaan kirjan* sivut (vasen), sivulta 5 alkaen. Opettajalle suunnatut ohjeet, tiedot ja työkalut ovat sivun oikealla puolella. Opettajan opas sisältää esimerkkiratkaisut haasteisiin, esimerkkiprojekteja, työkaluja arviointiin ja sisällön laajentamiseen.

Symbolien selitykset

Oppilaan kirjassa toistuu neljä toimintaa kuvaavaa symbolia: ***Ohjeet, Testaa ja tutki, Haaste!*** *ja* ***Keskustele***.

Ohjeet

→ ***Oppilas seuraa kirjan ohjeita tai opettaja ohjaa osion.***

Testaa ja tutki!

→ ***Oppilaat tutkivat ja kokeilevat eri työkaluja ilman tarkkoja ohjeita.***

Haaste!

→ ***Oppilaat ratkaisevat rajattuja tai avoimia ongelmanratkaisutehtäviä.***

Arviointi

→ ***Tämä symboli esiintyy ainoastaan opettajan oppaassa. Sen yhteydessä esitellään työkaluja ja vinkkejä arviointiin.***

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.

Osa 1, *sivut 5-19*

Yleiskatsaus

Osassa 1 tutustutaan tekstuaalisen ohjelmoinnin perusteisiin Python-ohjelmointikielellä. ***Luetaan***-osioissa esitellään uudet käsitteet ja työkalut. ***Ohjelmoidaan***-osioissa työkalujen käyttöä harjoitellaan käytännössä seuraamalla pääasiassa tutkien, testaten ja ratkaisten haasteita. Osan 1 suorittaminen kokonaan on suositeltua ennen osaan 2 siirtymistä.

Osassa 1 suositaan parityöskentelyä, mutta toteutuksen voi tehdä myös yksin tai 3 hengen ryhmissä.

Ajankäyttö

Osa 1 koostuu noin 5 oppitunnista (45 min/oppitunti).

Tarvikkeet

- Tietokone, näppäimistö, hiiri (väh. 1/oppilaspari)
- Oppilaan kirja, tulostettu tai digitaalinen (väh. 1/oppilaspari)

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.

   _____

   _____

   _____

Aloitus - Johdatus osaan 1

Ajankäyttö

15min

Työtapa

Tavoitteet asetetaan **yksin tai pareittain.**

Arviointi

Oppimistavoitteet

Tässä oppimiskokonaisuudessa korostetaan itsearviointia. Tähän on tarkoitus käyttää oppimiskokonaisuuteen suunniteltua ***Taulukkoa oppimistavoitteista*** (erillinen tiedosto) tai opettajan itse kokoamaa tavoitteistoa. Pääasia on, että oppilaat tietävän oppimistavoitteet ja heidän itselleen antama arvosana perustuu niihin.

Oppimistavoitteet voidaan esitellä jo tässä vaiheessa tai myöhemmin, kun oppilas arvioi omaa osaamistaan.

Koodipäiväkirja

Opettajan kannattaa jakaa oppilaille ***Koodipäiväkirja*** (erillinen tiedosto), johon he keräävät tavoitteet, tehtävien ratkaisut ja itsearvioinnin.

Motivointi

Tekstuaalinen ohjelmointi ei lähtökohtaisesti ole kovin motivoivaa, ainakaan aluksi. Opettaja voi motivoida oppilaita esittelemällä osan 2 ajatuksen: Oppilaat tulevat muodostamaan leikkimielisiä ohjelmistoyrityksiä tai tiimejä, joissa he ratkaisevat asiakkaalta sähköpostitse saatuja toimeksiantoja. Tätä ennen täytyy kuitenkin harjoitella ja tätä tehdään osassa 1.

Osan 1 sisältö

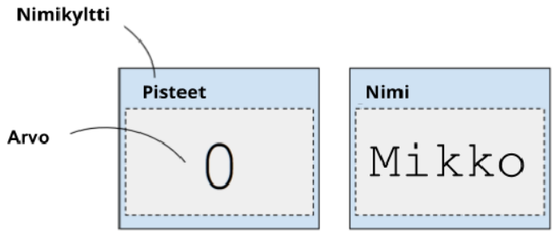
Perusteiden harjoittelu on jaettu neljään osioon, joihin jokaiseen liittyy lukemista ja ohjelmointia. Osiot ovat

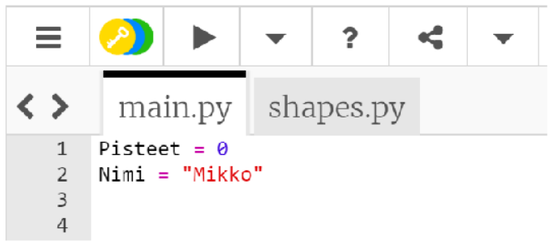
- *Muuttujat ja print-funktio,*
- *Input ja str-funktio,*
- *Datatyypit*
- *Laskutoimitukset.*

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).*

Muuttujat ja print-funktio

Ajankäyttö

30-45min

Työtapa

Työskennellään **1-3** hengen yksiköissä, mieluiten **pareittain**

Avainsanat

Koodi, print, teksti, muuttuja, funktio, tallentaa, tieto, arvo, kommentti

---

Print()

Print-funktion avulla saamme ohjelman tulostamaan tekstiä ruudulle. Tämä oppimiskokonaisuus on rakennettu print-funktion ympärille. Tämä kuulostaa ajatuksena hyvin yksinkertaiselta ja kapealta, mutta todellisuudessa print-funktion avulla pääsee hyvinkin syvälle ohjelmointiin.

Komento, rivi, funktio, lause, rakenne...

Yksittäistä riviä koodissa voidaan kutsutaan yleensä riviksi tai komennoksi, mutta esimerkiksi print() on oikein ilmaistuna funktio. Funktion tunnistaa suluista. Sulkujen sisälle voidaan kirjoittaa argumentteja, joilla ilmaistaan mitä funktion halutaan tekevän. Tyypillinen argumentti print()-funktiolle on lainausmerkeillä kirjoitettu teksti, jonka funktio tulostaa ruudulle.

```
print("Moi")
Kertoja = 2
print(Kertoja * 6)
```

Yllä olevasta koodista voidaan sanoa, että siinä on 3 komentoa tai 3 riviä koodia. Print-funktio esiintyy kahdesti. Komennot voisi laittaa myös samalle riville erottamalla ne puolipisteellä, mutta tätä kannattaa käyttää harkiten. Koodin luettavuus on ensisijaisen tärkeää.

Myöhemmin käytetään myös ehtolauseita (*if-else*) ja toistorakenteita (*for*), joilla voidaan kontrolloida komentojen suorittamista.

