Ohjelmien tekeminen Pythonilla

1. *(Lataa ja asenna Python sivulta https://www.python.org/downloads/)*
2. Avaa IDLE-ohjelmointiympäristö
3. Avaa ohjelmaikkuna.
   - Avaa valitsemalla komentotulkin yläpalkista *File* → *New File* tai pikanäppäinyhdistelmällä C*trl + N*
4. Kirjoita ohjelman koodi ohjelmaikkunaan.
5. Tallenna tiedosto (omaan hakemistoosi).
   - Tallenna valitsemalla ohjelmaikkunan yläpalkista *File* → *Save* tai pikanäppäinyhdistelmällä *Ctrl + S*
     - Anna tiedostolle hyvä ja kuvaava nimi, esimerkiksi *tehtävä2*
     - Varo, ettet vahingossa tallenna minkään aiemman tiedoston päälle!
6. Aja tekemäsi ohjelma.
   - Aja valitsemalla ohjelmaikkunan yläpalkista *Run* → *Run Module* tai pikanäppäimellä *F5*

Python 3:n koko dokumentaatio: https://docs.python.org/3/

Sisällysluettelo

print()

Esimerkkiohjelma	Ohjelman tuloste
`print("merkkijonon ympärille pitää laittaa lainausmerkit")`	`merkkijonon ympärille pitää laittaa lainausmerkit`
`print(1 + 2 * 3)`	`7`
`print("2" + "2")`	`22`
`print("Py" + "thon")`	`Python`
`print("Py", "thon")`	`Py thon`
`print("Moi" * 5)`	`MoiMoiMoiMoiMoi`

Kommentit

Esimerkkiohjelma	Ohjelman tuloste

`#Rivin alussa risuaidalla merkitään`
`#kommentti.`
`#Tulkki ei tulkkaa kommentteja.`
`#Kommenteilla annetaan tietoa ohjelmasta`
`#tai ohjelman osasta`	

`print("tekstiä") #Näinkin voi kommentoida`	`tekstiä`

Matemaattiset operaattorit

Esimerkkiohjelma	Ohjelman tuloste
`#yhteenlasku` `print(5 + 2)`	`7`
`#vähennyslasku` `print(5 - 2)`	`3`
`#kertolasku` `print(5 * 2)`	`10`
`#jakolasku` `print(5 / 2)`	`2.5`
`#katkaiseva jakolasku – ei pyöristystä` `print(5 // 2)`	`2`
`#jakojäännös` `print(5 % 2)`	`1`

Muuttuja & asetuslause

Esimerkkiohjelma	Ohjelman tuloste

`#muuttujaan asetus merkitään merkillä =`
`teksti = "muuttujaan voidaan asettaa tekstiä"`
`print(teksti)`
	`muuttujaan voidaan sijoittaa tekstiä`

`luku1 = 1`
`print(luku1)`
`luku1 = 2`
`print(luku1)`
	`1`
	`2`

`luku1 = 7`
`luku2 = 6`
`summa = luku1 + luku2`
`print(summa)`
	`13`

`teksti_ensimmainen = "Pythonissa useamman sanan mittaiset muuttujien nimet"`
 	
`teksti_toinen = "kirjoitetaan pienellä ja yhdistetään alaviivoilla."`
 	
`print(teksti_ensimmainen, teksti_toinen)`
	`Pythonissa useamman sanan mittaiset muuttujien nimet kirjoitetaan pienellä ja yhdistetään alaviivoilla.`

`#muuttujan arvoa voidaan käyttää asetuslauseessa`
`luku = 1`
`luku = luku + 1 #eli luku = 1 + 1`
`print(luku)`
`luku = luku + 1 #eli luku = 2 + 1`
`print(luku)`
	`2`
	`3`

Tyyppimuunnos merkkijonosta kokonaisluvuksi ja toisin päin

Esimerkkiohjelma	Ohjelman tuloste
`esim = "15" #esim on merkkijono` `esim = int(esim) #nyt esim on kokonaisluku`	

`merkkijono = "12"`
`tavallinen_luku = 3`
`#Tulostetaan muuttujien kertolasku ilman`
`#tyyppimuunnosta.`
`print(merkkijono * tavallinen_luku)`
 	
`#Tehdään nyt tyyppimuunnos.`
`tavallinen_luku2 = int(merkkijono)`
`#Lasketaan tavallinen kertolasku`
`print(tavallinen_luku * tavallinen_luku2)`
	`121212`
	`36`

