Ohjelmoinnin peruskurssi Y1

CS-A1111

3.10.2018

Oppimistavoitteet: tämän luennon jälkeen

- Tiedät, miten ohjelma voidaan jakaa pienempiin osiin käyttämällä funktioita.
- Osaat määritellä funktion.
- Osaat kutsua funktiota eli kirjoittaa muualle ohjelmaan käskyn, joka saa aikaan funktion suorittamisen.
- Tiedät, miten funktiolle voi antaa lähtötietoja parametrien avulla.
- Tiedät, miten funktio voi välittää muulle ohjelmalle tiedon laskemistaan arvoista tms. paluuarvojen avulla.
- Voit luennon aikana lähettää kysymyksiä ja kommentteja sivulla `http://presemo.aalto.fi/y1s2018`

Funktiot

- Todellisessa elämässä tarvittavat ohjelmat ovat usein tuhansien tai kymmenien tuhansien rivien mittaisia.
- Jos koko ohjelma muodostuu tuolloin yhdestä pääohjelmasta, on sen rakenteen ja toiminnan hahmottaminen vaikeaa.
- Lisäksi ohjelmissa tehdään usein sama asia monta kertaa.
- Ratkaisu: käytetään funktioita.
- Funktio on ohjelmakoodin osa, jolle on annettu oma nimi.
- Funktion nimeä käyttämällä voidaan *kutsua* funktiota eli pyytää funktiota suoritettavaksi muualta ohjelmasta.

Funktioiden käytön etuja

- Ohjelmakoodi selkiytyy.
- Saman asian tekevä koodi joudutaan kirjoittamaan vain kerran.
- Ohjelman ylläpito helpottuu.
- Ohjelman testaus helpottuu.
- Ohjelman kirjoittaminen ryhmätyönä helpottuu.
- Ohjelman osia on helpompi käyttää uudelleen toisissa ohjelmissa.

Esimerkki: kolmioiden tulostus

- Halutaan kirjoittaa ohjelma, joka tulostaa seuraavan kuvion.

```
  *
 ***
*****
  *
 ***
*****
  *
 ***
*****
```

- Kirjoitetaan funktio yhden kolmion tulostamiseen.

Esimerkin koodi

```
def tulosta_kolmio():
    print("  *  ")
    print(" *** ")
    print("*****")


def main():
    tulosta_kolmio()
    tulosta_kolmio()
    tulosta_kolmio()


main()
```

Toinen versio: kutsut toistokäskyn sisällä

```
def tulosta_kolmio():
    print("  *  ")
    print(" *** ")
    print("*****")


def main():
    KERRAT = 3
    for i in range(KERRAT):
        tulosta_kolmio()


main()
```

Parametrit

- Halutaan kirjoittaa ohjelma, jolle annetaan massa paunoina ja unsseina ja joka tulostaa saman massan grammoina.
- Muunnoksen laskeminen sopii hyvin omaksi funktioksi.
- Tarvitaan kuitenkin jokin tapa kertoa funktiolle lähtötietoina annetut paunat ja unssit.
- Tieto voidaan välittää *parametrien* avulla.
- Parametri on funktion otsikossa sulkujen sisällä annettu nimi, jota voi käyttää funktion sisällä kuin mitä tahansa muuttujaa.
- Kun funktiota kutsutaan, määrätään parametrille tuleva alkuarvo.

Massamuunnos

```
def muunna_grammoiksi(paunat, unssit):
    UNSSIKERROIN = 28.35
    PAUNAKERROIN = 16 * UNSSIKERROIN
    massa = PAUNAKERROIN * paunat + UNSSIKERROIN * unssit
    print("Massa on grammoina {:.1f} g".format(massa))

def main():
    print("Paljonko on 6 paunaa 5 unssia?")
    muunna_grammoiksi(6, 5)
    print("Enta 22 paunaa 0 unssia?")
    muunna_grammoiksi(22, 0)

main()
```

Massamuunnos, parempi pääohjelma

```
def main():
    print("Ohjelma muuntaa massoja grammoiksi.")
    rivi = input("Montako paunaa?\n")
    paunamaara = int(rivi)
    rivi = input("Montako unssia?\n")
    unssimaara = int(rivi)
    muunna_grammoiksi(paunamaara, unssimaara)

main()
```