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.

main.py

```
1 print ("Kommentit") # Käytä #-merkkiä kommentoidaksesi koodiasi
2 # Voit esimerkiksi kirjoittaa muistiin mitä komennot tekevät
3 # Tässä tapauksessa: print-komento tulostaa saman Kommentit
4
```

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:])
```

Harjoitukset

Osassa 1 olevat harjoitukset perustuvat pääasiassa tutkimiseen ja ongelmanratkaisuun. “***Testaa ja tutki***” -otsikon alla olevaa koodia kirjoitetaan itse ohjelmointiympäristöön, sitä testataan ja siihen liittyen jätetään havaintoja kommentteina. Pääasia on, että oppilaat kirjoittavat koodia itse, suorittavat aina uuden rivin ja tekevät sen perusteella havaintoja.

“***Haaste***”-otsikon alla oppilasta haastetaan ratkaisemaan pieniä koodaustehtäviä. Näillä tehtävillä ohjataan oppilasta tekemään oppimisen kannalta haluttuja oivalluksia.

Harjoituksen ydinaines

- ***print()***-funktio tulostaa tekstiä
- **+** yhdistää erilliset tekstipätkät
- Ohjelman ei arvaa mitä ohjelmoija haluaa ohjelman tekevän. Ohjelmointi on tarkkaa ja johdonmukaista. Tästä esimerkkinä välilyönti rivillä 2 sanan *“Hei “* yhteydessä.
- Muuttujiin voidaan tallentaa esimerkiksi tekstiä.
- Kun halutaan käyttää muuttujan arvoa, muuttujaan viitataan ilman lainausmerkkejä.
- Kun halutaan viitata tiettyyn kirjaimeen muuttujassa, voidaan käyttää hakasulkeita. Laskeminen aloitetaan nollasta eli ensimmäinen kirjain on paikalla [0].
    - Tiedoksi opettajalle: Hakasulkeilla muuttujaan viitataan listana. Eli *Nimi[2]* viittaa muuttujaan listana ja hakee sieltä listan alkion paikalta 2. Koska *Nimi* on merkkijono, ohjelma ajattelee jokaisen kirjaimen olevan oma alkionsa.
- ***Nimi[0:2]*** antaa kirjaimet paikoilta 0 and 1. Miksei myös se anna kolmatta kirjainta paikalta [2]? Helpoin tapa ymmärtää tämä on ajatella, että paikalla 0 on ensimmäinen mukaan otettava kirjain ja paikalla 2 ensimmäinen kirjain, joka jätetään pois.

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?

Kilpikonnilla lisää tekemistä

Jokaiseen perusharjoitukseen on suositeltavaa käyttää yksi oppitunti. Osa oppilaista etenee nopeammin ja heitä voi ohjata auttamaan toisia. Heille voi myös tarjota lisähaastetta esimerkiksi piirto-ohjelmoinnin muodossa.

Pohja, josta voi lähteä muokkaamaan:

```
import turtle

myturtle = turtle.Turtle()

myturtle.forward(100)
myturtle.right(90)
myturtle.forward(100)
myturtle.right(90)
myturtle.forward(100)
myturtle.right(90)
myturtle.forward(100)
myturtle.right(90)
```

Lisää funktioita, esimerkkejä ja sisältöä:

- *https://geeksforgeeks.org/turtle-programming-python/*
- *https://tie.koodariksi.fi/alkeet/turtle*
- *https://peda.net/jyvaskyla/ict/palvelut/ohjelmointi-robotiikka/pop/k*

Esimerkkiratkaisut **Haasteisiin 1-3** sivulla 9

```
# Haaste 1
Sukunimi = "Järvinen"
print(Sukunimi)

# Haaste 2
print(Sukunimi[1:])

# Haaste 3 vaihtoehto A
Nimi = "Mikko"
print(Nimi + Nimi + Nimi + "!")

# vaihtoehto B
print(Nimi * 3 + "!")
```

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.

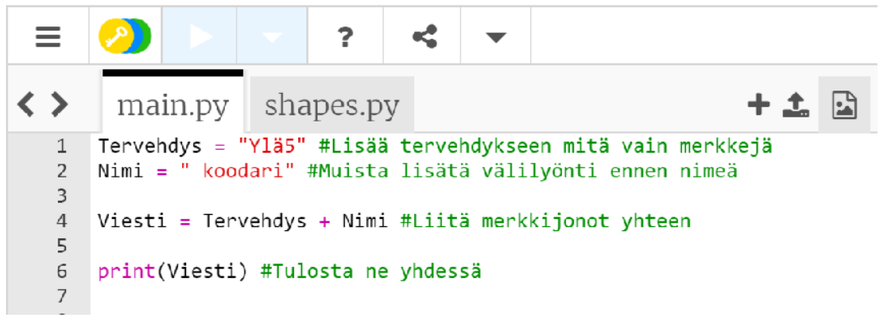
Tässä on esimerkki siitä miten ohjelma pyytää käyttäjää kirjoittamaan nimensä ja tallentaa sen sitten muuttujaan:

```
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 "yläs koodari".

Input ja str-funktiot

Ajankäyttö

30-45min

Työtapa

Työskennellään **1-3** hengen yksiköissä, mieluiten **pareittain**

Avainsanat

Input, tallentaa, print, merkkijono, funktio, koodi, pituus

Sisäänrakennetut funktiot

Pythonissa on valmiina paljon funktioita. Näitä kutsutaan sisäänrakennetuiksi (built-in) funktioiksi. Lisää funktioita saadaan käyttämällä ***import***-komentoa. Tätä käytetään tässä oppimiskokonaisuudessa muun muassa satunnaislukujen käyttämiseksi.

Alla linkki sivustolle, josta löydät kaikki sisäänrakennetut funktiot:
https://www.w3schools.com/python/python_ref_functions.asp

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.
```


Python-ohjelmoinnin muistilista

Oppilaan kirjan sivuilta 44-45 löytyy kaikki tähän oppimiskokonaisuuteen liittyvät tärkeimmät funktiot ja rakenteet esimerkkeineen. Ohjelmoinnissa tavoite ei ole muistaa vaan soveltaa. Siksi esimerkiksi ohjelmointiin liittyvässä kokeessakaan ei tule vaatia muistamista vaan soveltavaa osaamista. Tieto on myös ammattilaisilla aina käden ulottuvilla; Merkittävä osa ohjelmoijan työtä on löytää valmiita ratkaisuja verkosta ja soveltaa niitä.

Esimerkkiratkaisut **haasteisiin 4-6** sivuilla 11-12

```
# Haaste 4
Suosikkielain = input ("Mikä on suosikkieläimesi: ")
print(Suosikkielain)

