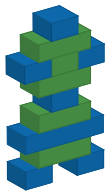
Valmiiden kirjastojen käyttö

Ohjelmoinnin perusteet

8



Turun yliopisto
University of Turku

Valmiit kirjastot

- Aina ohjelmoijan ei tarvitse keksiä pyörää uudestaan: useiden ohjelmointikielten mukana tulee valmiita rutiineja erilaisiin tarpeisiin.

- Ohjelmaa kirjoittaessa onkin hyvä tarkistaa ohjelmointikielen dokumentaatiosta tai Googlella löytyykö tarkoitukseen jo valmis rutiini.

Valmiiden rutiinien etuja

- Ohjelmointikielten mukana tulevat rutiinit ovat yleensä hyvin **testattuja** ja **optimoituja**. Ne siis todennäköisesti toimivat nopeasti ja virheettä ainakin useimmissa tilanteissa.

- Lisäksi ne toteuttavat usein tehtäviä (esimerkiksi verkkoyhteyksiin tai grafiikkaan liittyviä), joiden oma toteutus on työlästä.

Valmiit kirjastot Pythonissa

- Lista Pythonin mukana tulevista kirjastoista löytyy Pythonin dokumentaatiosta (tosin englanniksi):

  https://docs.python.org/2/library/

Esimerkkejä kirjastoista

- Tänään käydään esimerkkinä läpi seuraavat kirjastot:

- `random` – satunnaislukujen hallintaan
- `math` – matemattisia operaatioita
- `string` – erilaisia hyödyllisiä merkkijonovakioita

Moduulien käyttö

- Ennen kuin kirjastoa voi käyttää, se pitää avata oman ohjelman käyttöön. Tämä tapahtuu Pythonissa `import`-lauseella. Syntaksi:

```
import <moduulin nimi>
```

...tai

```
from <moduulin nimi> import <rutiinit>
```

Esimerkiksi

- Otetaan käyttöön `random`-kirjasto kokonaisuudessaan:

```
import random
```

- Otetaan käyttöön `random`-kirjastosta vain operaatiot `randint` ja `choice`:

```
from random import randint, choice
```

import-lauseen käytöstä

- On hyvä sijoittaa `import`-lauseet aina heti ohjelman alkuun (eli ennen aliohjelmien määrittelyä).

- Tällä tavalla ohjelmakoodia lukiessa on heti alusta asti selvää, mistä kirjastoista ja mitä rutiineja ohjelman käyttöön on avattu.

Operaatioiden käyttö

- Operaatioiden käyttö avatusta kirjastosta riippuu siitä, miten kirjasto avattiin. Jos avataan koko kirjasto import-lauseella, täytyy operaatioon viitatessa käyttää myös kirjaston nimeä:

```
import random

luku = random.randint(1,10)
```

Operaatioiden käyttö (2)

- Jos otetaan käyttöön tietty operaatio (tai operaatiot) `from...import` -rakenteella, ei kirjaston nimeä tarvitse käyttää operaatiota käytettäessä:

```
from random import randint

luku = randint(1,10)
```

Kirjasto random

- Kirjastoa `random` tarvitaan satunnaislukujen luomiseen ja muihin satunnaisoperaatioihin.

- Huomaa, että Pythonin (tai yleensä tietokoneen) luomat satunnaisluvut eivät ole aitoja satunnaislukuja vaan niin sanottuja pseudosatunnaislukuja. Yleensä luvut ovat kuitenkin *riittävän satunnaisia*.

Kirjastosta random löytyviä operaatioita

- Kirjastosta löytyvät esimerkiksi seuraavat operaatiot:

Operaatio	Tarkoitus	Esimerkki
`randint(a,b)`	Arpoo satunnaisen luvun väliltä [a, b]	`luku = randint(1,10)`
`choice(lista)`	Arpoo satunnaisen alkion annetulta listalta (tai muusta iteroitavasta rakenteesta)	`lista = range(1,10)` `luku = choice(lista)`

Esimerkki

- Ohjelma arpoo 7 lukua väliltä [1, 39], ja sijoittaa ne listaan joka lopuksi palautetaan:

```
from random import randint

def lottoNumerot():
    lista = [] # Tyhjä lista
    for i in range(0,7): # 7 iteraatiota
        lista.append(randint(1,39))
    return lista

# Testi
numerot = lottoNumerot()
print numerot
```

Esimerkki 2

- Funktio poimii oppilaslistasta kaksi "vapaaehtoista" esittelemään ratkaisunsa:

```
from random import choice

def poimiEsittajat(oppilasLista):
    opp1 = choice(oppilasLista)
    # Varmista, ettei samaa valita kahdesti
    opp2 = opp1
    while opp1 == opp2: # Toista niin kauan kun samat
        opp2 = choice(oppilasLista) # Arvo 2. esittäjä
    return [opp1, opp2] # palauta listana

# Testi
oppilaat = ["Arto","Anna","Pekka","Jarmo","Kimmo","Liisa"]
valitut = poimiEsittajat(oppilaat)
print valitut
```

Kirjasto math

- Kirjastosta math löytyy lukuisia matemaattisia operaatioita eri tarpeisiin, esimerkiksi

Operaatio	Tarkoitus	Esimerkki
`sqrt(x)`	Palauttaa luvun x neliöjuuren	`juuri = sqrt(16)`

`sin(x)`
`cox(x)`
`tan(x)`
	Palauttaa x:n sinin, kosinin tai tangentin, kun x on annettu radiaaneissa	`c = cos(2)`

`radians(x)` `degrees(y)`	Muuttaa x astetta radiaaneiksi tai y radiaania asteiksi	`deg = 90` `rad = radians(deg)`
`log(x, k)`	Palauttaa x:n k-kantaisen logaritmin	`print log(16,2) # 4.0`
`pi`	Palauttaa piin likiarvon	`print pi`

Esimerkki

- Funktio laskee ympyrän kehän pituuden annetun säteen mukaan:

```
from math import pi

def ympyranKeha(r):
    return 2 * pi * r

# Testi
sade = input("Anna ympyrän säde: ")
keha = ympyranKeha(sade)
print "Kehä on", keha
```

Kirjasto string

- Useimmat merkkijono-operaatiot (kuten alijonon poimiminen, etsiminen tai korvaaminen) löytyvät Pythonista ilman ulkoisen kirjaston avaamista.

- Kirjastosta `string` löytyy kuitenkin eräitä hyödyllisiä **vakioita** (vakio = valmiiksi määritelty muuttuja)

Kirjasto string

- Esimerkkejä vakioista:

Operaatio	Tarkoitus	Esimerkki
`ascii_lowercase`	Kaikki pienet kirjaimet [a…z]	`a = ascii_lowercase`
`ascii_uppercase`	Kaikki isot kirjaimet [A…Z]	`b = ascii_uppercase`
`ascii_letters`	Edellisten yhdistelmä	`ab = ascii_letters`
`punctuation`	Tyypilliset välimerkit	`d = punctuation`

Esimerkki

- Funktio, joka poistaa annetusta merkkijonosta välimerkit:

```
from string import punctuation

def poistaValimerkit(mjono):
    tulos = ""
    for merkki in mjono:
    # in-operaattori palauttaa True, jos
    # annettu alkio löytyy jonosta / listasta.
    if merkki not in punctuation:
        tulos = tulos + merkki
    return tulos

# Testi
mj = "Moi, tämä on 'testi'.!?"
print poistaValimerkit(mj)
```