Vielä parametreista

- Parametrit saavat funktion kutsussa annetut alkuarvot samassa järjestyksessä kuin parametrit ovat funktion otsikossa.
- Funktion kutsussa parametrina arvo voidaan antaa minä tahansa lausekkeena, jonka arvo voidaan laskea, esimerkiksi:
  - suoraan lukuarvo
  - muuttuja
  - monimutkaisempi lauseke
- Esimerkkejä

```
muunna_grammoiksi(8, 2)
muunna_grammoiksi(paunamaara, unssimaara)
muunna_grammoiksi(2 * paunamaara, unssimaara - 5)
```

Välitehtävä 1

Vastaa sivulla http://presemo.aalto.fi/y1s2018

- Mitä seuraava ohjelma tulostaa?

```
def tulosta_tuplana(luku):
    tulos = 2 * luku
    print(tulos)

def main():
    numero = 3
    luku = 5
    tulosta_tuplana(numero)

main()
```

Toinen esimerkki: monipuolisempi kolmio

- Kirjoita funktio, joka tulostaa alla olevan mallin mukaisen kolmion

```
    %
   %%%
  %%%%%
 %%%%%%%
%%%%%%%%%
```

- Kolmion korkeus ja tulostuksessa käytettävä merkki annetaan funktion parametreina.

Monipuolisempi kolmio, koodi

```
def piirra_kolmio(korkeus, merkki):
    tyhjaa_perakkain = korkeus - 1
    merkkeja_perakkain = 1
    for i in range(korkeus):
        for j in range(tyhjaa_perakkain):
            print(" ", end = "")
        for j in range(merkkeja_perakkain):
            print(merkki, end = "")
        print()
        tyhjaa_perakkain -= 1
        merkkeja_perakkain += 2
```

Monipuolisempi kolmio, pääohjelma

```
def main():
    rivi = input("Anna kolmion korkeus.\n")
    kolmion_korkeus = int(rivi)
    piirrosmerkki = input("Anna kaytettava merkki.\n")
    piirra_kolmio(kolmion_korkeus, piirrosmerkki)


main()
```

Arvon palauttavat funktiot

- Halutaan tieto funktion laskemasta arvosta muualle ohjelmaan.
- Esimerkki: verkkokaupassa on alennusmyynti. Jokaisesta vähintään 100 euroa maksavasta tuotteesta saa 30 % alennuksen ja alle 100, mutta vähintään 20 euroa maksavasta tuotteesta 10 % allennuksen.
- Ohjelma pyytää käyttäjän ostamien tuotteiden alkuperäiset hinnat ja laskee niiden yhteishinnan alennusten jälkeen.
- Kirjoitetaan funktio, joka saa parametrina yhden tuotteen alkuperäisen hinnan ja laskee sen alennetun hinnan.
- Funktiossa laskettu arvo pitää se saada jotenkin tietoon funktion ulkopuolelle.
- Funktio voi välittää tiedon laskemastaan arvosta *palauttamalla* tämän arvon.

Arvon palauttaminen

- Arvo palautetaan `return`-käskyllä:

  ```
  return lauseke
  ```

- Palautettu arvo voidaan ottaa sijoituskäskyllä talteen siellä, missä funktiota kutsuttiin:

  ```
  muuttuja = funktio(parametrit)
  ```

- Palautetun arvon voi myös tulostaa suoraan esimerkiksi `print`-käskyssä:

  ```
  print("Tulos on", funktio(parametrit))
  ```

- Palautettua arvoa voi myös käyttää hyväksi suoraan toisen lausekkeen arvoa laskettaessa:

  ```
  uusi_tulos = 2 * funktio(parametrit) - 5
  ```

Huomaa

- Arvon palauttaminen ja arvon tulostaminen ovat täysin eri asiat.
- Arvon *tulostaminen* tarkoittaa sitä, että ohjelma tulostaa arvon näkyviin esimerkiksi kuvaruudulle.
- Arvon *palauttaminen* ei vielä tulosta arvoa minnekään näkyviin. Se vain välittää funktion laskeman arvon käytettäväksi sinne, missä funktiota kutsuttiin.