# Haaste 5
Nimi = "Iiris"
Nimi2 = input("Mikä on nimesi?")
print("Moi " + Nimi2 + ". Minä olen " + Nimi + ".")

# Haaste 6
Nimi3 = input("Syötä nimesi: ")
print("Nimesi alkaa kirjaimella " + Nimi3[0] + ".")
```

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?

str-funktio

Lopuksi esitellään vielä str-funktio. Sen avulla voidaan muuttaa eri datatyypit merkkijonoiksi, mutta myös tehdä niille merkkijonoihin sopivia muutoksia. Str-funktion alla on lisää funktioita, joihin päästään käsiksi pisteellä. Esimerkiksi str.upper() muuttaa kaikki sulkeiden sisällä olevan merkkijonon isot kirjaimet pieniksi.

Viimeisellä rivillä oleva len()-funktio taas kertoo muuttujan tai muun objektin pituuden. Merkkijonon pituus on siinä olevien kirjaimien määrä.

Harjoituksen ydinaines

- ***input()***-funktio pyytää käyttäjältä kirjoitettua vastausta
- ***input()***-funktio ei itse tallenna mitään, ellei sen sisältämää tietoa tallenneta muuttujaan
- ***input()***-funktion argumentiksi voidaan laittaa tekstiä, joka esitetään käyttäjälle. Sen on hyvä olla esimerkiksi kysymys tai kehotus.
- Kun funktioita laitetaan funktioiden sisälle, tulee olla erityisen tarkkana sulkeiden kanssa.
- Jos koodissa on virheitä, ei virheen jälkeistä koodia voida suorittaa. Virhe löytyy yleensä ohjelmointiympäristön osoittamalta virheelliseltä riviltä tai sitä edeltävältä riviltä.

- Tiedoksi opettajalle: Pisteen avulla päästään funktion alla oleviin alifunktioihin. Tästä esimerkkinä ***str.upper(Nimi)***. Funktiot edustavat luokkaa tai kirjastoa, jonka alle liittyy useita siihen liittyviä funktioita. Esimerkiksi str-luokasta löytyy funktiot upper() ja lower().

Luetaan- Datatyypit

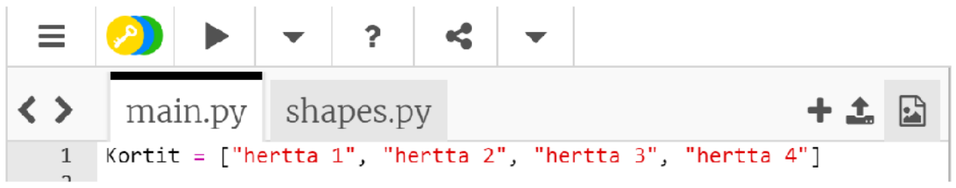
Olet toistaiseksi tutustunut yhteen yleiseen datatyyppiin, nimittäin merkkijonoihin. Olet tallentanut sanoja ja tekstiä muuttujaan merkkijonona lisäämällä sanan tai tekstin ympärille lainausmerkit. Lisäksi tiedät, että merkkijonon lyhenne **str** tulee englanninkielisestä sanasta **string**.

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

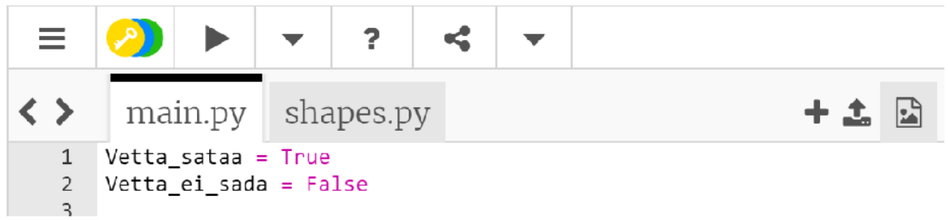
Aloitetaan siitä, miten Pythonissa tallennetaan ja käytetään lukuja. Kuten saatat matematiikan tunnilta muistaa on olemassa erityyppisiä lukuja. **Kokonaisluvuilla** ilmoitetaan kohteiden lukumäärää. Pythonissa kokonaislukujen lyhenne on **int** ja se tulee englannin sanasta **integer**. Kokonaislukuja 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.*

Datatyypit

Ajankäyttö

30-45min

Työtapa

Työskennellään **1-3** hengen yksiköissä, mieluiten **pareittain**

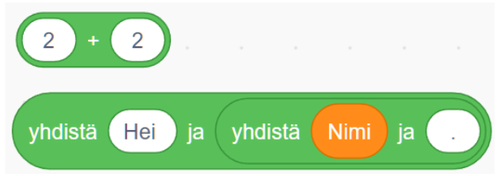
Avainsanat

Data, tieto, merkkijono, kokonaisluku, liukuluku, lista, totuusarvo

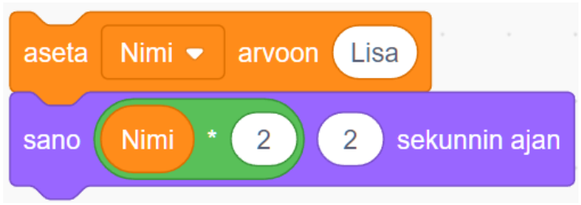
---

Datatyypeistä

Datatyypit ovat usein yksi isoimmista haasteista siirryttäessä visuaalisesta ohjelmoinnista tekstuaaliseen ohjelmointiin. Visuaaliset ympäristöt, kuten Scratch, pyrkivät “suojelemaan” ohjelmoijaa virheiltä. Erityisesti tiedon käsittelyyn liittyvät virheet kuitataan erilaisilla poikkeuksilla. Kun siirrytään ammattimaisiin ohjelmointiympäristöihin niin datatyyppien käsittely tulee ymmärtää, vaikka visuaalinen ohjelmointi olisikin mahdollista (esimerkiksi Unreal Engine).



Scratchissa “+”-lohkolla lasketaan lukuja yhteen ja “yhdistä”-lohkolla yhdistetään tekstiä.



Jos tekstiä yritetään esimerkiksi kertoa Scratchissa, vastaus on 0; Scratchissa ei ole virheilmoituksia vaan oudot tapaukset ohitetaan poikkeussäännöillä.

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.")
```

Datatyypin valinta

Miten valita sopiva datatyyppi? Miksi numerot eivät voisi aina olla liukulukuja? Hyvä tapa on käyttää pienintä mahdollista datatyyppiä: Esimerkiksi jos muuttujan on syytä kertoa, sataako ulkona, tätä tietoa voidaan toki kuvata numeroilla 0 ja 1. Parempi olisi kuitenkin käyttää totuusarvoja *true* ja *false.* Näin kenen vain on helpompi tietää, mitä muuttujalla haetaan.

Kun ohjelma kasvaa esimerkiksi massiiviseksi peliprojektiksi, jokainen muuttuja lisää ohjelman tiedostokokoa. On siis hyvä käyttää pienintä mahdollista muuttujaa, joka riittää tiedon käsittelyyn.

Harjoituksen ydinaines

- On olemassa useita datatyyppejä tai tietotyyppejä.
- Kaikki tieto ei ole esimerkiksi yhdistettävissä toisen tiedon kanssa. Esimerkiksi kokonaisluvun (int) ja merkkijonon (str) yhdistäminen +-operaattorilla ei onnistu, mutta *-operaattori toimii kyllä.
- ***str()***-funktiolla voidaan helposti muuttaa esimerkiksi numerot merkkijonoiksi.
- ***int()***-funktio taas muuttaa datatyypin kokonaisluvuksi.

```
# 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?

Esimerkkiratkaisut **haasteisiin 7-8** sivulla 15

```
# Haaste 7
Nimi = input("Syötä nimi: ")
Ika = input("Syötä ikä: ")
print("Hei! Nimesi on " + Nimi + ". " + "Olet " + Ika +
"-vuotta vanha.")

# Haaste 8
Nimi = input("Syötä nimi: ")
Syntymavuosi = input("Syötä syntymävuosi: ")
Ika = 2022 - int(Syntymavuosi)
print("Hei! Nimesi on " + Nimi + ". " + "Täytät tänä vuonna "
+ str(Ika) + "-vuotta.")
```

HUOMIO: *Viimeinen komento ei täysin mahdu tähän sivulle. Kopioi koodi se ohjelmointiympäristöön nähdäksesi, miltä koodi todellisuudessa näyttää.*

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

Laskutoimitukset

Ajankäyttö

30-45min

Työtapa

Työskennellään **1-3** hengen yksiköissä, mieluiten **pareittain**

Avainsanat

Vertailu, laskutoimitus, operaattori, matemaattinen, looginen

---

Lisää operaattoreista

Lisätietoa operaattoreista: w3schools.com/python/python_operators.asp

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.*

Harjoituksen ydinaines

- Matemaattisilla operaattoreilla voidaan muuttaa tiedon, esimerkiksi muuttujien, arvoa.
  - Matemaattisia operaattoreita kutsutaan myös aritmeettisiksi operaattoreiksi.
- Jos operaattori on lainausmerkkien sisällä, se muuttuu merkkijonoksi ja menettää toimintonsa.
- Muuttujan arvoa voi muuttaa aina uudelleen. Tyypillistä on muuttaa esimerkiksi kokonaislukua yhdellä: *Numero = Numero + 1*.
  - Tiedoksi opettajalle. Edellä kuvattu komento on matematiikan näkökulmasta outo yhtälö, koska kyse ei olekaan yhtälöstä. Yhtäsuuruusmerkki yksinään viittaa asettamiseen: Tämän muuttujan arvo asetetaan nyt uudelleen. Se voidaan sanallistaa "Numero on tästä lähtien itseään yhdellä suurempi".

HUOMIO: *Vertailuoperaattorit ja loogiset operaattorit tulevat käsittelyyn myöhemmin ehtolauseiden yhteydessä.*

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?

Esimerkkiratkasut **haasteisiin 9-10** sivulla 15

```
# Haaste 9
Luku1 = input("Syötä luku: ")
Luku2 = input("Syötä toinen luku: ")
# Ennen kertomista muuttujat tulee muuttaa kokonaisluvuksi.
Luku1 = int(Luku1)
Luku2 = int(Luku2)
print(Luku1*Luku2) # Nyt tulo voidaan esittää.

