

# Ohjelmoinnin peruskurssi Y1

CS-A1111

7.11.2018

## Oppimistavoitteet: tämän luennon jälkeen

- ▶ Tiedät, mitä *oliot* ovat ja miksi niitä käytetään ohjelmoinnissa.
- ▶ Tiedät, miten olioiden ominaisuuksia ja niille mahdollisia toimenpiteitä voi määrittellä *luokan* avulla.
- ▶ Tiedät, miten olioita voi luoda ja käyttää niitä määrittelevän luokan ulkopuolella.
- ▶ Voit luennon aikana lähettää kysymyksiä ja kommentteja sivulla <http://presemo.aalto.fi/y1s2018>

# Mitä oliot ovat?

- ▶ Esimerkki: halutaan laatia ohjelma, joka käsittelee erään ohjelmointikurssin opiskelijoita. Kurssilla on noin 100 opiskelijaa.
- ▶ Jokaisesta opiskelijasta halutaan ohjelman käyttöön ainakin nimi, opiskelijanumero, tenttiarvosana ja harjoitusarvosana.
- ▶ Ongelma: miten opiskelijoiden tietoja esitetään ja käsitellään ohjelmassa?
- ▶ Kirjoita ehdotuksia sivulle <http://presemoo.aalto.fi/y1s2018>

## Mitä oliot ovat? (jatkoa)

- ▶ 1. ratkaisu (surkea): otetaan käyttöön 400 muuttujaa eri arvoja varten.
- ▶ 2. ratkaisu (huono): otetaan käyttöön 4 eri listaa: nimet, opiskelijanumerot, tenttiarvosanat ja harjoitusarvosanat. Jokaisessa listassa on 100 alkioita ja saman opiskelijan tiedot ovat listassa aina samalla indeksillä.
- ▶ 3. ratkaisu (parempi): tehdään yhden opiskelijan tiedoista lista, jossa on neljä alkioita. Kurssin kaikista opiskelijoista muodostetaan lista, jonka alkiot ovat listoja.

## Mitä oliot ovat (jatkoa)

- ▶ Oliota käyttävä ratkaisu:
  - ▶ Tehdään jokaista oikeaa opiskelijaa kohti ohjelmaan yksi `Opiskelija`-olio.
  - ▶ Luokassa `Opiskelija` kerrotaan, millaisia `Opiskelija`-oliot ovat ja mitä toimintoja niille voi tehdä.
  - ▶ Kurssin kaikkia opiskelijoita esitetään `Opiskelija`-olioita sisältävänä listana.
- ▶ Oliota käyttävän ratkaisun etuja:
  - ▶ Yhden opiskelijan tiedot yksi kokonaisuus.
  - ▶ Opiskelijan eri tiedot voidaan nimetä selvästi.
  - ▶ Voidaan määritellä myös oliolle mahdolliset toimenpiteet.

# Olioista

- ▶ Olioilla on kenttiä, esimerkiksi Opiskelija-oliolla kentät nimi, opiskelijanumero, harjoitusarvosana ja tenttiarvosana.
- ▶ Jokaisella oliolla on omat kenttien arvot.

nimi = "Teemu Teekkari"  
opiskelijanumero = "445522"  
tenttiarvosana = 3  
harjoitusarvosana = 5

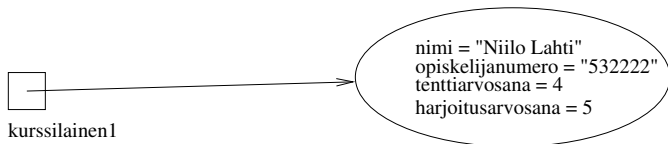
nimi = "Oili Opiskelija"  
opiskelijanumero = "532111"  
tenttiarvosana = 4  
harjoitusarvosana = 4

nimi = "Iiro Ikiteekkari"  
opiskelijanumero = "18999T"  
tenttiarvosana = 2  
harjoitusarvosana = 3

## Olion kenttien arvon muuttaminen

- ▶ Periaatteessa olion kenttien arvoihin voi viitata pistenotaation avulla (kurssilainen1 viittaa luotuun Opiskelija-olioon):

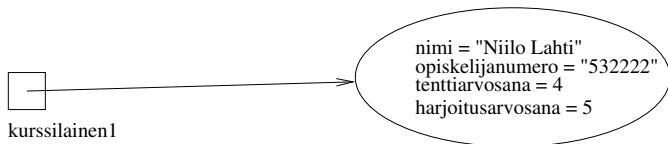
```
kurssilainen1.nimi = "Niilo Lahti"  
kurssilainen1.harjoitusarvosana = 5  
print(kurssilainen1.harjoitusarvosana)
```



## Olion kenttien arvon muuttaminen

- ▶ Periaatteessa olion kenttien arvoihin voi viitata pistenotaation avulla (kurssilainen1 viittaa luotuun Opiskelija-olioon):

```
kurssilainen1.nimi = "Niilo Lahti"  
kurssilainen1.harjoitusarvosana = 5  
print(kurssilainen1.harjoitusarvosana)
```



- ▶ Tämä tapa ei ole kuitenkaan suositeltava (syy selitetty A+:-n kurssimateriaalissa).