`#Luku merkkijonoksi`
`ei_enaan_luku = str(3)`
`print(ei_enaan_luku*3)`
	`333`

input()

Esimerkkiohjelma	Ohjelman tuloste
`kayttajan_syote = input()` `print(kayttajan_syote)`	`*tulostaa merkkijonosyötteen, jonka käyttäjä on antanut komentotulkissa*`
`annettu_luku = int(input())` `print(annettu_luku)`	`*tulostaa kokonaisluvun, jonka käyttäjä on antanut komentotulkissa merkkijonona. Jos merkkijonoa ei voi muuttaa luvuksi, ohjelma kaatuu ajon aikana*`
`syote = input("Anna ohjelmalle syöte: ")` `print(syote)`	`Anna ohjelmalle syöte: *käyttäjän syöte*` `*käyttäjän syöte*`

Vertailuoperaattorit

Esimerkkiohjelma	Ohjelman tuloste

`# == on yhtä suuri kuin`
`print(1 == 1)`
`print("kissa" == "koira")`
	`True`
	`False`

`#jaollisuus == -vertailua käyttäen`
`#Alla: neljä on jaollinen kahdella`
`#(eli jakojäännös on nolla)`
 	
`print(4 % 2 == 0)`
	`True`

`# != erisuuri kuin`
`print(1 != 1)`
`print(1 != 2)`
`print(1 != "1") #toinen on merkkijono`
	`False`
	`True`
	`True`

`# < pienempi kuin`
`print(1 < 1)`
`print(1 < 2)`
 	
`# <= pienempi tai yhtä suuri kuin`
`print(1 <= 1)`
`print(1 <= 2)`
	`False`
	`True`
	`True`
	`True`

`# > suurempi kuin`
`print(1 > 1)`
`print(2 > 1)`
 	
`# >= suurempi tai yhtä suuri kuin`
`print(1 >= 1)`
`print(2 >= 1)`
	`False`
	`True`
	`True`
	`True`

and, or ja not

Esimerkkiohjelma	Ohjelman tuloste

`totuusarvo = 1 < 2 and "kissa" == "koira"`
`print(totuusarvo)`
 	
`totuusarvo = 2 < 5 and 0 != 1`
`print(totuusarvo)`
 	
`totuusarvo = 1 == 2 and 2 + 4 == 5`
`print(totuusarvo)`
 	
`totuusarvo = 1 == 2 and 2 + 4 != 5`
`print(totuusarvo)`
	`False`
	`True`
	`False`
	`False`

`totuusarvo = 1 < 2 or "kissa" == "koira"`
`print(totuusarvo)`
 	
`totuusarvo = 2 < 5 or 0 != 1`
`print(totuusarvo)`
 	
`totuusarvo = 1 == 2 or 2 + 4 == 5`
`print(totuusarvo)`
	`True`
	`True`
	`False`

`print(not True)`
`print(not False)`
`print(not 1 == 2)`
	`False`
	`True`
	`True`

if-valintarakenne

Esimerkkiohjelma	Ohjelman tuloste
`if 1 < 2:` `    print("Ehto on True.")`	`Ehto on True.`
`if 2 < 1:` `    print("Ehto on True.")`	

`luku = int(input("Anna luku: "))`
`if luku >= 5:`
`    print("Antamasi luku oli vähintään 5")`
	`Anna luku: *syöte*`
	`*jos syöte on suurempi tai yhtä suuri kuin 5*`
	`Antamasi luku oli vähintään 5`
	
	`*Ei tulosta mitään, jos syöte on pienempi kuin 5*`

`#samalla sisennystasolla oleva koodi`
`#kuuluu samaan loogiseen kokonaisuuteen`
`if 1 < 2:`
`    print("Tässä on tekstiä")`
`    print("ja tässä myös.")`
	`Tässä on tekstiä`
	`ja tässä myös.`

`if 2 < 1:`
`    print("Tulostuu kun ehto on tosi.")`
`print("Tämä tulostuu aina.")`
	`Tämä tulostuu aina.`

if-else-valintarakenne

Esimerkkiohjelma	Ohjelman tuloste

`elain = "kissa"`
 	
`if elain == "koira":`
`  print("Eläin on koira.")`
`else:`
`  print("Eläin ei ole koira.")`
	`Eläin ei ole koira.`