# Haaste 10
R = input("Syötä ympyrän säde: ")
R = float(R)
Pii = 3.14 # Tarkkuutta voidaan parantaa

Ala = Pii*R*R
Ala = str(Ala)

print("Ympyrän pinta-ala on " + Ala)
```

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?

Välikoe, yhteenveto

Ajankäyttö

45 min

Työtapa

Keskustelua **pareittain**, itsearviointi **yksin.**

Arviointi

Oppimistavoitteet

Tässä oppimiskokonaisuudessa korostetaan itsearviointia. Tähän on tarkoitus käyttää oppimiskokonaisuuteen suunniteltua ***Taulukkoa oppimistavoitteista*** (erillinen tiedosto) tai opettajan itse kokoamaa tavoitteistoa. Pääasia on, että oppilaat tietävän oppimistavoitteet ja heidän itselleen antama arvosana perustuu niihin.

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.

Osa 2, *sivut 20-28*

Yleiskatsaus

Osassa 2 sovelletaan harjoitellaan erilaisten ohjelmoinnillisten työkalujen käyttöä ongelmanratkaisun ja pienten projektien kautta, työelämän kontekstissa. Oppilaat perustavat ryhmissä leikkimielisesti ohjelmistoyritykset, jotka saavat toimeksiantoja asiakkaalta. Osassa 1 opittujen taitojen lisäksi oppilaat opiskelevat uutta sivujen 29-43 Python-oppaista.

Osa 2 on suunniteltu toteutettavaksi 2-4 hengen ryhmissä.

Ajankäyttö

Osa 2 koostuu noin 7-9 oppitunnista (45 min/oppitunti).

Tarvikkeet

- Tietokone, näppäimistö, hiiri (väh. 1/oppilaspari)
- Oppilaan kirja, tulostettu tai digitaalinen (väh. 1/oppilaspari)

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:

_____

_____

_____

_____

Aloitus

Ajankäyttö

20 min

Työtapa

Työskennellään 2-4 hengen yksiköissä (“yrityksissä”).

Yrityksen arvot tai säännöt

Oppilaat sopivat ryhmänsä eli yrityksensä alla säännöistä ja arvoista. Näistä esimerkkejä alla:

- Jokainen yrittää parhaansa mukaan ajaa yrityksen etuja
- Yrityksemme pyrkii auttamaan toisia yrityksiä toiminnassaan
- Emme toivo vahinkoa toisille toimialalla toimiville yrityksille

Motivoi kilpailulla

Opettaja voi järjestää projektityön yhteydessä kilpailun: Yritys, joka tienaa eniten rahaa jakson päätyttyä, voittaa. Voitte myös keksiä luokan kesken valuutalle leikkimielisen nimen.

Esimerkki ansioiden järjestämisestä (valuutan nimi “kultaraha”):

Palkkio suoritetuista asiakastöistä:
Jokainen suoritettu asiakastyö tuo yritykselle 400 kultarahaa. Jos yritys kykenee toimittamaan ylöspäin eriytetyn version asiakastyöstä (keltaisella korostuksella), he tienaavat vielä 200 ylimääräistä kultarahaa.

Palkkio avun antamisesta:
Projektin lopuksi opettaja osoittaa tai oppilaat äänestävät, mikä yritys oli ystävällismielisin ja tarjosi eniten apua muille yrityksille. Tämä yritys saa ylimääräisen 600 kultarahan rahapalkinnon.

Säännöt tulee esitellä oppilaille ennen töiden aloittamista.

Arviointi

Koodipäiväkirja

Osiossa 2 oppilaat täyttävät tunnin tai kaksoistunnin alussa ja jälkeen tehtävät, tavoitteet ja reflektointiosuuden.

Koodausongelmien kohdatessa