# Luokka ja olio

- ▶ Luokassa määritellään, millaisia luokan oliot ovat ja mitä operaatioita olioille voidaan tehdä.
- ▶ Opiskelija-olioiden käsittelemiseksi kirjoitetaan luokka `Opiskelija`.
- ▶ Luokkaan kirjoitetaan metodit, jotka määrittelevät `Opiskelija`-olioille mahdolliset operaatiot.
- ▶ Luokan määrittely aloitetaan luokan otsikolla:  
`class Opiskelija:`

# Olioiden luonti

- ▶ Metodi `__init__` määrittelee, mitä tapahtuu, kun luodaan uusi luokan olio.

```
def __init__(self, annettu_nimi, numero):  
    self.__nimi = annettu_nimi  
    self.__opiskelijanumero = numero  
    self.__tenttiarvosana = 0  
    self.__harjoitusarvosana = 0
```

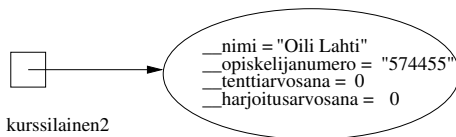
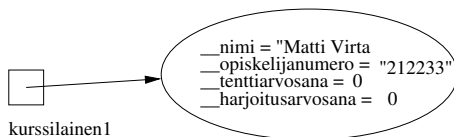
- ▶ Tällöin voidaan luoda uusia `Opiskelija`-olioita:

```
kurssilainen1 = Opiskelija("Matti Virta", "212233")  
kurssilainen2 = Opiskelija("Oili Lahti", "574455")
```

# Olioiden luonti, esimerkki

```
kurssilainen1 = Opiskelija("Matti Virta", "212233")
```

```
kurssilainen2 = Opiskelija("Oili Lahti", "574455")
```



## Muiden metodien määrittely ja kutsuminen

- ▶ Määritellään metodi, jonka avulla voidaan vaihtaa Opiskelija-olion tenttiarvosana.

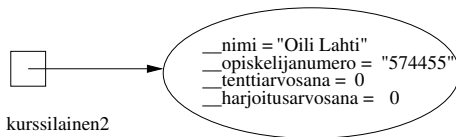
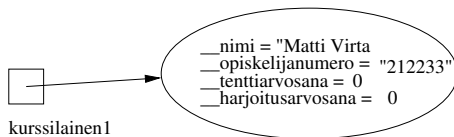
```
def muuta_tenttiarvosana(self, arvosana):  
    if 0 <= arvosana <= 5:  
        self.__tenttiarvosana = arvosana
```

- ▶ Metodin kutsuminen Opiskelija-oliolle, johon viitataan muuttujalla kurssilainen1:

```
kurssilainen1.muuta_tenttiarvosana(4)
```

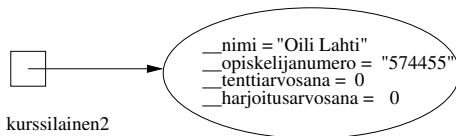
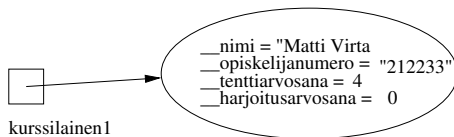
# Metodin kutsuminen, esimerkki

`kurssilainen1.muuta_tenttiarvosana(4)`



# Metodin kutsuminen, esimerkki

`kurssilainen1.muuta_tenttiarvosana(4)`



## Arvon palauttava metodi

- ▶ Metodi Opiskelija-olion kokonaisarvosanan laskemiseen:

```
def laske_kokonaisarvosana(self):
    if self.__tenttiarvosana == 0 or \
        self.__harjoitusarvosana == 0:
        arvosana = 0
    else:
        arvosana = (self.__tenttiarvosana + \
                    self.__harjoitusarvosana + 1) // 2
    return arvosana
```

- ▶ Esimerkki metodin kutsumisesta (luokan ulkopuolella):

```
tulos = kurssilainen1.laske_kokonaisarvosana()
print("Kokonaisarvosana on", tulos)
```