`elain = "koira"`
 	
`if elain == "koira":`
`  print("Eläin on koira.")`
`else:`
`  print("Eläin ei ole koira.")`
	`Eläin on koira.`

`elain = "kissa"`
 	
`if elain == "koira":`
`  print("Eläin on koira.")`
`  print("Hau hau.")`
`else:`
`  print("Eläin ei ole koira.")`
`  print("Ei siis hau.")`
`print("Eläin oli", elain)`
	`Eläin ei ole koira.`
	`Ei siis hau.`
	`Eläin oli kissa`

if-elif-else-valintarakenne

Esimerkkiohjelma	Ohjelman tuloste

`elain = "kissa"`
 	
`if elain == "koira":`
`  print("Eläin on koira.")`
`elif elain == "ankka":`
`  print("Eläin on ankka.")`
`else:`
`  print("Ei ankka eikä koira.")`
	`Ei ankka eikä koira.`

`elain = "ankka"`
 	
`if elain == "koira":`
`  print("Eläin on koira.")`
`elif elain == "ankka":`
`  print("Eläin on ankka.")`
`else:`
`  print("Eläin ei ole koira eikä ankka")`
	`Eläin on ankka.`

while-toistorakenne

Esimerkkiohjelma	**Ohjelman tuloste**

`laskuri = 0`
`while(laskuri < 3):`
`    print(laskuri)`
`    laskuri = laskuri + 1   #laskurin arvo kasvaa yhdellä`
	`0`
	`1`
	`2`

`while(1 != 2):`
`    print("Jumiin jäi.")`
`#jos while-rakenteen ehto ei ole koskaan totta, ohjelma`
`#jää jumiin ja sen voi pysäyttää painamalla ctrl + c`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi.`
	`Jumiin jäi. [jatkuu ikuisesti...]`

Lista

Esimerkkiohjelma	Ohjelman tuloste

`#listan alkiot luetellaan hakasulkeissa`
`#pilkkujen erottamana`
`lista = ["kissa", "koira", "kani"]`
`print(lista)`
	`["kissa", "koira", "kani"]`

`#lista voi sisältää monentyyppisiä alkioita`
`lista2 = ["teksti", True, 2]`
`print(lista2)`
	`["teksti", True, 2]`

`#listan alkioiden määrän saa selville len()-komennolla`
`lista = ["kissa", "koira", "kani"]`
`print(len(lista))`
	`3`

`#listasta voidaan valikoida tietty alkio`
`#huomaa, että listan alkioiden indeksöinti eli`
`#numerointi alkaa nollasta`
`lista = ["a", "b", "c", "d", "e"]`
`print(lista[0]) #tulostetaan listan ensimmäinen alkio`
`print(lista[4]) #tulostetaan listan viides alkio`
	`a`
	`e`

`#listat voidaan yhdistää + -operaattorilla`
`lista1 = ["a", "b", "c"]`
`lista2 = [1, 2, 3]`
`molemmat = lista1 + lista2`
`print(molemmat)`
	`["a", "b", "c", 1, 2, 3]`

`#listaan perään voidaan lisätä yksi uusi alkio`
`lista = ["a", "b", "c"]`
`lista.append("d")`
`print(lista)`
	`["a", "b", "c", "d"]`

`#avainsanan is avulla voidaan tutkia`
`#löytyykö alkio listasta`
`lista = ["a", "b", "c"]`
 	
`print("a" in lista)`
`print("d" in lista)`
	`True`
	`False`

Monikko (Tuple)

Esimerkkiohjelma	Ohjelman tuloste

`#Tuple on kuin lista, mutta tuplen alkioiden arvoja ei voi muuttaa`
`#Lisäksi tuplesta ei voi poistaa eikä tupleen voi lisätä alkioita`
 	
`origo = (0,0)`
`print(origo)`
`print(origo[0])`
 	
`#aiheuttaa virheen`
`origo[0]=1`
	`(0, 0)`
	`0`
	`TypeError:`
	`'tuple' object`
	`does not support`
	`item assignment`

range()

Esimerkkiohjelma	Ohjelman tuloste
`luvut = range(5) #luo lukujonon nollasta annettuun ylärajaan asti` `print(luvut)`	`range(0, 5)`
`luvut2 = range(2, 6) #luo lukujonon kahdesta kuuteen` `print(luvut2)`	`range(2, 6)`
`sataa_pienemmat_parilliset = range(0, 100, 2)` `print(sataa_pienemmat_parilliset)`	`range(0, 100, 2)`