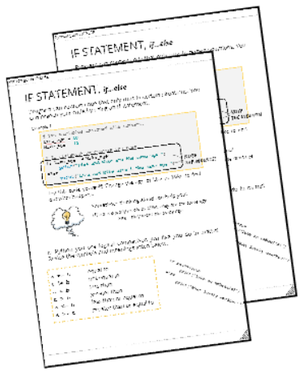
Parhaimmatkin koodarit jäävät joskus jumiin koodinsa kanssa. Tässä on muutamia vinkkejä, joita voit kokeilla ongelmatilanteissa:

- 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.

***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!

Projektityö - Asiakastöiden ratkaiseminen

Ajankäyttö

6-9 oppituntia

Työtapa

Työskennellään 2-4 hengen yksiköissä (“yrityksissä”).

Avainsanat

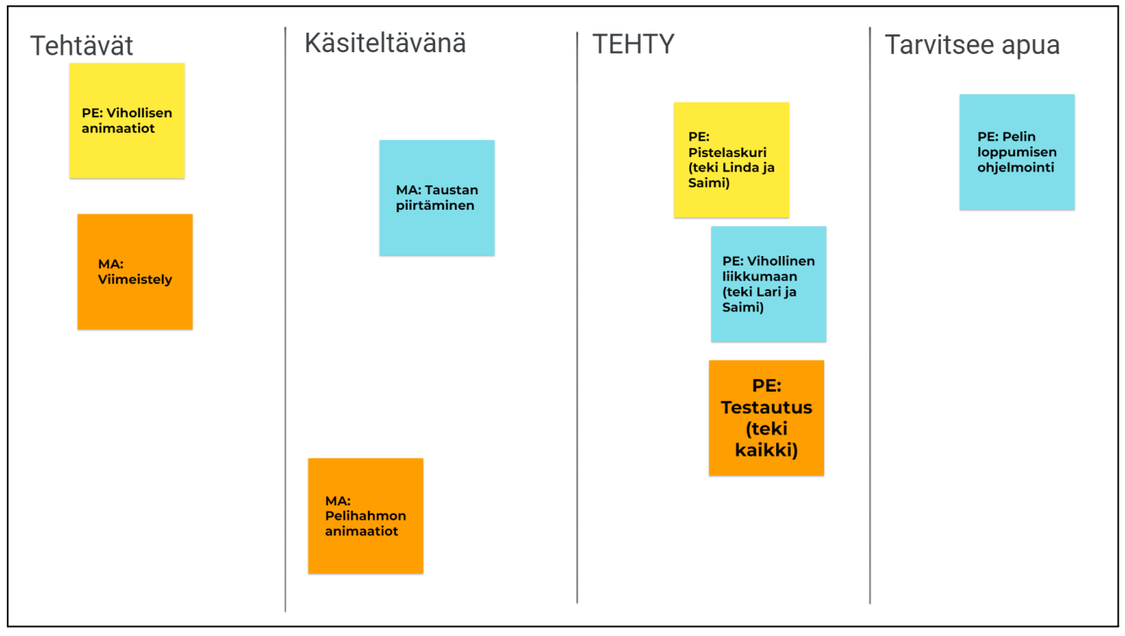
Projekti, ryhmätyö, yhteistyö, ongelmanratkaisu, ohjelmointi, työelämä, yrittäjyys

---

Projektin hallinta - Kanban-taulu

Opettaja voi esitellä oppilaille projektinhallintatyökaluja, jotka auttavat projektin suunnittelussa, toteuttamisessa ja aikataulutuksessa. Yksi helppo esimerkki on Kanban-taulu. Siinä projektin tehtävät jaetaan esimerkiksi osioihin *Tehtävät, Käsiteltävänä, Tehty* ja *Tarvitsee apua.* Ryhmän jäsenillä voi olla oma väri, josta tunnistaa, kenen vastuulla mikäkin tehtävä on. Tehtäviä tehdään silti yhdessä. Kun yksittäinen tehtävä on tehty, siihen voidaan kirjoittaa ketkä sen lopulta tekivät.

Vaikka tämä työkalu onkin oppilaita varten, se auttaa myös arvioinnissa. Opettaja voi arvioida tauluista kunkin oppilaan työpanosta ja siihen voidaan palata vertaisarvioinnin näkökulmasta vielä projektin lopussakin.



Kanban-taulu voidaan toteuttaa kartonkia ja tarralappuja käyttäen tai esimerkiksi PowerPointilla.

1. Asiakastyö - Arvontaviesti

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

ASIAKASTYÖ 1 - ARVONTAVIESTI

ONLINE

From: **jouni@moblies.com** → To: **minä**

Hei!

Olemme järjestämässä suurta arvontaa. Haluaisimme teiltä ohjelman, jolle syötetään osallistujan etunimi ja sukunimi, ja ohjelma suoraan luo sähköpostiviestin, jossa osallistujaa tervehditään nimellä ja kiitetään osallistumisesta. Haluamme käyttää tätä sähköpostiviestiä myös tulevina vuosina, joten toivomme, että ohjelma myös hakee automaattisesti sähköpostiviestin loppuun vuosiluvun. Alla on viesti, josta haluamme automatisoidun version:

"Hei (osallistujan etunimi)!

Kiitos arvontaan osallistumisesta! Valitettavasti arpaonni ei osunut tällä kertaa teidän kohdallenne.

Vaikka tuuri ei käynytkään, voitte nyt ostaa arvontatuotteen tarjoushintaan 199€ seuraavalla koodilla: KIITOS_(osallistujan sukunimi, mielellään isoilla kirjaimilla)

Ystävällisin terveisin,
Jouni Virtuaalinen,
Moblies Oy (Vuosiluku)"

- Jouni

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!

Vaikeustaso

Opettajan oppaassa on tähdin merkitty (1-5), kuinka haastava kukin asiakastyö on.

1. Asiakastyö - Arvontaviesti

Vaikeusaste: ★★☆☆☆

Vinkki: Opettaja voi ohjata oppilaat aloittamaan ensimmäisestä asiakastyöstä. Koodin rakentaminen voidaan aloittaa yhdessä ja opettaja voi antaa oppilaiden sopivassa kohdassa jatkaa itse loppuun.

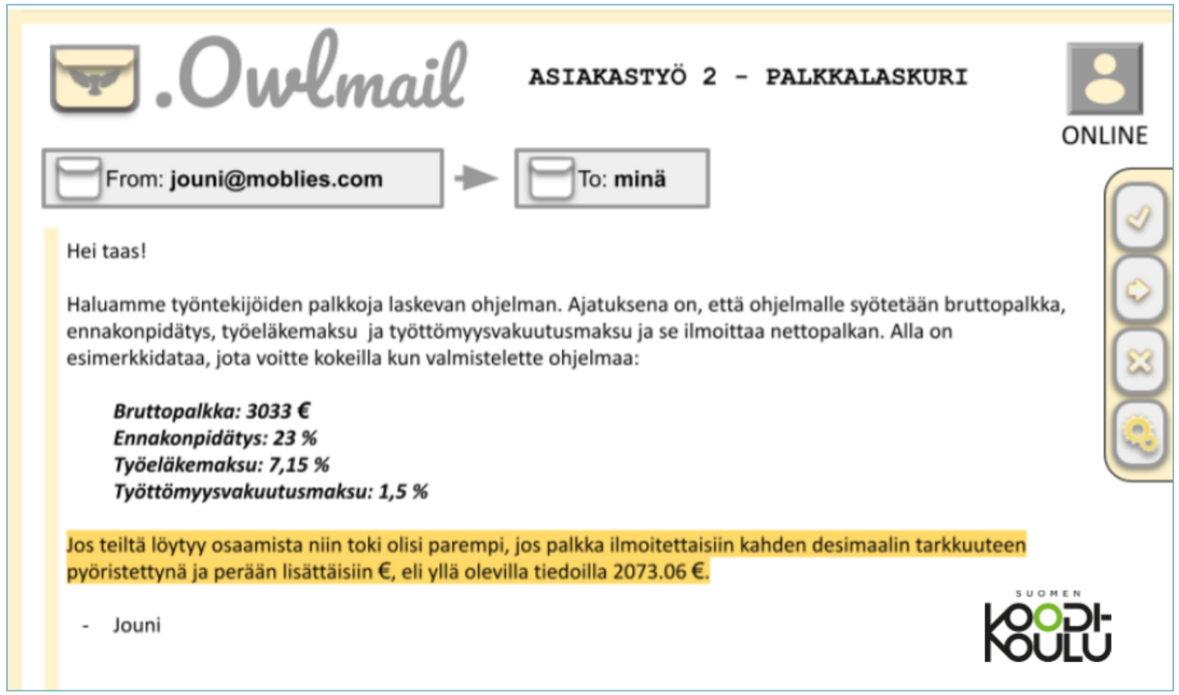
Esimerkkikoodi:

```
import datetime  # Tuodaan tähän ohjelmaan valmis funktio 'datetime'

Paivamaara = str(datetime.date.today())     # Tallennetaan päivämäärä
print(Paivamaara)                           # Tulostetaan päivämäärä(nähdään onko
oikein)

Vuosi = Paivamaara[0:4]        # Haetaan neljä ensimmäistä merkkiä
'Vuosi'-muuttujaan.
print(Vuosi)                   # Tulostetaan se, niin nähdään sen toimivuus.

Etunimi = input("Etunimi: ")  # Kysytään etunimi ja sukunimi.
Sukunimi = input("Sukunimi: ")

# Seuraavassa itse sähköpostiviesti
# Rivinvaihto onnistuu aloittamalla uuden rivin tai käyttämällä \n toimintoa
# Jos halutaan yksi tyhjä rivi, täytyy tehdä kaksi rivinvaihtoa
# Esim. uusi print-rivi yhdistettynä \n toimintoon ajaa asian.

print("Hei " + Etunimi + "!")
print("")
print("Kiitos arvontaan osallistumisesta! Valitettavasti arpaonni ei osunut tällä
kertaa teidän kohdallenne.")
print("Vaikka tuuri ei käynytkään, voitte nyt ostaa arvontatuotteen tarjoushintaan
199€ ")
# Merkkijonoa voi jatkaa toisella rivillä:
print("seuraavalla koodilla: KIITOS_" + str.upper(Sukunimi)) # Merkkijonoa voi
jatkaa toisella rivillä.
print("")
print("Ystävällisin terveisin,")
print("Jouni Virtuaalinen,")
print("Moblies Oy " + Vuosi)
```

HUOMIO: *Kaikki koodi ei mahdu tällä sivulla omille riveilleen. Kopioi koodi ohjelmointiympäristöön.*

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!

2. Asiakastyö - Palkkalaskuri

Vaikeusaste: ★★★☆☆

Esimerkkikoodi:

```
# Tehdään muuttujat, joihin tallennetaan syötetyt luvut
Palkka = input("Syötä bruttopalkka: ")
Ennakonpid = input("Syötä ennakonpidätysprosentti: ")
Tyoelake = input("Syötä työeläkemaksuprosentti: ")
Tyotvak = input("Syötä työttömyysvakuutusmaksuprosentti: ")

Palkka = float(Palkka)          # Tehdään datatyypin muutos laskutoimituksia varten.
Ennakonpid = float(Ennakonpid)
Tyoelake = float(Tyoelake)
Tyotvak = float(Tyotvak)

Ennakonpid = Ennakonpid / 100     # Muutetaan desimaaliluvuksi
Tyoelake = Tyoelake / 100
Tyotvak = Tyotvak / 100

# Lasketaan pidätykset:
Verojen_osuus = Palkka * Ennakonpid     # Vähennetään verojen osuus
Tyoelake_osuus = Palkka * Tyoelake      # Vähennetään työeläkemaksu
Tyotvak_osuus = Palkka * Tyotvak        # Vähennetään työttömyysvakuutusmaksu

#Lasketaan nettopalkka:
Nettopalkka = Palkka - Verojen_osuus - Tyoelake_osuus - Tyotvak_osuus
print(Nettopalkka)

# Tässä ylöspäin eriyttävä osio:
print(str(round(Nettopalkka, 2)) + " €")
```

3. Asiakastyö - Räppinimigeneraattori

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

.Owlmail

ASIAKASTYÖ 3 - Räppinimigeneraattori

ONLINE

From: **jouni@moblies.com** → To: **minä**

Heips!

Järjestämme yrityksen vuosijuhlan, johon olen luvannut keksiä jokaiselle työntekijälle räppinimen tunnelman keventämiseksi. Tiesitkö, että esimerkiksi maailmankuulun artistin Post Malonen artisti- nimi on peräisin räppinimigeneraattorista? Anyways huomasin, että minulla ei ole aikaa keksiä kaikille yksilöllistä räppinimeä. Tähän tarvitsen teiltä apua.

Tarvitsen ohjelman, johon syötetään vain etunimi, josta ohjelma sitten muuttaa ensimmäisen kirjaimen toiseksi. Lisäksi ohjelman pitäisi arpoa räppinimeen jokin stereotypinen etuliite, esimerkiksi "Lil" tai "Dj" tai jotain.

Esimerkiksi jos syöttäisin ohjelmaan "Jouni" vastauksena tulisi: ***Dj Zouni.***

Jos kyvyt riittää niin ohjelma voisi muokata etunimeä vielä niin, että se katsoo onko viimeinen kirjain vokaali. Jos viimeinen kirjain oli vokaali, se poistetaan ja katsotaan onko toiseksi ja kolmanneksi viimeiset kirjaimet konsonantteja. Jos on, toinen niistä poistetaan. Get it?
En tiedä auttaako, mutta löysin pienen koodinpätkän tähän liittyen. Se saattaa nopeuttaa työtä.

```
if Loppuosa[len(Loppuosa) - 1] in "aeiouyää":
        Loppuosa = Loppuosa[0:len(Loppuosa) - 1]
```

- Jouni

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!

3. Asiakastyö - Räppinimigeneraattori

Vaikeusaste: ★★★★☆

Vinkki: Koodin rakentaminen voidaan aloittaa yhdessä ja opettaja voi antaa oppilaiden sopivassa kohdassa jatkaa itse loppuun.

Esimerkkikoodi:

```
import random
Lista_alkuja = ["Lil", "Insane", "Hardcore", "Sugar", "Snoop", "Suppa", "CraZy",
                "Holy", "Post", "Ninja", "Gunner", "Spitter", "Sick", "Dank",
                "Hunter", "DJ"]

Alkuosa = random.choice(Lista_alkuja)     # Alkuosa arvotaan listasta
Loppuosa = input("Anna etunimi: ")

# Loppuosa perustuu käyttäjän etunimeen:
if Loppuosa[0] == "R":
   Loppuosa = "Wr" + Loppuosa[1:]
elif Loppuosa[0] == "T":
   Loppuosa = "D" + Loppuosa[1:]
elif Loppuosa[0] == "P":
   Loppuosa = "B" + Loppuosa[1:]
elif Loppuosa[0] == "S":
   Loppuosa = "X" + Loppuosa[1:]
elif Loppuosa[0] == "K":
   Loppuosa = "G" + Loppuosa[1:]
elif Loppuosa[0] == "H":
   Loppuosa = "W" + Loppuosa[1:]
elif Loppuosa[0] == "N":
   Loppuosa = "Kn" + Loppuosa[1:]
elif Loppuosa[0] == "M":
   Loppuosa = "Z" + Loppuosa[1:]
elif Loppuosa[0] == "J":
   Loppuosa = "Z" + Loppuosa[1:]

print(Alkuosa + " " + Loppuosa)

# Ylöspäin eriytetty osuus
if Loppuosa[len(Loppuosa) - 1] in "aeiouyää":
   Loppuosa = Loppuosa[0:len(Loppuosa) - 1]
   if Loppuosa[len(Loppuosa) - 1] not in "aeiouyää" and Loppuosa[len(Loppuosa) -
2] not in "aeiouyää":
        Loppuosa = Loppuosa[0:len(Loppuosa) - 1]

print(Alkuosa + " " + Loppuosa)
```

HUOMIO: *Kaikki koodi ei mahdu tällä sivulla omille riveilleen. Kopioi koodi ohjelmointiympäristöön.*

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!

4. Asiakastyö - Onnennumero

Vaikeusaste: ★★★☆☆

Esimerkkikoodi:

```
# Onnennumero perustuu henkilön etunimen kirjaimien määrään, kengän kokooon,, ikään, päivämäärään.
# Luvun on oltava aina alle 100. Ohjelmaan syötetään vain nimi, kengännumero ja ikä.
import datetime

# Haetaan päivämäärä. Tehdään siitä merkkijono ja tallennetaan "Paivamaara"-muuttujaan.
Paivamaara = str(datetime.date.today())

# Seuraavaksi erotellaan päivämäärä eri muuttujille. Funktiolta 'datetime' saatu päivämäärä ei
# itseasiassa ole kokonaislukumuodossa, joten samalla pitää muuttaa jokainen kokonaisuluvuksi.
Pv = int(Paivamaara[8:10])     # Näin saadaan muuttujaksi "pv" pelkkä kaksinumeroinen päivä.
print(Pv)                      # Kannattaa tulostaa välissä, jotta nähdään menikö määrittely oikein.
Kk = int(Paivamaara[5:7])# Näin saadaan muuttujaksi "kk" pelkkä kaksinumeroinen kuukausi.
print(Kk)
Vs = int(Paivamaara[2:4])# Näin saadaan muuttujaksi "vs" pelkkä kaksinumeroinen vuosiluku (2019->19).
print(Vs)