## Metodit kenttien arvojen selvittämiseen

- ▶ Metodit palauttavat suoraan jonkin kentän arvon:

```
def kerro_nimi(self):  
    return self.__nimi
```

```
def kerro_tenttiarvosana(self):  
    return self.__tenttiarvosana
```

```
def kerro_tenttiarvosana(self):  
    return self.__tenttiarvosana
```

```
def kerro_harjoitusarvosana(self):  
    return self.__harjoitusarvosana
```

- ▶ Esimerkki metodin kutsumisesta:

```
print("Nimi on", kurssilainen1.kerro_nimi())
```



## Opiskelija-luokka kokonaan

```
class Opiskelija:

    def __init__(self, annettu_nimi, numero):
        self.__nimi = annettu_nimi
        self.__opiskelijanumero = numero
        self.__tenttiarvosana = 0
        self.__harjoitusarvosana = 0

    def kerro_nimi(self):
        return self.__nimi

    def kerro_opiskelijanumero(self):
        return self.__opiskelijanumero
```

## Opiskelija-luokka jatkuu

```
def kerro_tenttiarvosana(self):  
    return self.__tenttiarvosana  
  
def kerro_harjoitusarvosana(self):  
    return self.__harjoitusarvosana  
  
def muuta_tenttiarvosana(self, arvosana):  
    if 0 <= arvosana <= 5:  
        self.__tenttiarvosana = arvosana  
  
def muuta_harjoitusarvosana(self, arvosana):  
    if 0 <= arvosana <= 5:  
        self.__harjoitusarvosana = arvosana
```

## Opiskelija-luokka jatkuu

```
def laske_kokonaisarvosana(self):
    if self.__tenttiarvosana == 0 or \
        self.__harjoitusarvosana == 0:
        arvosana = 0
    else:
        arvosana = (self.__tenttiarvosana + \
                    self.__harjoitusarvosana + 1) // 2
    return arvosana
```

# Välitehtävä

- ▶ Kirjoita pääohjelma, joka tekee seuraavat asiat käyttämällä hyväksi `Opiskelija`-luokan metodeita:
  - ▶ Luo uuden `Opiskelija`-olion, jonka nimi on Tiina Teekkari ja opiskelijanumero 223340.
  - ▶ Asettaa Tiinan tenttiarvosanaksi 3:n ja harjoitusarvosanaksi 5:n.
  - ▶ Laskee Tiinan kokonaisarvosanan ja tulostaa sen.
- ▶ Jatkokysymys: miten muutat ohjelmaa niin, että nimi ja arvosanat kysytäänkin käyttäjältä?

## Esimerkki luokkaa käyttävästä pääohjelmasta

- ▶ Seuraavalla kalvolla on esimerkkiohjelma, joka pyytää kahden opiskelijan tiedot ja luo heitä vastaavat Opiskelija-oliot.
- ▶ Ohjelma on kirjoitettu luokan Opiskelija ulkopuolelle.
- ▶ Lisäksi on määritelty apufunktio kokonaisluvun lukemiseen.

## Opiskelija-olioita käyttävä ohjelma, koodi

```
def lue_kokonaisluku():
    luku_onnistui = False
    while not luku_onnistui:
        try:
            luku = int(input())
            luku_onnistui = True
        except ValueError:
            print("Virheellinen kokonaisluku!")
            print("Anna uusi!")
    return luku
```

## Opiskelija-olioita käyttävä ohjelma, koodi jatkuu

```
def main():
    nimi1 = input("Anna 1. opiskelijan nimi: ")
    op_nro1 = input("Anna 1. opiskelijan numero: ")
    kurssilainen1 = Opiskelija(nimi1, op_nro1)
    nimi2 = input("Anna 2. opiskelijan nimi: ")
    op_nro2 = input("Anna 2. opiskelijan numero: ")
    kurssilainen2 = Opiskelija(nimi2, op_nro2)

    print("Anna 1. opiskelijan tenttiarvosana.")
    tentti1 = lue_kokonaisluku()
    kurssilainen1.muuta_tenttiarvosana(tentti1)
    print("Anna 1. opiskelijan harjoitusarvosana.")
    harjoitus1 = lue_kokonaisluku()
    kurssilainen1.muuta_harjoitusarvosana(harjoitus1)
```