`#myös lukujonosta voidaan poimia tietty alkio`
`luvut = range(10)`
`print(luvut[0]) #lukujonon ensimmäinen alkio löytyy indeksillä 0`
`print(luvut[9]) #lukujonon kymmenes alkio`
	`0`
	`9`

for-toistorakenne

Esimerkkiohjelma	Ohjelman tuloste

`#for-toistorakenteen iteraattori (tässä iteraattoriksi valittu`
`#kirjain i) käy vuorotellen läpi listan tai lukujonon alkiot`
`for i in ["kissa", "koira", "kani", "marsu"]:`
`  print(i)`
	`kissa`
	`koira`
	`kani`
	`marsu`

`#tässä iteraattoriksi on valittu sana ”iteraattori”`
`#rangen luoma jakso ei sisällä koskaan ylärajaansa!`
`for iteraattori in range(5):`
`  print(iteraattori)`
	`0`
	`1`
	`2`
	`3`
	`4`

`#myös merkkijonon suhteen voidaan iteroida`
`#tässä esimerkissä iteraattorina on kirjain j`
`for j in "maksalaatikko":`
`  print(j)`
	`m`
	`a`
	`k`
	`s`
	`a`
	`l`
	`a`
	`a`
	`t`
	`i`
	`k`
	`k`
	`o`

`#for-toistorakenteen yhteydessä voidaan käyttää range-funktion`
`#toimintoja`
`print("Kolmen kertotaulu:")`
`for k in range(0, 31, 3):`
`  print(k)`
	`Kolmen kertotaulu:`
	`0`
	`3`
	`6`
	`9`
	`12`
	`15`
	`18`
	`21`
	`24`
	`27`
	`30`

`#for-silmukoita voi olla useita sisäkkäisiä`
`for i in range(3):`
`  for j in range(3):`
`  print("Ulompi indeksi: ", i, "Sisempi indeksi:", j)`
	`Ulompi indeksi:  0`
	`Sisempi indeksi: 0`
	`Ulompi indeksi:  0`
	`Sisempi indeksi: 1`
	`Ulompi indeksi:  0`
	`Sisempi indeksi: 2`
	`Ulompi indeksi:  1`
	`Sisempi indeksi: 0`
	`Ulompi indeksi:  1`
	`Sisempi indeksi: 1`
	`Ulompi indeksi:  1`
	`Sisempi indeksi: 2`
	`Ulompi indeksi:  2`
	`Sisempi indeksi: 0`
	`Ulompi indeksi:  2`
	`Sisempi indeksi: 1`
	`Ulompi indeksi:  2`
	`Sisempi indeksi: 2`

Funktio

Funktioita käyttämällä samaa toiminnallisuutta hyödyntäessä samaa koodia ei tarvitse kirjoittaa uudelleen. Lisäksi funktioiden käyttö parantaa koodin luettavuutta koodin lyhentyessä ja tiivistyessä.

Huom. Joissain lähteissä ja yhteyksissä funktioista käytetään myös sanaa *metodi.*

def-*avainsana* (eli funktion määrittely)

Esimerkkiohjelma	Ohjelman tuloste

`#Funktio määritellään käyttämällä def-avainsanaa`
`#funktion nimen voi päättää itse, tässä se on hauku`
`def hauku():`
`    #funktion koodi kirjoitetaan sisennettynä`
`    print("Hau!")`
 	
`#määrittelyn jälkeen funktiota voidaan kutsua`
`#(eli käyttää)`
`hauku()`
	`Hau!`

`def tervehdi():`
`    print("Terveisiä funktiosta!")`
 	
`tervehdi()`
`tervehdi()`
`tervehdi()`
	`Terveisiä funktiosta!`
	`Terveisiä funktiosta!`
	`Terveisiä funktiosta!`

main()

Esimerkkiohjelma	Ohjelman tuloste

`#ensin määritellään tarvittavat funktiot`
`def tervehdi():`
`    print("Terveisiä funktiosta!")`
 	
`#main-funktion sisällä on varsinainen ohjelman suoritus`
`def main():`
`    print("Tervehdin seuraavaksi funktion avulla.")`
`    tervehdi()`
 	
`#kutsutaan lopuksi main-funktiota`[1]
`main()`
	`Tervehdin seuraavaksi`
	`funktion avulla.`
	`Terveisiä funktiosta!`

[1] Tarkalleen ottaen yleisesti main-funktiota kutsutaan seuraavalla tavalla:

```
if __name__ == "__main__":
    main()
```

Em. syntaksin ymmärtäminen ja mielekäs hyödyntäminen vaatisi kuitenkin edistyneempien ohjelmointikäsitteiden kuten moduulien hallintaa eikä ole tavoittelemallamme alkeistason perustasolla tarkoituksenmukaista.