Hintalaskuri, koodi

```
def laske_alennettu_hinta(hinta):
    RAJA1 = 100.0
    RAJA2 = 20.0
    ISO_ALENNUS = 30.0
    PIENI_ALENNUS = 10.0
    if hinta >= RAJA1:
        alennus = ISO_ALENNUS / 100.0 * hinta
    elif hinta >= RAJA2:
        alennus = PIENI_ALENNUS / 100.0 * hinta
    else:
        alennus = 0.0
    uusi_hinta = hinta - alennus
    return uusi_hinta
```

Hintalaskuri, koodi jatkuu

```
def main():
    yhteensa = 0.0
    print("Anna tuotteiden hinnat, lopeta negatiivisella.")
    syote = input("Seuraava hinta (eur): ")
    luettu_hinta = float(syote)
    while luettu_hinta >= 0.0:
        maksettu_hinta = laske_alennettu_hinta(luettu_hinta)
        yhteensa += maksettu_hinta
        syote = input("Seuraava hinta (eur): ")
        luettu_hinta = float(syote)
    print("Yhteishinta {:.2f} eur.".format(yhteensa))

main()
```

Välitehtävä 2

Vastaa sivulla http://presemo.aalto.fi/y1s2018

- Mitä seuraava ohjelma tulostaa?

```
def muuta_jotain(arvo):
    luku = 10
    tulos = 3 * arvo
    return tulos

def main():
    maara = 15
    arvo = 7
    luku = muuta_jotain(maara)
    print(luku)

main()
```

Sisäkkäiset funktiokutsut

- Funktion kutsussa funktiolle voi antaa parametrina toisen funktion kutsun.
- Sisempi funktio suoritetaan ensin, ja sen paluuarvoa käytetään ulomman funktion parametrin arvona.
- Esimerkki (`float` ja `input` ovat Pythonin valmiita funktioita):

```
luettu_hinta = float(input("Seuraava hinta (eur): "))
```

Moduuli `math`

- Pythonissa on valmiina suuri joukko funktioita erilaisten toimintojen tekemiseen.
- Suurin osa näistä funktioista on jaettu moduuleihin. Yksi moduuli sisältää tyypillisesti samaan asiaan liittyviä funktioita ja mahdollisesti myös vakioita.
- Moduuli `math` sisältää joukon matemaattisia funktioita sekä vakiot `math.pi` ja `math.e`.
- Jotta moduulin vakioita tai funktioita voisi käyttää, on ohjelmatiedoston alkuun kirjoitettava

```
import math
```

Tärkeitä `math`-moduulin funktioita

`ceil(x)`	pienin kokonaisluku, joka on $\geq$ `x`.
`floor(x)`	suurin kokonaisluku, joka on $\leq$ `x`.
`trunc(x)`	`x`:n kokonaisosa ilman pyöristyksiä.
`sqrt(x)`	neliöjuuri
`exp(x)`	e potenssiin `x`.
`log(x)`	luonnollinen logaritmi.
`log10(x)`	10-kantainen logaritmi.
`cos(x)`	kosini.
`sin(x)`	sini.
`tan(x)`	tangentti.

Esimerkki: toisen asteen yhtälön ratkaisu

- Kirjoitetaan ohjelma, joka ratkaisee toisen asteen yhtälön $ax^2 + bx + c = 0$
- Käytetään ratkaisukaavaa

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Ohjelma tulostaa vain reaalijuuret.
- Ratkaisujen lukumäärää tutkitaan neliöjuuren alla olevan diskriminantin arvon perusteella.

Toisen asteen yhtälö, koodi

```
import math

def ratkaise_yhtalo(a, b, c):
    diskrim = b * b - 4 * a * c
    if diskrim < 0:
        print("Ei reaalijuuria.")
    elif diskrim == 0:
        x = -1.0 * b / (2 * a)
        print("Yhtalon ratkaisu on {:.2f}.".format(x))
    else:
        x1 =  (-1.0 * b  + math.sqrt(diskrim) ) / (2 * a)
        x2 =  (-1.0 * b  - math.sqrt(diskrim) ) / (2 * a)
        print("Ratkaisut ovat {:.2f} ja {:.2f}.".format(
              x1, x2))
```

Toisen asteen yhtälö, koodi jatkuu

```
def main():
    print("Anna 2. asteen yhtalon kertoimet a, b ja c.")
    eka_kerroin = int(input())
    toka_kerroin = int(input())
    vakio = int(input())
    if eka_kerroin == 0:
        print("Yhtalo ei ole toista astetta.")
    else:
        ratkaise_yhtalo(eka_kerroin, toka_kerroin, vakio)


main()
```