## Opiskelija-olioita käyttävä ohjelma, koodi jatkuu

```
print("Anna 2. opiskelijan tenttiarvosana.")
tentti2 = lue_kokonaisluku()
kurssilainen2.muuta_tenttiarvosana(tentti2)
print("Anna 2. opiskelijan harjoitusarvosana.")
harjoitus2 = lue_kokonaisluku()
kurssilainen2.muuta_harjoitusarvosana(harjoitus2)

print("1. opiskelijan tiedot:")
print(kurssilainen1.kerro_opiskelijanumero())
print(kurssilainen1.kerro_nimi())
print("Tenttiarvosana:",
      kurssilainen1.kerro_tenttiarvosana())
print("Harjoitusarvosana:",
      kurssilainen1.kerro_harjoitusarvosana())
print("Kurssiarvosana:",
      kurssilainen1.laske_kokonaisarvosana())
```



## Opiskelija-olioita käyttävä ohjelma, koodi jatkuu

```
print("2. opiskelijan tiedot:")
print(kurssilainen2.kerro_opiskelijanumero())
print(kurssilainen2.kerro_nimi())
print("Tenttiarvosana:",
      kurssilainen2.kerro_tenttiarvosana())
print("Harjoitusarvosana:",
      kurssilainen2.kerro_harjoitusarvosana())
print("Kurssiarvosana:",
      kurssilainen2.laske_kokonaisarvosana())
```

```
main()
```

# Merkkijonoesitys oliosta

- ▶ Ohjelmissa halutaan hyvin usein tulostaa jonkin oliion kaikkien kenttien arvot.
- ▶ Tulostaminen helpottuu, jos luokkaan määritellään metodi, joka tekee oliosta merkkijonoesityksen.
- ▶ Metodi palauttaa merkkijonon, joka sisältää oliion kenttien arvot tai muuta haluttua tietoa oliosta.
- ▶ Tämän metodin nimeksi annetaan `__str__`. Tällöin oliion tiedot voi tulostaa (esimerkiksi pääohjelmassa) käyttämällä suoraan oliioon viittaavan muuttujan nimeä, esimerkiksi

```
print(kurssilainen1)
```

## Merkkijonoesitys Opiskelija-olioista

```
def __str__(self):  
    mjono = self.__nimi + ", " + \  
            self.__opiskelijanumero + \  
            ", tenttiarvosana: " + \  
            str(self.__tenttiarvosana) + \  
            ", harjoitusarvosana: " + \  
            str(self.__harjoitusarvosana)  
    return mjono
```

## Opiskelijoiden tietojen tulostus, koodi

```
print("1. opiskelijan tiedot:")
print(kurssilainen1)
print("Kurssiarvosana:",
      kurssilainen1.laske_kokonaisarvosana())

print("2. opiskelijan tiedot:")
print(kurssilainen2)
print("Kurssiarvosana:",
      kurssilainen2.laske_kokonaisarvosana())
```

## Luokka ja pääohjelma eri moduuleissa

- ▶ Käytännössä käytetään usein ohjelmia, jotka koostuvat useista eri luokista.
- ▶ Yleensä on selvintä kirjoittaa kukin luokka omaan moduuliinsa.
- ▶ Jos luokka `Opiskelija` on tallennettu tiedostoon `opiskelija.py` ja sen olioita käyttävä pääohjelma (tai muu ohjelma) toiseen moduuliin, pitää pääohjelmamoduulin alkuun kirjoittaa

```
import opiskelija
```

- ▶ `Opiskelija`-olioita luodessa pitää luokan nimen edessä käyttää moduulin nimeä:

```
kurssilainen1 = opiskelija.Opiskelija(nimi1, op_nro1)
```

```
kurssilainen2 = opiskelija.Opiskelija(nimi2, op_nro2)
```

## Toinen tapa käyttää import-käskyä

- ▶ Käskyä `import` voi käyttää myös toisella tavalla:  
`from opiskelija import *`
- ▶ Tätä tapaa käytettäessä ei luokan olioita luodessa tarvitse kirjoittaa moduulin nimeä luokan nimen eteen:

```
kurssilainen1 = Opiskelija(nimi1, op_nro1)
```

```
kurssilainen2 = Opiskelija(nimi2, op_nro2)
```

## Olio metodin parametrina: luokka Tasovektori

- ▶ Kirjoitetaan luokka kaksiulotteisen vektorin  $a\vec{i} + b\vec{j}$  kuvaamiseen.
- ▶ Kuvataan kertoimia  $a$  ja  $b$  kentillä `__x_kerroin` ja `__y_kerroin`.
- ▶ Määritellään metodit mm. vektorin pituuden ja kahden vektorin pistetulon laskemiseen.
- ▶ Pistetulon laskeva metodi tarvitsee tiedon kahdesta eri `Tasovektori`-oliosta.
- ▶ Toinen olio yksilöidään metodin kutsussa ennen pistettä ja metodin nimeä, toinen annetaan metodille parametrina.