Funktion parametrit

Esimerkkiohjelma	Ohjelman tuloste

`#funktiolle voidaan välittää arvoja parametrien arvoina`
`#parametrien ”nimet” tulevat funktion määrittelyssä sulkuihin`
`def summa(eka, toka):`
`  tulos = eka + toka`
`  print(tulos)`
 	
`def main():`
`  #eka = 2, toka = 3`
`  summa(2,3)`
 	
`main()`
	`5`

`def hauku():`
`  print("Hau!")`
 	
`def hauku_monesti(lkm):`
`  #funktioiden koodissa voidaan käyttää tuttuja rakenteita`
`  for i in range(lkm):`
`  #funktioissa voidaan kutsua funktioita`
`  hauku()`
 	
`def main():`
`  maara = int(input("Haukkumäärä: "))`
`  hauku_monesti(maara)`
 	
`main()`
	`Haukkumäärä: *syöte*`
	`*haukkujen määrä riippuu käyttäjän syötteestä*`

return-avainsana

Esimerkkiohjelma	Ohjelman tuloste

`#funktio voi palauttaa arvon myöhempää käyttöä varten`
`#return-avainsanan avulla`
`def summa(eka, toka):`
`  return eka + toka #palauttaa parametrien summan`
 	
`def main():`
`  #palautettava arvo otetaan talteen muuttujaan`
`  luku = summa(2, 3)`
`  print(luku)`
 	
`  luku2 = summa(luku, luku)`
`  print(luku2)`
 	
`main()`
	`5`
	`10`

Muuttujien näkyvyys

Esimerkkiohjelma	Ohjelman tuloste

`#funktiossa luodut muuttujat eivät ”näy” funktion ulkopuolelle`
`#myöskään funktion määrittelyn yhteydessä luotuja parametrejä`
`#ei voi käyttää kyseisen funktion ulkopuolella`
`def summa(eka, toka):`
`  tulos = eka + toka #muuttujaa tulos ei voi käyttää`
`  #funktion summa ulkopuolella`
	`*tämä ohjelma ei tee mitään*`

`def summa(eka, toka):`
`  tulos = eka + toka`
`  return tulos`
 	
`def main():`
`  summa(2,4)`
`  print(tulos)`
 	
`main()`
	`NameError: name 'tulos' is not defined`

Rekursio

Esimerkkiohjelma	Ohjelman tuloste

`#funktio voi kutsua itseään`
`#kertoma on luvun n ja sitä pienempien luonnollisten lukujen tulo`
`def kertoma(n):`
`  if n < 2:`
`  return 1`
`  else:`
`  return n*kertoma(n-1)`
 	
`def main():`
`  #printtaa luvun 5 kertoman eli tulon 5*4*3*2*1`
`  print(kertoma(5))`
 	
`main()`
	`120`

Parametrin oletusarvo

Esimerkkiohjelma	Ohjelman tuloste

`#funktion parametrille voidaan asettaa oletusarvo`
`def kerro_kahdella(n=1):`
`  return n*2`
 	
`def main():`
`  print(kerro_kahdella(2))`
`  print(kerro_kahdella())`
 	
`main()`
	`4`
	`2`

Tuntematon määrä argumentteja

Esimerkkiohjelma	Ohjelman tuloste

`#parametrin nimen edessä oleva * muodostaa funktiolle annetuista`
`#argumenteista tuplen`
`def yhdista_alkukirjaimet(*sanat):`
`  alkukirjaimet = ""`
`  #ei ole merkitystä, kuinka monta argumenttia annetaan`
`  for i in sanat:`
`  alkukirjaimet = alkukirjaimet + i[0]`
`  return alkukirjaimet`
 	
`def main():`
`  print(yhdista_alkukirjaimet("Timo", "Veera", "Tapani"))`
 	
`main()`
	TVT