# Määritellään funktio, joka generoi onnenluvun ja antaa sen paluuarvona. HUOM:
# Hyviin tapoihin kuuluu, että funktion sisällä määritellyt muuttujat aloitetaan pienellä kirjaimella.
# Kun muuttuja määritellään funktion sisällä, sitä ei voida käyttää funktion ulkopuolella ("lokaali").

def Onnennumerofunktio(nimi, kengannumero, ika):
   loppuluku = len(nimi) * int(kengannumero) / int(ika) * Pv + Kk * Vs

   if loppuluku >= 100:    # Jos luku menee yli sadan, sitä voidaan esimerkiksi jakamalla pienentää.
       loppuluku = loppuluku/7

   loppuluku = int(loppuluku)
   return loppuluku
```

JATKUU SEURAAVALLA SIVULLA

5. Asiakastyö - Nimigeneraattori

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

ASIAKASTYÖ 5 - Nimigeneraattori

Owlmail

ONLINE

From: **jouni@moblies.com** → To: **minä**

Moi!
Vastaan tällä hetkellä puhelin app -projektin testaamisesta. Tarvitsen teiltä ohjelman, joka tekee vähintään 100 erilaista, oikealta kuulostavaa nimeä. Saa tietenkin olla useita, joilla on sama etunimi tai sukunimi, mutta kellään ei saa olla sama koko nimi. Tämän verran sain itse tehtyä:

```
lista_etunimet = ["Anni", "Matti", "Daniel", "Milla"]
lista_suku_alkuja = ["Iso", "Pieni", "Puu"]
lista_suku_loppuja = ["järvi", "vesi", "laakso"]
```

Koitin saada ohjelmoitua niin, että ensin tulostetaan kaikki Annit (Anni Isojärvi, Anni Isovesi, Anni Isolaakso, Anni Pienijärvi jne.) ja sitten siirrytään Matteihin ja sama toistuu kunnes kaikki on käyty läpi. Mutta tässä olevilla nimillä saisin vain 36 nimeä (kokeilin paperille kirjoittamalla).

Kollegani puhui jotain **sisäkkäisistä toistorakenteista**, mutta en ymmärrä mitä se tarkoittaa. Tämän tekemiseen lienee kyllä useita tapoja, mutta en itse keksi yhtään. Jos paukkuja löytyy enempään, 1000 erilaista nimeä olisi vielä parempi!

- Jouni

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!

```
# Pyydetään käyttäjää syöttämään nimi, kengännumero ja ikä.
S_nimi = input("Syötä nimi: ")
S_knumero = input("Syötä kengännumero: ")
S_ika = input("Syötä ikä: ")

# Asetetaan muuttujaksi "Onnennumero" aikaisemmin määritellystä funktiosta saatu
paluuarvo. Tulostetaan.
Onnennumero = Onnennumerofunktio(S_nimi, S_knumero, S_ika)
print("Henkilön " + S_nimi + " onnennumero on tänään " + str(Onnennumero) + ".")
input("Paina ENTER lopettaaksesi.") # Näin voidaan pitää ohjelma "elossa" kunnes
painetaan
# ENTER-näppäintä. Tämä ei ole pakollista, mutta hyvä tietää.
```

HUOMIO: *Kaikki koodi ei mahdu tällä sivulla omille riveilleen. Kopioi koodi ohjelmointiympäristöön.*

5. Asiakastyö - Nimigeneraattori

Vaikeusaste: ★★★★★

Esimerkkikoodi:

```
# Tämä on ylöspäin eriytetty versio. Helpompi versio eroaa nimien ja toistojen
määrässä.

lista_etunimet = ["Mikko", "Matti", "Kalle", "Jussi", "Eelis", "Sami", "Samu",
                  "Samuli", "Hannu", "Aaro", "Harri","Ari","Daniel", "Milla",
                  "Matilda", "Karoliina", "Elina", "Aura", "Lilja", "Jenni",
                  "Jenna", "Hilda"]

lista_suku_alkuja = ["Iso", "Pieni", "Puu", "Koivu", "Mänty", "Ala", "Ylä"]
lista_suku_loppuja = ["järvi", "vesi", "laakso", "niemi", "joki", "harju",
                      "vaara"]

# Ensimmäisen nimen luominen
for i in range(0,22):
   etunimi = lista_etunimet[i]
   for u in range(0, 7):
       sukunimi_alku = lista_suku_alkuja[u]
       for y in range(0, 7):
           sukunimi_loppu = lista_suku_loppuja[y]
           print(etunimi + " " + sukunimi_alku + sukunimi_loppu)
```

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?

_____

_____

Testaa tietosi, itsearviointi

Ajankäyttö

45 min

Työtapa

Keskustelua **pareittain**, itsearviointi **yksin.**

Arviointi

Testaa tietosi (erillinen tiedosto)

Erillisessä tiedostossa olevat ***Testaa tietosi*** -tehtävät voi teettää oppilailla ennen itsearviointia. Ne on tarkoitus tehdä yksin, mutta oppilaan kirjaa saa käyttää avuksi. Tehtäviä yksin yrittämällä oppilas saa paremman käsityksen omasta soveltavasta osaamisestaan. Opettaja voi käyttää tätä myös arvosanan antamisen tueksi. Oikeat vastaukset tehtäviin ovat seuraavalla sivulla.

Itsearviointi perustuu lähtökohtaisesti ***Taulukkoon oppimistavoitteista*** (erillinen tiedosto).

Arviointisuunnitelma

Opettaja voi halutessaan antaa kokonaisarvosanan oppimiskokonaisuudesta. Tässä esimerkki arviointisuunnitelmasta arvosanan antamiseksi:

Prosenttia arvosanasta	**Arvioinnin kohde**
33.3 %	**Itsearviointi** sivulla 28
33.3 %	Ryhmän tekemien onnistuneiden asiakastöiden määrä: 0-1→ Arvosana 4 2 → Arvosana 5-6 3 → Arvosana 7-8 4-5 → Arvosana 9-10
33.3 %	Opettajan arviointi *Testaa tietosi* -tehtävien ja oppimistavoite taulukon perusteella

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.

Testaa tietosi (esimerkkivastaukset)

Oppilaan versio erillisessä tiedostossa.

1. Tee ohjelma, jossa on tallennettu kaikki viikonpäivät (7) listaan. Kun ohjelma suoritetaan, se tulostaa yhden satunnaisen viikonpäivän nimen.

```
import random
paivat = ["Maanantai", "Tiistai", "Keskiviikko", "Torstai",
                    "Perjantai", "Lauantai", "Sunnuntai"]
print(random.choice(paivat))
```

Kirjoita ohjelma Pythonissa. Kun se on valmis, kirjoita se tähän.

2. Tee oma funktio, joka laskee suorakulmion pinta-alan käyttäen kantaa ja korkeutta parametreinä.