## Parametriolion kenttien arvon selvittäminen

- ▶ Parametrina saadun olion kentät voidaan selvittää joko pistenotaation avulla

```
def pistetulo(self, toinen_vektori):  
    tulo = self.__x_kerroin * toinen_vektori.__x_kerroin + \  
           self.__y_kerroin * toinen_vektori.__y_kerroin  
    return tulo
```

- ▶ Toinen vaihtoehto on käyttää luokan metodeita:

```
def pistetulo2(self, toinen_vektori):  
    tulo = self.__x_kerroin * \  
           toinen_vektori.kerro_x_kerroin() + \  
           self.__y_kerroin * \  
           toinen_vektori.kerro_y_kerroin()  
    return tulo
```



## Luokka Tasovektori, koodi

```
import math

class Tasovektori:

    def __init__(self, eka_kerroin, toka_kerroin):
        self.__x_kerroin = eka_kerroin
        self.__y_kerroin = toka_kerroin

    def kerro_x_kerroin(self):
        return self.__x_kerroin

    def kerro_y_kerroin(self):
        return self.__y_kerroin
```

## Luokka Tasovektori, koodi jatkuu

```
def laske_pituus(self):  
    return math.sqrt(self.__x_kerroin ** 2 + \  
                      self.__y_kerroin ** 2)  
  
def kerro_luvulla(self, kertoja):  
    self.__x_kerroin *= kertoja  
    self.__y_kerroin *= kertoja  
  
def pistetulo(self, toinen_vektori):  
    tulo = self.__x_kerroin * \  
           toinen_vektori.__x_kerroin + \  
           self.__y_kerroin * \  
           toinen_vektori.__y_kerroin  
    return tulo
```

## Luokka Tasovektori, koodi jatkuu

```
def __str__(self):
    if self.__y_kerroin >= 0:
        miono = "{:.3f}i + {:.3f}j".format(\
            self.__x_kerroin, self.__y_kerroin)
    else:
        miono = "{:.3f}i - {:.3f}j".format(\
            self.__x_kerroin, - self.__y_kerroin)
    return miono
```

## Esimerkki Tasovektori-olioiden käytöstä

- ▶ Seuraavien kalvojen pääohjelma luo kaksi Tasovektori-oliota ja kutsuu niille eri metodeita.
- ▶ Käsiteltävien vektoreiden tiedot pyydetään käyttäjältä.
- ▶ Desimaalilukujen lukemista varten on jälleen määritelty oma apufunktio.

## Esimerkki vektoreiden käsittelystä, koodi

```
import tasovektori

def lue_desimaaliluku():
    luku_onnistui = False
    while not luku_onnistui:
        try:
            luku = float(input())
            luku_onnistui = True
        except ValueError:
            print("Virheellinen desimaaliluku!")
            print("Anna uusi!")
    return luku
```

## Esimerkki vektoreiden käsittelystä, koodi jatkuu

```
def main():
    print("Anna ensimmäisen vektorin i-kerroin.")
    i_kerroin = lue_desimaaliluku()
    print("Anna ensimmäisen vektorin j-kerroin.")
    j_kerroin = lue_desimaaliluku()
    a_vektori = tasovektori.Tasovektori(i_kerroin, j_kerroin)
    print("Antamasi vektori on", a_vektori)

    print("Anna toisen vektorin i-kerroin.")
    i_kerroin = lue_desimaaliluku()
    print("Anna toisen vektorin j-kerroin.")
    j_kerroin = lue_desimaaliluku()
    b_vektori = tasovektori.Tasovektori(i_kerroin, j_kerroin)
    print("Antamasi vektori on", b_vektori)
```

## Esimerkki vektoreiden käsittelystä, koodi jatkuu

```
pituus1 = a_vektori.laske_pituus()
pituus2 = b_vektori.laske_pituus()
print("Vektorin {:s} pituus on {:.3f}".format(\
      str(a_vektori), pituus1))
print("Vektorin {:s} pituus on {:.3f}".format(\
      str(b_vektori), pituus2))

laskettu_tulo = a_vektori.pistetulo(b_vektori)
print("Vektoreiden pistetulo on {:.3f}.".format(\
      laskettu_tulo))
print("Milla luvulla ensimmäinen vektori kerrotaan?")
kerroin = lue_desimaaliluku()
a_vektori.kerro_luvulla(kerroin)
print("Eka vektori kertomisen jälkeen", a_vektori)
```

```
main()
```