```
def P_ala(kanta, korkeus):
    ala = kanta*korkeus
    return ala

print("Suorakulmion ala on " + str(P_ala(10, 12)))
```

Kirjoita ohjelma Pythonissa. Kun se on valmis, kirjoita se tähän.

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!

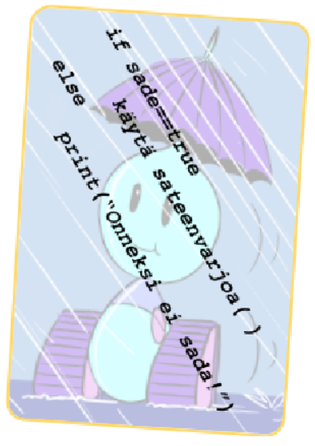
Kokeile yllä olevaa koodia itse. Muuttamalla henkilöiden ikää saatat saada eri tuloksia!

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



Ehtolause

Sisennykset: Sisennyksillä määritetään, mikä kuuluu tietyn rakenteen alle. Sisennyksen voi tehdä käyttämällä TAB-näppäintä tai kahta välilyöntiä rivin alussa.

*Lisätietoa **if**-ehtolauseesta: w3schools.com/python/python_conditions.asp*

Esimerkki 1

```
# 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.")
```

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)
```

Esimerkki 2

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

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

Esimerkki 3

```
# 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.")
```

Esimerkki 4

```
# 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:

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

Toistojen määrä (10)

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.

for-silmukka

*Lisätietoa **for**-silmukasta: www.w3schools.com/python/python_for_loops.asp*

Esimerkki 1

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

Esimerkki 2

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

Esimerkki 3

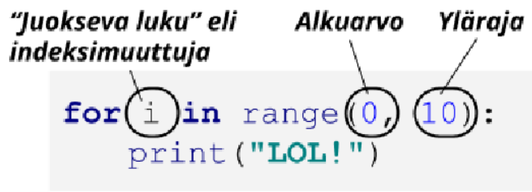
```
luku = 0    # Tehdään muuttuja 'luku', joka on aluksi 0.

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

Esimerkki 4 - Kirjaimia lisäävä silmukka:

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

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

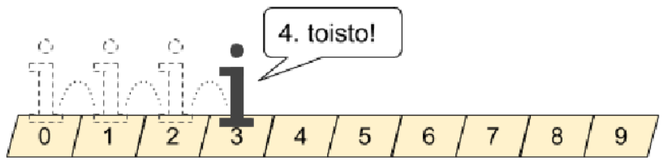
Kokeile yllä olevaa koodia itse. Miksi vain viimeisessä sanassa on perässä huutomerkki? Koska toisto pitää saada päätökseen ennen kuin päästään jatkamaan seuraaviin komentoihin!

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.

Esimerkki 4

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

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

Esimerkki 5

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

While-silmukka

For-silmukka ei ole ainoa tapa toistaa komentoja. ***While***-silmukka toistaa kunnes ehto ei enää ole tosi.

```
i = 0
while i < 100:
    print(i)
    i = i + 1
```

*Lisätietoa **while**-silmukasta: w3schools.com/python/python_while_loops.asp*

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:

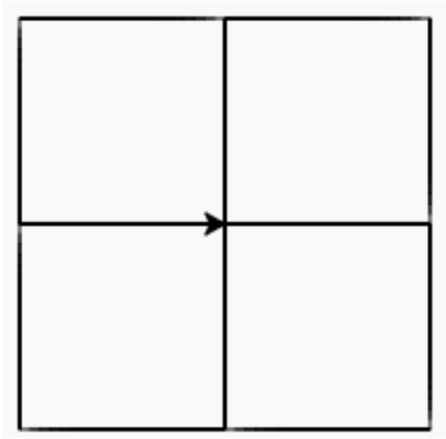
```
for i in range(0, 5):
   print("Tämä on kolmio numero " + str(i))
   for u in range(0, 10):
       print("#" * u)
```

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

Kilpikonna harjoitus

Sisäkkäisiä silmukoita voidaan lähestyä kilpikonnan avulla. Tässä esimerkki, jonka voi esitellä oppilaille:

```
import turtle

myturtle = turtle.Turtle()
for i in range(0, 4):  # Kääntyy vasemmalle ja piirtää neljä neliötä
   myturtle.left(90)
   for u in range(0, 4):  # Piirtää yhden neliön
       myturtle.forward(100)
       myturtle.right(90)
```



Esimerkki 1

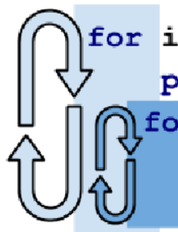
```
for i in range(0, 10):  # Toista 10 kertaa.
  print("#" * i)        # Tuloista '#' kertaa i.
```

Esimerkki 2

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

Esimerkki 3

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



```
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)
```

Esimerkki 4

```
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!

Listat

Hakasulkeiden käyttö: Perusharjoituksissa käytettiin hakasulkeita merkkijonojen kanssa. Merkkijonoon viitattiin siis listana.

*Lisätietoa **listoista**: w3schools.com/python/python_lists.asp*

Esimerkki 1

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

Esimerkki 2

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

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

Esimerkki 3

```
# 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])
```

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?

Esimerkki 4

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

Esimerkki 5

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

Esimerkki 6

```
# 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)
```

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!
```

Funktion määrittely

Parametrit vs argumentit: Kun määrittellään funktio, paikallisia muuttujia kutsutaan **parametreiksi**. Kun kutsumme funktiota viitaten määriteltyihin parametreihin, annamme funktion suorittamiselle **argumentit**.

*Lisätietoa **funktioista**: w3schools.com/python/python_functions.asp*

Esimerkki 1

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

Esimerkki 2

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

tervehdi()
```

Esimerkki 3

```
# 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.

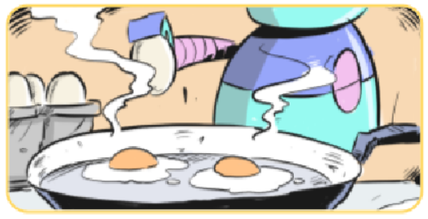
Kokonaislukujen lisäksi voidaan arpoa alkioita listasta* (eli kokonaisia merkkijonoja tai numerosarjoja). Tässä esimerkissä arvotaan, onko ruokana hampurilainen vai munakasta:

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"*



Satunnaisuus

*Lisätietoa **satunnais-funktioista**: docs.python.org/3.8/library/random.html*

Esimerkki 1

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

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

Esimerkki 2

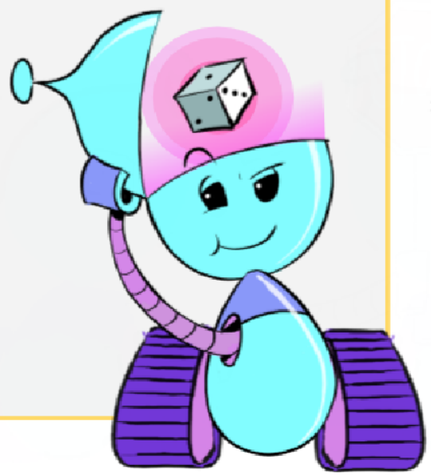
```
import random  # Otetaan 'random' käyttöön tässä ohjelmassa.

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

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**".

Esimerkki 3

```
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("*   *")
```

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.

Päivämäärä, aika ja selain

*Lisätietoa **datetime**-moduulista: w3schools.com/python/python_datetime.asp*

*Lisätietoa **time**-moduulista: programiz.com/python-programming/time*

Esimerkki 1

```
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)
```

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).

Esimerkki 2

```
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...")
```

Esimerkki 3

```
import webbrowser
import time

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

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:

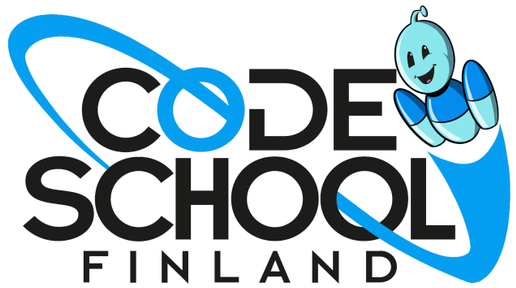
Rakenteen nimi	Esimerkki

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")`



www.codeschool.fi

ISBN: 978-952-7403-32-7